



**UNIVERSITAT POLITÈCNICA  
DE CATALUNYA  
BARCELONATECH**

Proyecto Final de Carrera  
**INGENIERÍA INDUSTRIAL**

RosPiBot

Restauración y mejora de una plataforma  
robótica incorporando un SBC y ROS

Memoria

Autor: Joan Guasch Iglesias  
Directores: Cecilio Angulo Bahon y Manel Velasco Garcia  
Convocatoria: Junio 2014

Escuela Técnica Superior de Ingeniería Industrial de Barcelona





## Resumen

Este proyecto consiste en el aprovechamiento de la plataforma robótica comercial WifiBot que corre el riesgo de quedarse obsoleta y rescatarla de su destino. Para ello se hace uso de una Raspberry Pi, un ordenador de placa reducida que ha revolucionado el mundo de la automatización desde el día de su aparición en el 2012, como unidad de procesamiento de la nueva plataforma y que está basada en Linux. En el apartado de software, además de incorporar herramientas que faciliten el trabajo a futuros usuarios se ha configurado para poder trabajar con ROS, una infraestructura digital para el desarrollo de software de robots creada por Willow Garage y extensamente utilizada en este sector.

A lo largo de este documento se exponen el estado inicial del robot, los nuevos requerimientos a cumplir, las modificaciones efectuadas incluyendo las complicaciones encontradas a lo largo del proyecto y el resultado final obtenido. Haciendo especial hincapié en el desarrollo de soluciones para cumplir los objetivos del proyecto, tales como el diseño de circuitos electrónicos, la integración de los nuevos componentes o la creación de una mejor interficie para el usuario.



# Índice

	Página
<b>Resumen</b>	<b>1</b>
<b>Índice de figuras</b>	<b>6</b>
<b>Índice de tablas</b>	<b>8</b>
<b>Acrónimos</b>	<b>9</b>
<b>1. Introducción</b>	<b>10</b>
1.1. Objetivos . . . . .	11
1.2. Alcance . . . . .	12
<b>2. Descripción</b>	<b>13</b>
2.1. WifiBot . . . . .	13
2.1.1. Características . . . . .	13
2.1.2. Estructura . . . . .	14
2.1.3. Grupo Motriz . . . . .	14
2.1.4. Codificador óptico de cuadratura . . . . .	15
2.1.5. Estado Inicial . . . . .	16
2.2. Raspberry Pi . . . . .	17
2.2.1. Características Técnicas . . . . .	17
2.2.2. Sistemas Operativos . . . . .	18
2.2.3. Programación . . . . .	19
2.2.4. GPIO . . . . .	19
2.3. ROS . . . . .	20
<b>3. Especificaciones</b>	<b>22</b>
3.1. Software . . . . .	22
3.1.1. GitHub . . . . .	22
3.1.2. Librerías . . . . .	22
3.2. Electrónica . . . . .	23
3.2.1. Funcionamiento . . . . .	23
3.2.2. Alimentación . . . . .	23
3.2.3. Lógica . . . . .	23
3.2.4. Comunicación . . . . .	23
3.3. Mecánica . . . . .	24
3.3.1. Estructura . . . . .	24

3.3.2. Tornilleria . . . . .	24
<b>4. Modificaciones</b>	<b>25</b>
4.1. Componentes Electrónicos . . . . .	25
4.1.1. Motorización . . . . .	25
4.1.2. Sensores de proximidad . . . . .	26
4.1.3. Batería . . . . .	27
4.1.4. Punto de acceso WiFi . . . . .	28
4.1.5. Conmutador de red . . . . .	28
4.1.6. Descodificador de cuadratura . . . . .	29
4.1.7. Adaptador de nivel de tensión . . . . .	30
4.1.8. Lectura de valores analógicos . . . . .	31
4.1.9. Monitorización de la Batería . . . . .	31
4.1.10. Alimentación de lógica . . . . .	32
4.1.11. Control de motores . . . . .	33
4.2. Integración de los componentes . . . . .	33
4.2.1. SecurePi . . . . .	33
4.2.2. SensorPi . . . . .	35
4.2.3. MotorPi . . . . .	35
4.3. Conexionado . . . . .	37
4.3.1. Potencia . . . . .	37
4.3.2. Alimentación de lógica . . . . .	37
4.3.3. Cableado SensorPi . . . . .	38
4.3.4. Comunicación . . . . .	39
4.4. Distribución interna . . . . .	40
4.4.1. Diseño preliminar . . . . .	40
4.4.2. Soportes . . . . .	41
4.4.3. Diseño Final . . . . .	42
4.5. Componentes mecánicos . . . . .	43
4.5.1. Mecanizados . . . . .	43
4.5.2. Paneles . . . . .	44
4.5.3. Tornillería . . . . .	45
4.6. Montaje . . . . .	46
4.6.1. Grupos individuales . . . . .	46
4.6.2. Conjunto . . . . .	47
4.7. Configuración Router WiFi . . . . .	48
4.8. Configuración Raspberry Pi . . . . .	49
4.8.1. Grabación Raspbian . . . . .	49
4.8.2. Configuración inicial . . . . .	49

4.8.3. Instalación de librerías . . . . .	50
4.9. Programas básicos . . . . .	54
4.10. Módulo Rospibot . . . . .	56
4.11. Instalación ROS Groovy . . . . .	58
4.11.1. Instalación básica . . . . .	58
4.11.2. Inicializar dependencias y librerías . . . . .	58
4.11.3. Creación directorio de trabajo . . . . .	59
4.11.4. Creación paquete "beginner_tutorials" . . . . .	59
<b>5. Presupuesto</b>	<b>60</b>
5.1. Coste Material . . . . .	60
5.1.1. Coste de Dispositivos . . . . .	60
5.1.2. Coste de la Electrónica . . . . .	61
5.1.3. Coste placa SecurePi . . . . .	61
5.1.4. Coste de la Mecánica . . . . .	62
5.1.5. Coste placa SensorPi . . . . .	62
5.1.6. Coste placa MotorPi . . . . .	63
5.2. Coste Personal . . . . .	64
5.2.1. Descripción de las labores . . . . .	64
5.3. Planificación . . . . .	65
<b>6. Impacto Ambiental</b>	<b>66</b>
<b>7. Conclusiones</b>	<b>67</b>
<b>8. Agradecimientos</b>	<b>68</b>
<b>Referencias</b>	<b>69</b>
<b>Soporte informático</b>	<b>71</b>

# Índice de figuras

2.1.	Wifi Bot 4G . . . . .	13
2.2.	Dimensiones [mm] motor HN-GH7.2-2414T-50:1 . . . . .	14
2.3.	Gráfica de curvas específicas del motor HN-GH7.2-2414T-50:1 . . . . .	15
2.4.	Esquema de funcionamiento de un encoder de cuadratura . . . . .	15
2.5.	Raspberri Pi . . . . .	17
2.6.	Esquema Modelo B . . . . .	17
2.7.	Icono de S.O. Raspbian . . . . .	18
2.8.	Nomenclatura del GPIO . . . . .	19
2.9.	Logotipo de ROS . . . . .	20
2.10.	Gráfico ejemplo de la comunicación entre nodos. . . . .	20
2.11.	Simulación del robot PR2 en Gazebo y publicación en Rviz. . . . .	21
4.1.	Sensor de distancia por ultrasonidos KS103 . . . . .	26
4.2.	Diagrama de radiación del sensor KS103 . . . . .	26
4.3.	Batería LiPo con PCM instalado . . . . .	27
4.4.	Router TP-Link WR702N . . . . .	28
4.5.	Switch antes y después de la modificación . . . . .	29
4.6.	Diagrama de pines LS7366 . . . . .	29
4.7.	Circuito adaptador de tensión . . . . .	30
4.8.	Diagrama BSS138 . . . . .	30
4.9.	Diagrama de pines MCP3428 . . . . .	31
4.10.	Diagrama de pines LM3914 . . . . .	31
4.11.	Circuito regulador Buck 5V . . . . .	32
4.12.	Guia de selección de inductancia . . . . .	32
4.13.	Diagrama de pines L298N . . . . .	33
4.14.	Layout y PCB de la placa SecurePi . . . . .	34
4.15.	SecurePi colocada con la Raspberry Pi . . . . .	34
4.16.	Layout y PCB de la placa SensorPi . . . . .	35
4.17.	Layout y PCB de la placa MotorPi . . . . .	36
4.18.	Esquema eléctrico del conexionado de potencia . . . . .	37
4.19.	Esquema eléctrico de la alimentación de lógica . . . . .	38
4.20.	Esquema eléctrico del cableado de la placa SensorPi . . . . .	38
4.21.	Diseño preliminar con la ubicación de todos los componentes . . . . .	40
4.22.	Conjunto de soportes orientados para impresión 3D . . . . .	41
4.23.	Acabado final delantero del robot . . . . .	42
4.24.	Acabado final trasero del robot . . . . .	43
4.25.	Modelo por ordenador de una barra longitudinal de aluminio. . . . .	44
4.26.	Modelo por ordenador de un acoplador de aluminio. . . . .	44

4.27. Conjunto de paneles obtenidos por corte láser. . . . .	45
4.28. Conjunto de paneles montados individualmente. . . . .	46
4.29. Acabado final de la plataforma RospiBot. . . . .	47
4.30. Configuración IP del router. . . . .	48
4.31. Configuración servidor DCHP. . . . .	48
4.32. Interfície del programa Win32 Disk Imager. . . . .	49
4.33. Assignación de IP por DHCP. . . . .	50
4.34. Consola de acceso remotor por SSH. . . . .	50
4.35. Menú de opciones de configuración de Raspbian. . . . .	51
4.36. Archivo de configuración de los dispositivos I <sup>2</sup> C y SPI. . . . .	52
4.37. Archivo de configuración de los módulos. . . . .	52
5.1. Diagrama de Gantt . . . . .	65

## Índice de tablas

2.1.	Ensayo de consumo y señal de codificador del motor sin carga . . . . .	16
4.1.	Comparativa entre tipos de baterías . . . . .	27
5.1.	Distribución del presupuesto de material . . . . .	60
5.2.	Coste detallado de cada dispositivo . . . . .	60
5.3.	Coste detallado de la electrónica . . . . .	61
5.4.	Coste detallado placa SecurePi . . . . .	61
5.5.	Coste detallado de la mecánica . . . . .	62
5.6.	Coste detallado placa SensorPi . . . . .	62
5.7.	Coste detallado placa MotorPi . . . . .	63
5.8.	Coste detallado del personal necesario . . . . .	64

## Acrónimos

**SoC** System on a Chip

**CPU** Central Processing Unit

**GPU** Graphics Processing Unit

**RAM** Random Access Memory

**RCA** Radio Corporation of America

**HDMI** High-Definition Multimedia Interface

**DSI** Display Serial Interface

**CSI** Camera Serial Interface

**SD** Secure Digital

**GPIO** General-purpose input/output

**UART** Universal Asynchronous Receiver-Transmitter

**TX** Transmitter

**RX** Receiver

**SPI** Serial Protocol Interface

**MOSI** Master Output Slave Input

**MISO** Master Input Slave Output

**SCLK** Serial Clock

**CE** Chip Enable

**I2C** Inter-Integrated Circuit

**SDA** Serial Data

**SCL** Serial Clock

**ADC** Analog to Digital Converter

**ROS** Robot Operative System

**SLAM** Simultaneous Localization And Mapping

**RoHS** Restriction of Hazardous Substances Directive

## 1. Introducción

Este proyecto surge de la observación de cómo una plataforma robótica robusta deja de ser utilizada por estudiantes a causa de los problemas derivados de su unidad de procesamiento y de la falta de información relacionada con el robot. El hecho de recurrir a software desactualizado y que ya no dispone de soporte por parte de su fabricante, ha sentenciado este robot a formar parte de unos activos que ya no volverán a ser utilizados.

En tiempos de escasez pero de avances tecnológicos, uno empieza a descubrir el potencial que esconde este tipo de plataformas. Mecánicamente siguen funcionando sin problemas, los motores siguen aportando sus mismas características que al principio y los componentes más básicos siguen utilizándose en la actualidad. Su único punto débil es un procesador desfasado (pero adelantado en su época).

Y es en el momento de buscar el proyecto, uno se pregunta tipo de trabajo quiere realizar. En este caso, el objetivo era crear un proyecto que tuviese continuidad después de su realización. Así pues, se ha considerado que la restauración de un plataforma robótica obsoleta puede convertirse en ese proyecto deseado. Lo solo se recuperá una herramienta educacional, sino que al mejorarla, permitirá a futuros estudiantes llevar a cabo nuevo trabajos.

## 1.1. Objetivos

El objetivo principal de este proyecto es recuperar la plataforma WifiBot. Para ello no solo se restaurará, sino que se mejorará para ofrecer una interacción más cómoda con el usuario. Tal y como se ha comentado en la introducción, el principal problema se centra en su procesador; Por ello el primer paso será buscar un substituto que pueda cumplir la labor. Además para evitar que se repita la situación. se va a elegir un componente que goce de una amplia repercusión, que disponga de una gran comunidad a su espalda y que no requiera de extensa documentación para poder trabajar con él.

Una vez mejorado el cerebro del robot, se atacarán otros punto débiles localizados durante el largo periodo de observación que se experimentó. Una respuesta luminosa o acústica permitiría al usuario conocer algunos estados de la plataforma como por ejemplo: Si los dispositivos están operativos, si hay comunicación entre ellos e incluso si dispone de carga en su batería.

Otro campo a mejorar sería el apartado de sensores para percibir información de su entorno. Ya que se trata de un robot diseñado en su origen para moverse por terreno abrupto, sería recomendable reforzar este sector y disponer de sensores acorde a diseño.

Con una plataforma actualizada, el siguiente tema que deberíamos centrar nuestros esfuerzos, sería en encontrar de que manera podemos hacer apetecible el robot para que atraiga la mirada y transmita ganas de experimentar. Ofrecer un conjunto de herramientas que permita un acceso a todo el robot pero que a la vez evite problemas indeseados.

## 1.2. Alcance

Puesto que el objetivo primordial es mejorar la unidad de procesamiento, se ha elegido que entre todas las opciones de introducir una Raspberry Pi. Esta placa cuenta con un gran número de periféricos que se adaptan perfectamente a las necesidades del robot. Como características técnicas más destacables, puede ejecutar un sistema operativo como Linux, cuenta con conectores de red y USB además de pines para múltiples funciones; Otra cualidad remarcable es que su disco duro es extraíble, de esta manera se puede manipular el sistema si la necesidad de estar obligatoriamente siempre conectado al robot.

En el caso de los sensores externos, un conjunto de sensores sonar, una cámara y un módulo IMU serán una buena opción a la hora de actualizar el robot. De esta manera se evita el problema de la interferencia de la luz solar en los sensores infrarrojos, además de ofrecer otras informaciones como la orientación del robot e incluso poder aplicar procesamiento de imagen.

Además de sensores, se hace necesaria una remodelación tanto del exterior como del interior del robot. Ofreciendo un sistema más cómodo a la hora de conectarse, tanto por cable como de modo inalámbrico. Aún así se mantendrá la estética de un robot formado por dos hemisferios los cuales pueden rotar respecto un eje transversal.

Para incitar al futuro usuario trabajar con la nueva plataforma, se ofrecerá un conjunto de funciones en lenguaje Python con las que obtener información del entorno y hacer mover el robot se convierta en una labor verdaderamente fácil. Para ello se crearán dos grupos de códigos. Los primeros se encargarán de la comunicación con los diferentes dispositivos y almacenarán la información. El segundo grupo tendrá la función de ofrecer herramientas para que el usuario sea capaz de recuperar la información guardada.

Finalmente, para darle un valor añadido al robot, se instalará ROS, este conjunto de herramientas está orientado para plataformas robóticas. El hecho de lanzar un núcleo de ROS dentro del robot ofrece un mayor número de posibilidades a futuros trabajos. En este proyecto únicamente se plantea la opción de instalar y configurar ROS. Para futuros proyectos se recomienda la lectura de libros que nos introduzcan en ROS. [1].

## 2. Descripción

### 2.1. WifiBot

**WifiBot** es una plataforma robótica, desarrollada por la empresa francesa **Nexter Robotics**, diseñada para poder navegar en múltiples escenarios gracias a su diseño. Su sistema de tracción a las cuatro ruedas, diseño reducido y bajo peso, le otorga una gran flexibilidad.

#### 2.1.1. Características

Este proyecto parte del modelo **WifiBot 4G** (figura 2.1, producido durante el periodo 2002–2006. Se'gun su guía de usuario [3] está formador por:



Figura 2.1: Wifi Bot 4G

- CPU:
  - Procesador AMD Au1500
  - 400MHz
  - Memória RAM de 64MB
  - Memória Flash de 32MB
- Interfícies:
  - 4x Ethernet 10/100
  - 1x USB
  - 1x I<sup>2</sup>C
  - 1x RS232
- WIFI:
  - WiFi con estándar 802.11a/b/g
  - Modos Access Point, Bridge, Client y Router
  - 1x Antena de 5dBi
- Motores:
  - Sensores:
    - 1x Cámara IP
    - 2x Sensores IR de distância (ADC)
    - 2x Codificadores óptico de cuadratura
    - 2x DSPIC30F2010
    - 1x Nivel de batería (ADC)

- 4x Motores de 7.2V
- Reductora  $i = 50 : 1$
- Par nominal 8.87Kg/cm
- Velocidad nominal 120Rpm
- Altura 20cm
- Peso 4.5Kg
- Dimensiones:
- Longitud 28cm
- Anchura 30cm
- Baterías:
- 9.6V NiMh (8 celdas)
- Capacidad 9500mAh
- Autonomía de 2 horas

### 2.1.2. Estructura

La estructura de la base está formada por dos secciones simétricas que llamaremos hemisferios izquierdo y derecho. Estas dos partes están unidas por una barra roscada que atraviesa transversalmente todo el robot, ofreciendo un eje de rotación entre los dos elementos, cualidad que le otorga una mayor adaptación a superficies irregulares.

### 2.1.3. Grupo Motriz

Los motores que componen el WifiBot son concretamente el modelo **HN-GH7.2-2414T-50:1** del fabricante **Hsiang Neng**. Este modelo está compuesto por un motor de continua de voltaje nominal de 7.2V y una reductora con relación 50:1. Tal y como se describe en la figura 2.2, dispone de un eje de 6mm de diámetro, una longitud de 81.1mm y un diámetro exterior de 37mm. Los datos se han extraído de la ficha técnica [?].

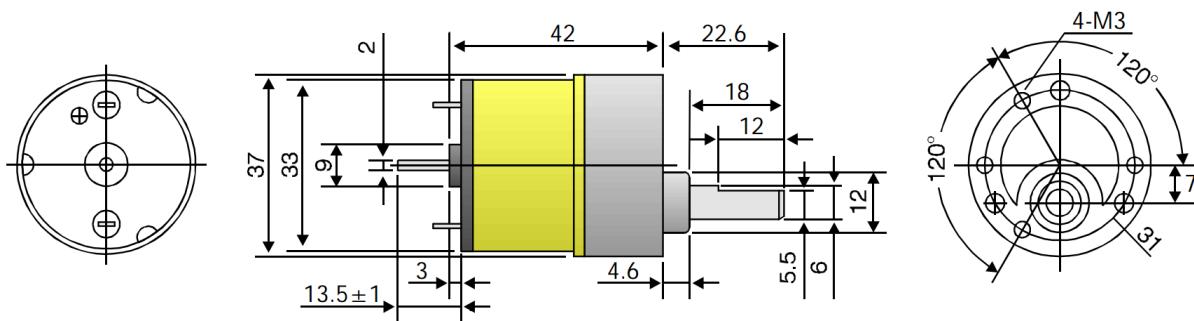


Figura 2.2: Dimensiones [mm] motor HN-GH7.2-2414T-50:1

En mediciones efectuadas, se determinó que la alimentación de los motores llegaba hasta los 8.2V con un consumo inferior a los 1.8A por motor (con carga), basándonos en la figura 2.3, podremos asegurar que el motor se encontraba ofreciendo un par de 4.7 Kg-cm.

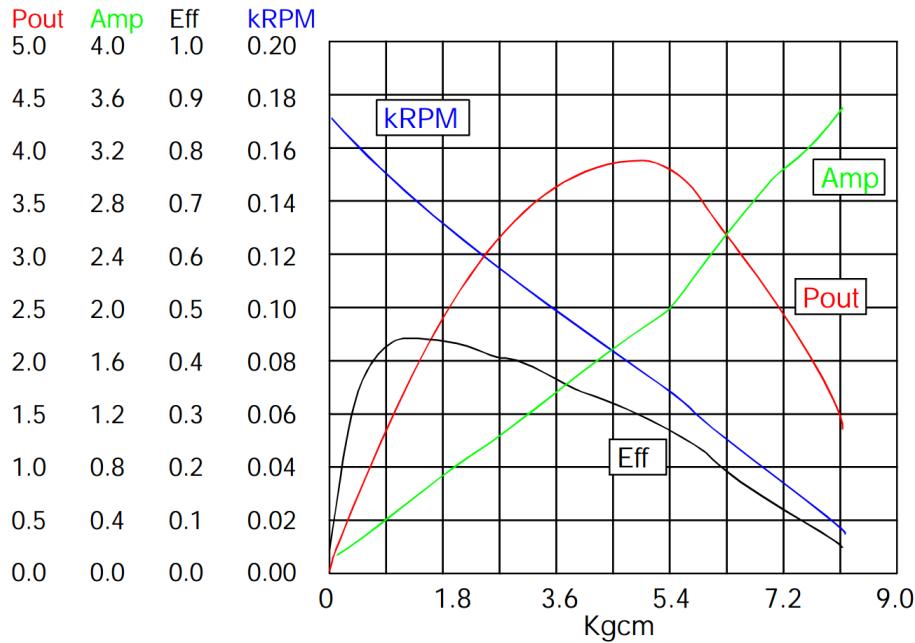


Figura 2.3: Gráfica de curvas específicas del motor HN-GH7.2-2414T-50:1

#### 2.1.4. Codificador óptico de cuadratura

Este tipo de sensor permite conocer tanto la velocidad como la posición de un eje. Tal y como se describe en la figura 2.4, su funcionamiento se basa en un disco ranurado que gira solidario al que se dispone un par de fototransistores. Estos transistores emiten una señal compuesta por dos canales, comúnmente llamados A y B, de onda cuadrada. Estos dos canales están desfasados un cuarto de fase, creando una secuencia que depende del sentido de giro.

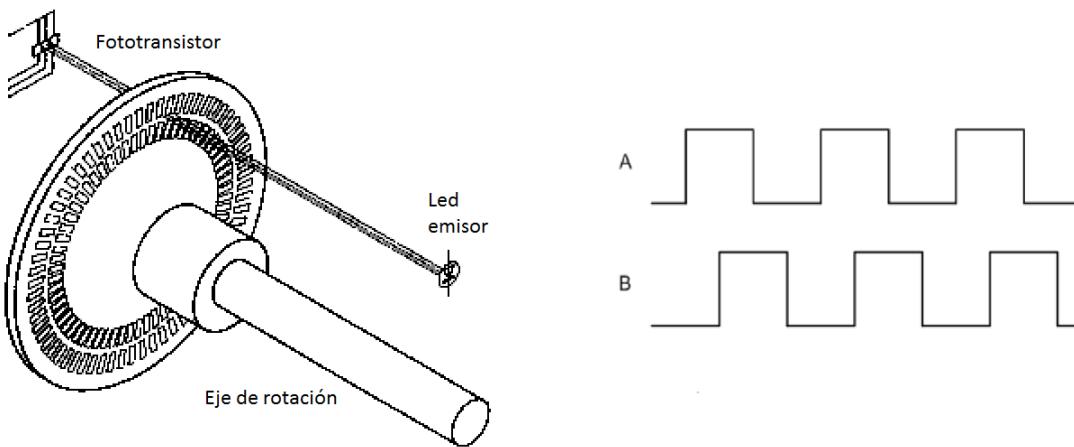


Figura 2.4: Esquema de funcionamiento de un encoder de cuadratura

De los 4 motores que conforman la plataforma, los dos delanteros disponen de un codificador óptico, concretamente el modelo **E4P-120-079-HT** del fabricante **US DIGITAL**. Seún si ficha técnica [4], este modelo ofrece unos 120 ciclos por revolución (o

480 interrupciones), está pensado para ejes de 2mm de diámetro y no dispone de canal de índice (permite corregir errores). Tras un ensayo del motor sin carga, cuyos datos se encuentran en la tabla 2.1, se determinó que se llegaba a un máximo de 80.000 pulsos por segundo.

Voltaje [V]	Consumo [A]	Período de Cuadratura [us]
0.5	0.07	2200
1.0	0.08	1000
1.5	0.09	320
2.0	0.10	220
2.5	0.11	170
3.0	0.11	140
3.5	0.12	120
4.0	0.12	100
4.5	0.13	90
5.0	0.13	80
5.5	0.14	70
6.0	0.15	65
6.5	0.15	60
7.0	0.15	55
7.5	0.16	50

Tabla 2.1: Ensayo de consumo y señal de codificador del motor sin carga

### 2.1.5. Estado Inicial

La plataforma de la que se disponía en un principio carecía de cierta cualidades. La más destacable era la inestabilidad de la red WiFi, dicha conexión sufría constantes caídas con lo que dificultaba enormemente su teleoperación. Su otro talón de Aquiles era su escasa documentación disponible, al ser un producto que su fabricante da por descontinuado y al no disponer de una comunidad que lo mantenga, dicha base robótica provocaba quebraderos de cabeza para usuarios noveles. Finalmente, las baterías al haber completado su vida útil, disponían de una carga efectiva muy inferior a la inicial. En cambio, tanto la estructura, ruedas, motores y sensores se encontraban en mejores condiciones

## 2.2. Raspberry Pi

Raspberry Pi es un ordenador del tamaño de una tarjeta de crédito desarrollado en Reino Unido por la **Raspberry Pi Foundation** con el objetivo de fomentar las bases de la computación en escuelas. El desarrollo de este diminuto ordenador empezó en el 2006 hasta alcanzar su madurez y comercialización en Febrero del 2012.



Figura 2.5: Raspberri Pi

### 2.2.1. Características Técnicas

Aunque existen varios modelos de Raspberry Pi, este proyecto se basa en el modelo B (segunda revisión). Este modelo se caracteriza por:

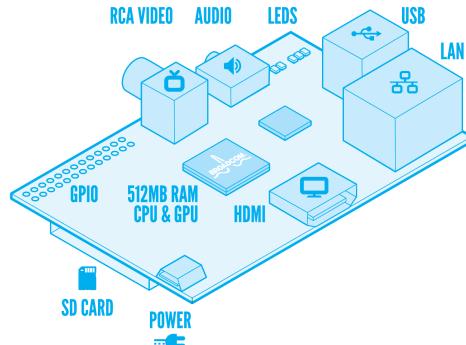


Figura 2.6: Esquema Modelo B

- Procesador Broadcom BCM2835 (SoC) compuesto por:
  - CPU: ARM1176JZF-S (Arquitectura ARM 11)
  - GPU: VIDEOCORE IV (250 Mhz)
  - Memória RAM de 512MB
- Connexiones multimedia:
  - 1x Entrada de vídeo: CSI
  - 3x Salidas de vídeo: RCA, HDMI y DSI

- 2x Salida de audio: 3.5mm jack y HDMI
  - 1x RS232
- Connexiones de datos:
- 1x Ranura tarjeta SD (Disco Duro)
  - 2x Puertos USB 2.0
  - 1x Puerto Ethernet RJ45 (10/100 MBit/s)
- Periféricos de bajo nivel:
- 8x GPIO
  - 1x UART
  - 1x I<sup>2</sup>C bus
  - 1x SPI bus (dos pines de selección)
- Otros datos:
- Alimentación: 5V via MicroUSB o GPIO
  - Medidas: 85.6mm x 56mm
  - Peso: 45 g

### 2.2.2. Sistemas Operativos

Debido a la gran revolución que han supuesto este tipo de ordenadores, se han creado a su alrededor más de 20 sistemas operativos, los más utilizados basados en Linux. Dentro de la comunidad que hay detrás de este pequeño ordenador, el sistema más utilizado con finalidades de disponer de un ordenador de bolsillo es Raspbian. Este sistema es una versión de Debian Wheezy para ARMv6 con soporte para coma flotante por hardware.

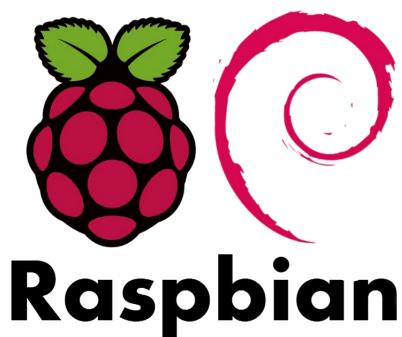


Figura 2.7: Icono de S.O. Raspbian

### 2.2.3. Programación

Debido al ímpetu de los desarrolladores de la Raspberry Pi por ofrecer una herramienta para las escuelas, esta placa posee un conjunto de librerías para ser programada en Python. Además cada vez hay más desarrolladores independiente que ofrecen módulos muy útiles, que nos ahorran tiempo y nos permiten crear más rápidamente nuestros propios proyectos.

### 2.2.4. GPIO

Uno de los grandes alicientes de esta placa comparada con otras ordenadores embebidos es el hecho de disponer de GPIOs, este elemento hace referencia al conjunto de pines que dispone la placa y pueden ser configurados como entradas o salidas digitales. Estos pines están conectados directamente a la CPU y sus niveles de operación se encuentran en los 0 o 3,3V. Por contra no pueden ofrecer más de 16mA por pin, y un máximo de unos 200mA en su totalidad.

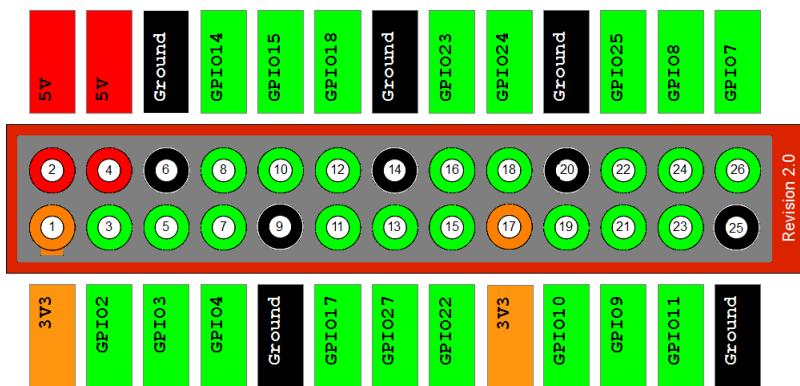


Figura 2.8: Nomenclatura del GPIO

Aunque todos los pins definidos como GPIO pueden usarse como entradas o salidas, algunos de estos disponen de otras funciones. En el caso del protocolo de comunicación I<sup>2</sup>C, el pin GPIO2 se convierte en **SDA** y el GPIO3 en **SCL**. Para el bus SPI, el GPIO10 pasa a ser **MOSI**, el GPIO9 a **MISO**, el GPIO11 a **SCLK** y los pins de selección de esclavo GPIO8 para el **CE0** y GPIO7 para el **CE1**. Finalmente la comunicación UART, **TXD** por el GPIO14 y **RXD** por el GPIO15.

### 2.3. ROS

El nombre de ROS viene de "Robot Operative System". ROS puede clasificarse como un Framework de Desarrollo en Robótica que ofrece herramientas y librerías para el desarrollo de sistemas robóticos. A pesar de su nombre, ROS no es un sistema operativo propiamente dicho (de hecho funciona sobre Linux).



Figura 2.9: Logotipo de ROS

El objetivo de ROS es ofrecer una solución integral al problema de desarrollo de robots. Para ello requiere de una abstracción del hardware, de tal manera que cada plataforma entienda la misma orden, es decir que si el ROS requiere que el robot se desplace hacia adelante, pues este lo cumpla independientemente de si se trata de un robot diferencial, bípedo o volador. Por su parte, ROS incorpora desde algoritmos de bajo nivel de control, cinemática, SLAM, etc. Hasta algoritmos de alto nivel como planificación o aprendizaje.

Lo que hace de ROS una herramienta tan extendida en el mundo de la robótica es su mecanismo de comunicaciones distribuido entre nodos. Un nodo es cualquier código de software del sistema (desde un algoritmo SLAM hasta un driver para el manejo de un motor). El hecho de usar nodos obliga a realizar un nivel de abstracción, de esta manera podemos asignar tareas para cada nodo y conseguir que muchos de estos se puedan reutilizar. Otra característica importante del uso de nodos es la independencia de donde se encuentre localizado el nodo, ya que ROS no hace distinciones.

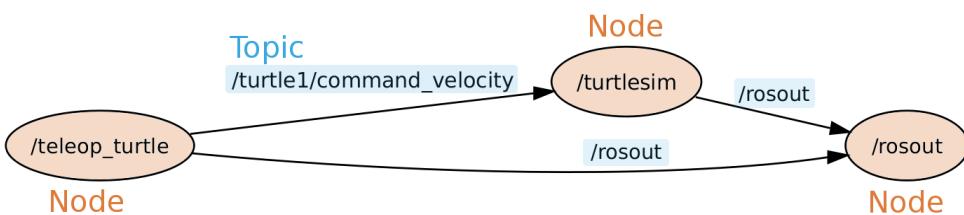


Figura 2.10: Gráfico ejemplo de la comunicación entre nodos.

Estos nodos se comunican entre ellos mediante mecanismos de intercambio de mensajes, entre estos ROS nos ofrece **Topics**, **Publishers**, **Subscribers** y **Services** entre otros. Para entender sus diferencias se propone un ejemplo: Imaginemos que tenemos un nodo que informa de la velocidad de cada rueda de un robot diferencial, debido a que la

velocidad puede variar en cualquier momento, se ha programado el nodo para que informe cada 0,1 segundos. En este caso el nodo publica la información, a este acto se define como crear un "Publisher". La información que envía se la llama "Topic" y su estructura "Messages". Todo esto se define en archivos de configuración o dentro del propio código del nodo.

Siguiendo con el ejemplo, si ahora hemos creado un nodo que se dedica a estudiar la odometría, haremos que se suscriba al tópico, es decir hemos creado un "Subscriber". Además de los tópicos, también existen los servicios, los servicio se diferencian en que un nodo llama a otro para intercambiar información. esta información no puede ser compartida entre otros nodos, excepto si estos utilizan el mismo servicio. Es decir, un tópico puede ser publicado por varios nodos, y varios nodos se pueden suscribir a él; Pero un "Service" solo funciona entre dos nodos.

Otro punto fuerte de ROS es la incorporación de simuladores virtuales donde podemos crear un modelo de nuestro robot y simular su comportamiento en cualquier ambiente, Gazebo y Rviz. Durante la simulación, el sistema es capaz de ofrecernos información sobre la dinámica de nuestro robot, obteniendo unos resultados muy parecidos a la realidad sin la necesidad que disponer del robot en la realidad.

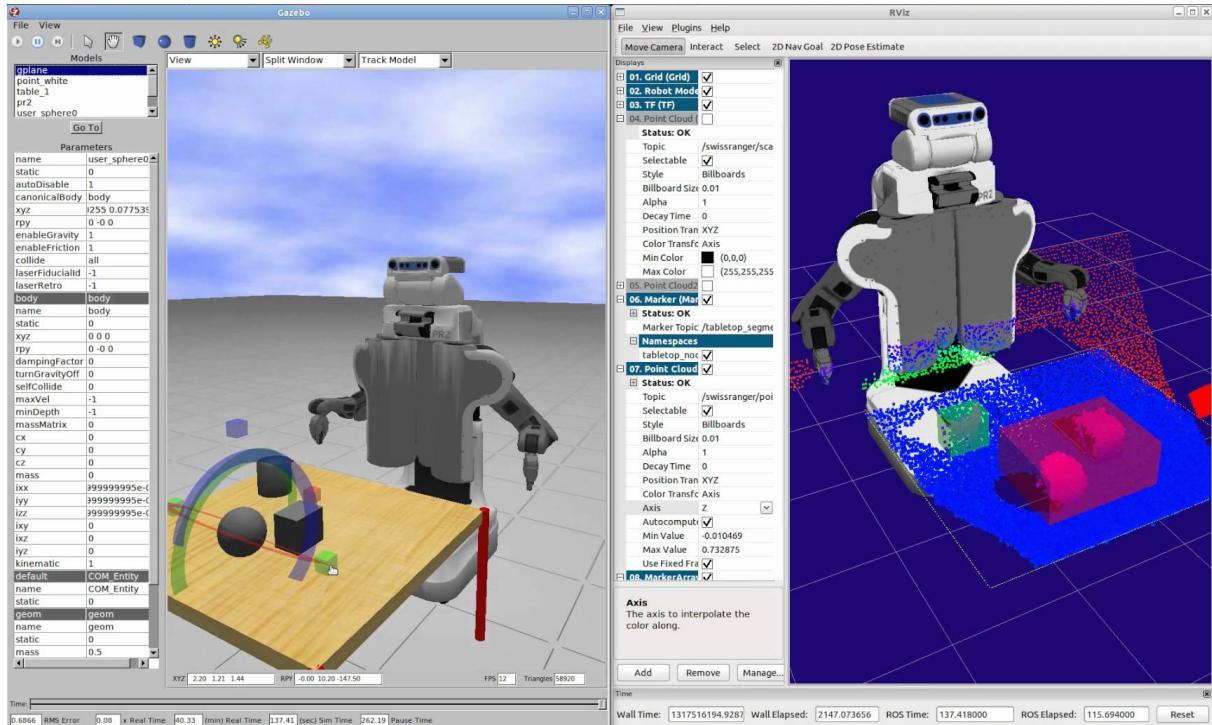


Figura 2.11: Simulación del robot PR2 en Gazebo y publicación en Rviz.

### 3. Especificaciones

Tal y como se ha definido anteriormente, el objetivo de este proyecto consiste en ofrecer una plataforma funcional para los futuros usuarios. Para definir esta "funcionalidad" se ha basado en el conocimiento aportado por antiguos usuarios, trabajadores en productos similares y en la experiencia propia.

Las especificaciones se han clasificado según su origen. Dependiendo de si son de carácter **software**, **electrónico** o **mecánico**. Las especificaciones de carácter informático engloban aspectos como la interficie con el usuario, las herramientas de desarrollo (librerías, documentación) y la disponibilidad a futuras modificaciones. Por contra, la electrónica ha de evitar que el usuario se encuentre obligado a manipular el interior del robot, pero que en el caso de dicha situación permita una solución simple del problema. Además, la electrónica ha de incorporar un sistema que permita conocer estados del robot de manera sencilla. Finalmente, la mecánica se encarga de incorporar las nuevas especificaciones en la plataforma, manteniendo el concepto inicial. Igual que en la electrónica, en el caso de una futura manipulación por parte del usuario, el diseño ha de facilitar el acceso a cualquier ubicación.

#### 3.1. Software

El principal requisito del software es ofrecer un acceso simple al usuario. Teniendo presente el hecho que pueda ser utilizado tanto para usuario noveles como para usuarios experimentados. Por ello se ha planteado las siguientes especificaciones.

##### 3.1.1. GitHub

Con el objetivo de ofrecer toda la información necesaria a futuros usuarios se ha creado el repositorio <https://github.com/Guasch5000/RosPiBot>, que permitirá acceder a todos los archivos relacionados con este proyecto, desde códigos de programación, hasta los diseños de los componentes a utilizar. De este modo se consigue que la información perdure, sea accesible y permite ponerse en contacto con su desarrollador.

##### 3.1.2. Librerías

Para aquel desarrollador que decida utilizar esta plataforma se le ofrecerá un conjunto de librerías que permitan trabajar con el robot de manera comoda. Estas librerías se compondrán de una sintaxis limpia y un código legible que proporcionará una fácil interpretación.

### 3.2. Electrónica

El objetivo de la electrónica es ser la parte más vital de la plataforma, pero pasando desapercibida por el usuario. Ofreciendo una mínima interacción pero suficiente para comprender el estado del robot. Esta interacción se hará mediante señales luminosas o acústicas, e indicarán los siguientes estados.

#### 3.2.1. Funcionamiento

En la plataforma de partida no se dispone de ningún medio que indique si se encuentra operativo, por ello la primera especificación de la parte electrónica será la incorporación de un señal luminosa que muestre si el robot está encendido o apagado. También existe la posibilidad de ofrecer información de otros estados.

#### 3.2.2. Alimentación

Las baterías ofrecerán una autonomía suficiente para moverse en un entorno exterior, se estipula un mínimo de 2 horas en movimiento como mínimo aceptable. Además de la duración, su recarga deberá de ser lo máximo de comodo para el usuario, evitando que tenga que manipular las celdas de energía. Se recomienda elementos de seguridad ante problemas como sobrecargas o cortocircuitos y de gestión de energía para aumentar la vida útil.

Otro punto a tener en cuenta es la disponibilidad en encontrar repuestos de la batería, por ello se usarán productos comerciales fáciles de adquirir. Finalmente, se monitorizará el nivel de carga a partir de unos elementos luminosos.

#### 3.2.3. Lógica

Para el control de la plataforma el proyecto se ha basa en uso de la Raspberry Pi. Dicho ordenador será el responsable de efectuar todo los cálculos y órdenes, Aún así, a causa de sus limitaciones se precisa de una lógica complementaria capaz de gestionar y contabilizar las lecturas analógicas de los sensores, el control de motores, la comunicación externa y la distribución de energía.

#### 3.2.4. Comunicación

Para poder interaccionar con el plataforma robótica se proponen 3 medios alternativos. Cada uno de estos canales permitirán el acceso al usuario. Un primer medio será por vía cableada, a partir de un cable de red, los otros dos serán por vía inalámbrica, por un lado se buscará un red Wi-Fi predefinida y por otro se creará una red Wi-Fi propia.

### 3.3. Mecánica

Se intentará en la medida de lo posible mantener la silueta característica del robot, asignando mayor prioridad a aquellas modificaciones motivadas por software o electrónica.

#### 3.3.1. Estructura

Se desea mantener la estructura básica de la plataforma, dejando su peculiar forma de dos bloques unidos por una barra roscada longitudinal. Se mantendrá el concepto de perfiles cuadrados como elemento estructural pero modificando la separación entre ellos. La ubicación de los distintos elementos se dispondrán según su relación con el resto de dispositivos, uniendo según se traten de control de potencia, lógica o comunicación.

#### 3.3.2. Tornilleria

A causa del deterioro de los elementos de tornilleria se procederá a cambiar el roscado de todos los tornillos por uno más estandarizado como el métrico. Este cambio también incluye las barras estructurales, los adaptadores de los ejes y los prisioneros

## 4. Modificaciones

Para la realización de este proyecto se deberá trabajar en varios sectores. Debido que el software es el sector que permite mejor adaptarse a los nuevos cambios y que la mecánica de la plataforma proporciona cierta flexibilidad, el componente electrónico será el que ofrezca las mayores limitaciones. Por ello se considera que los primeros esfuerzos deberán centrarse en este sector, aprovechando al máximo los recursos ya existentes.

Aunque la plataforma inicial seguía funcionando, los problemas derivados de su desgaste propiciaron la necesidad de modificar en gran medida su interior. En un principio se demostró que simplemente insertando una Raspberry Pi, conectada mediante puerto ethernet, se podía comandar el robot. También fué posible acceder a datos básicos del robot como el estado de las baterías o la lectura de los sensores de proximidad. Pero el hecho de la inestabilidad de su unidad de procesamiento hizo necesaria su substitución completa. A causa de esta decisión, se tuvo que buscar el modo de comunicarse con los demás componentes ya existentes: etapa de alimentación, gestor de baterías, control de motores y lectura de sensores.

### 4.1. Componentes Electrónicos

Para poder actuar con cada dispositivo, el robot hacía uso de las interficies de comunicación UART e I<sup>2</sup>C; Pero aún sabiendo el protocolo empleado, la comprensión de la información que circulaba fué prácticamente imposible. Por ello se determinó que para poder proseguir con el proyecto hacía falta el desarrollo de nuevos componentes que substituyesen a los actuales y que permitiesen a futuros usuarios comprender rápidamente que protocolos se utilizan y cómo interaccionar con ellos. A continuación se detallan los componentes utilizados.

#### 4.1.1. Motorización

Debido al buen estado de los motores y del hecho que un par incorpora codificadores de cuadratura, se ha optado por mantener estos motores. Con este gesto evitamos la búsqueda de su substituto y nos aseguramos que se mantenga la misma funcionalidad que al inicio. Al conservar este dispositivo se considera que el consumo (por lo que afecta a la parte de motorización) permanecerá semejante, por tanto la nueva etapa de control de motores dispondrá de características similares. Siendo el voltaje nominal del grupo motriz de unos 7.2 se requiere de una alimentación acorde a este voltaje; En el caso de utilizar baterías de voltaje superior, será necesario una etapa DC-DC para adecuar los niveles.

#### 4.1.2. Sensores de proximidad

Cualquier plataforma robótica dedicada a la exploración ha de disponer de sensores que les proporcionen información sobre el mundo que le rodea. Por ello se aprovecharán los sensores analógicos infrarrojos actuales del robot, Sharp GP2Y0A2YK, y se anadirán cuatro nuevos sensores por ultrasonidos KS103.



Figura 4.1: Sensor de distancia por ultrasonidos KS103

Las características de este sensor han sido el motivo de su elección, pueden encontrarse en su ficha técnica [5]: Su gran rango de medida (de 1cm a 550cm), una resolución de 1mm, comunicación vía I<sup>2</sup>C (con 20 direcciones posibles), capaz de operar de 3.0 a 5.5V y permite hasta unas 500 lecturas por segundo. Otro factor que se ha tenido en cuenta es el hecho que la luz de sol en determinadas circunstancias puede afectar a la lectura real de los sensores infrarrojos, en cambio los sensores de ultrasonidos no quedan afectados.

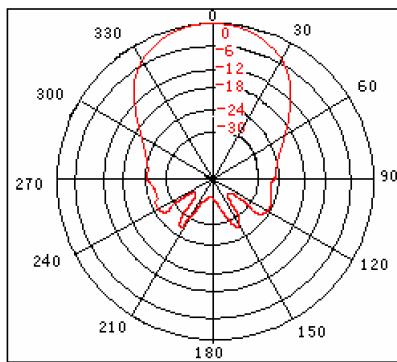


Figura 4.2: Diagrama de radiación del sensor KS103

El funcionamiento de este tipo de sensor de basa en el envío de un pulso y la recepción de su eco, luego el propio sensor determina el tiempo entre estos dos sucesos y calcula la distancia a la que se encuentra el obstáculo. Debido a que los ultrasonidos producidos pueden afectarse entre ellos si se realiza una lectura simultánea, se deberá esperar la finalización de la lectura de un sensor para activar el siguiente.

#### 4.1.3. Batería

Para la selección del sistema de acumulación de energía se ha barajado diferentes posibilidades que podemos encontrar en el mercado: Baterías de plomo, NiCd, NiMH, Ion Litio y Polímero de Litio (LiPo). Las de plomo han sido las primeras descartadas a causa de su baja densidad de carga y su elevado peso. Las constituidas por elementos de NiCd se encuentran en fase de desaparición por experimentar el efecto memoria (reducción de su capacidad al cabo del tiempo) y ser superadas por sus competidoras. Las baterías de Ion de Litio, aún disponiendo de una alta densidad de carga y bajo peso (motivo de su extendido uso en dispositivos móviles), no son capaces de ofrecer una gran capacidad de descarga.

Tipo	Densidad de carga [Wh/Kg]	Densidad de potencia [W/Kg]
NiCd	50	600
NiMh	70	700
Li-Ion	220	300
Li-Poly	150	2690

Tabla 4.1: Comparativa entre tipos de baterías

Llegados a este punto queda decidir entre las baterías constituidas por NiMH o LiPo. Ambas baterías están ampliamente comercializadas y ofrecen una alta capacidad de descarga si se comparan con las descartadas. Aunque las LiPo disponen de mejores características, según la tabla 4.1, se corre el riesgo de que lleguen a incendiarse si se sobrecargan o que queden inutilizadas si se alcanza un voltaje inferior de 2.5V por celda.



Figura 4.3: Batería LiPo con PCM instalado

Por ello se ha optado por el uso de unas baterías con un módulo PCM instalado, este pequeño dispositivo ofrece un control sobre la batería, protegiéndola de sobredescargas y gestionando su carga. Además ya no hace necesario cargadores específicos, simplemente una fuente de tensión acorde. Debido a que los motores tienen un voltaje nominal de 7.2V, se ha decidido el uso de baterías de 7.4V, estas baterías tienen un rango de funcionamiento

de 2.75 hasta los 4.2V.

Finalmente las la batería elegidas ha sido un conjunto formado por dos celdas de 3.7V de 6000mAh cada una conectadas en serie y con un modulo PCM instalado. Su consumo de carga llega a los 2A y a unos 6A en descarga.

#### 4.1.4. Punto de acceso WiFi

Tal y como se ha descrito en las especificaciones, se requiere de una sistema que ofrezca un acceso WiFi. Debido a que se ha descartado el uso del módulo que ya disponía el robot, se hace necesario un nuevo dispositivo que se encargue. En este caso de ha optado por el uso de un elemento comercial de tamaño reducido que se pueda alimentar fácilmente, ofrezca un alcance aceptable y de rápida configuración.



Figura 4.4: Router TP-Link WR702N

El dispositivo elegido se trata del mini router TL-WR702N de la empresa TP-LINK. Este modelo ofrece una velocidad máxima de transmisión de 150Mbps, se alimenta mediante un puerto MicroUSB (5V) y dispone de un puerto ethernet RJ45, que dependiendo del modo de trabajo se comporta como un puerto WAN o LAN.

Según su guía de usuario [6], entre todas sus configuraciones (AP Mode, Router Mode, Client Mode, Repeater Mode y Bridge Mode) se ha optado por la de punto de acceso con el nombre de RosPiBot y servidor DHCP activado. De esta manera el usuario sólo deberá conectarse a su red para poder acceder al robot.

#### 4.1.5. Comutador de red

Según las especificaciones se desea que la nueva plataforma disponga de tres medios de acceso a red. Dos de estos medios (Wifi y clavija ethernet) han de estar conectados directamente.

mente con el puerto ethernet de la Raspberry, por este motivo se necesita un commutador o switch. Puesto que el robot ya dispone de un switch de 5 puertos (4 habilitados), se aprovechará el mismo dispositivo.

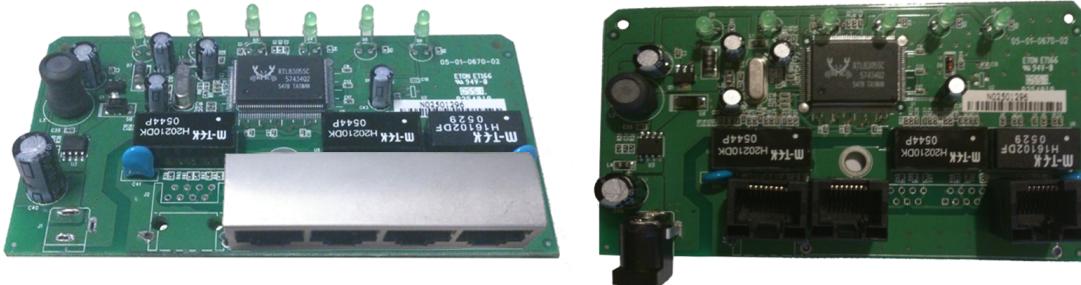


Figura 4.5: Switch antes y después de la modificación

Debido a que el conector que dispone el switch implica una que el conector se encuentre en posición horizontal, se ha optado por la substitución de la clavija de cuatro puertos por tres verticales, de esta manera se reduce la superficie que requiere el switch con los conectores.

#### 4.1.6. Descodificador de cuadratura

Uno de los elementos más útiles que podemos encontrar en la plataforma es la existencia de dos codificadores de cuadratura dispuestos en el grupo motriz delantero (uno a cada lado). Estos dispositivos permiten medir los cambios del eje del motor (antes de la etapa reductora) con gran precisión. Para su funcionamiento requieren de una alimentación de 5 voltios, siendo este mismo voltaje el usado en la salida del señal. Debido a su voltaje, superior a los 3.3 voltios que soportan los pines de la Raspberry Pi, se hace necesario un adaptador de nivel. Además, según los resultados experimentados, la frecuencia máxima de interrupciones generadas por el codificador llega hasta las 80.000 por segundo, Haciendo imposible que el microprocesador sea capaz de gestionarlo.

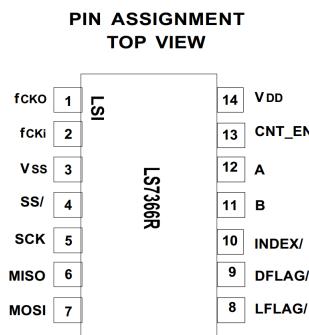


Figura 4.6: Diagrama de pines LS7366

Para solventar el problema, se hace uso del circuito integrado LS7366 [7], un contador de cuadratura de hasta 32 bits con interficie SPI. Este componente incrementa o decremente un contador interno dependiendo de la secuencia que proviene del codificador de cuadratura. Además en el momento que se quiera acceder al registro de dicho contador, el circuito procede a copiar el estado de contador en un segundo registro, permitiendo que la lectura no interfiera con el cómputo.

Gracias a este integrado, se evita que el microprocesador tenga que hacerse cargo del gran número de interrupciones, permitiéndole dedicarse a otros procesos. Para la comunicación se aprovecha el módulo propio SPI del BCM2835 y dos de los pines de selección de escalvo.

#### 4.1.7. Adaptador de nivel de tensión

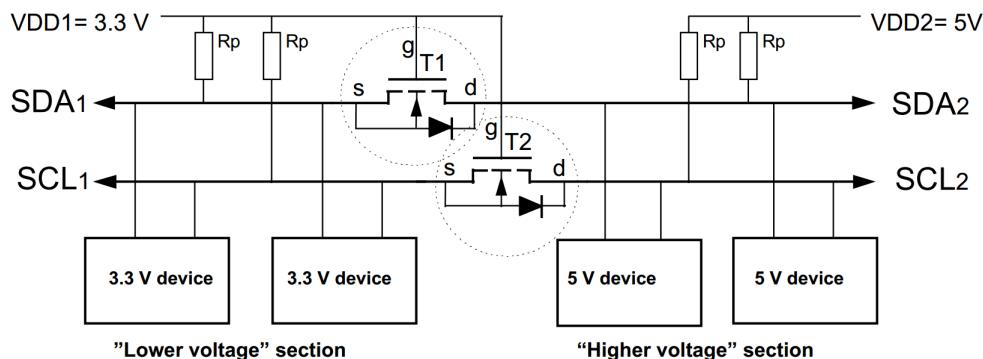


Figura 4.7: Circuito adaptador de tensión

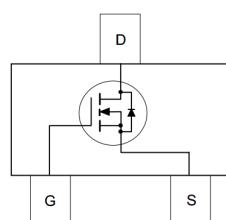


Figura 4.8: Diagrama BSS138

Para una correcta comunicación entre los diferentes dispositivos es preciso adaptar los diferentes voltajes. Para ello hay que tener en cuenta en qué rangos cada dispositivo se comunica; Mientras que la Raspberry Pi usa voltajes en su GPIO de 0 a 3.3V, el controlador de motores opera con tensiones superior a los 4.5V.

Debido a que algunos pines requieren bidireccionalidad (por ejemplo I<sup>2</sup>C) se ha descartado por el uso del circuito recomendado por la empresa **Philips Semiconductors**

explicado en su documento **AN97055** [8] y que corresponde a la figura 4.7. Finalmente para el circuito se utilizarán resistencias de  $10K\Omega$  para la función de "pull-up" y el mosfet de canal N **BSS138** [9] como el de la figura 4.8.

#### 4.1.8. Lectura de valores analógicos

Debido al uso de sensores de salida analógica (sensores de distancia por infrarrojos), el uso de resistencias Shunt y la monitorización de baterías, se requiere una lectura analógica. Por ello se utilizará un conversor analógico a digital o ADC ya que la Raspberry no dispone de uno propio. De los posibles componentes que cumplen esta función se ha elegido aquel que comunica por I<sup>2</sup>C, ofrezca una resolución aceptable, haya sido utilizado satisfactoriamente por otros usuarios y de menor coste. El componente elegido ha sido el MCP3428 [10], un conversor ADC Sigma-Delta de 4 canales diferenciales, resolución de hasta 16 bits y voltaje de referencia interno de 2.048 voltios.

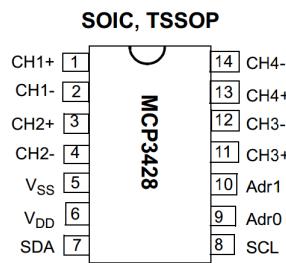


Figura 4.9: Diagrama de pines MCP3428

#### 4.1.9. Monitorización de la Batería

Conocer el estado de la batería en cualquier momento es crucial en plataformas móviles, por ello su monitorizaje es imprescindible. Dicho monitorizaje se efectúa de dos maneras distintas: La primera mediante un módulo ADC que digitaliza el estado de la batería, y la segunda utiliza el circuito integrado LM3914 [12], un dispositivo encargado de medir el voltaje y mostrar dicha información mediante LEDs.

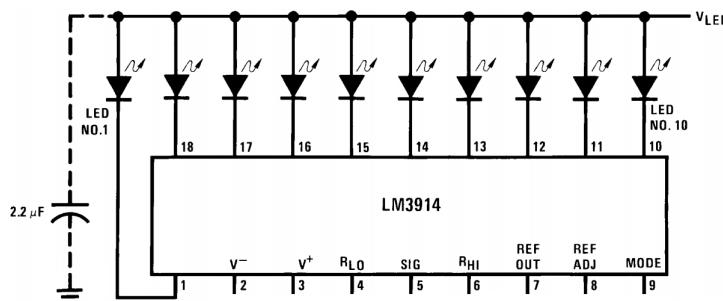


Figura 4.10: Diagrama de pines LM3914

#### 4.1.10. Alimentación de lógica

A causa de los diferentes niveles de tensión entre la batería que cada componente requiere, se hace necesario un sistema que lo acomode; Por ello se ha decidido usar un regulador que ofrezca una salida estable de 5 voltios. En el caso de la Raspberry Pi se utilizará el propio regulador interno de 5 a 3.3V. A partir de experimentaciones previas se estima un consumo medio de 1500mA. Para satisfacer dicho consumo es necesario un regulador de alta eficiencia, y debido a que siempre nos encontramos la situación de reducir el voltaje, deberemos usar un convertidor Buck.

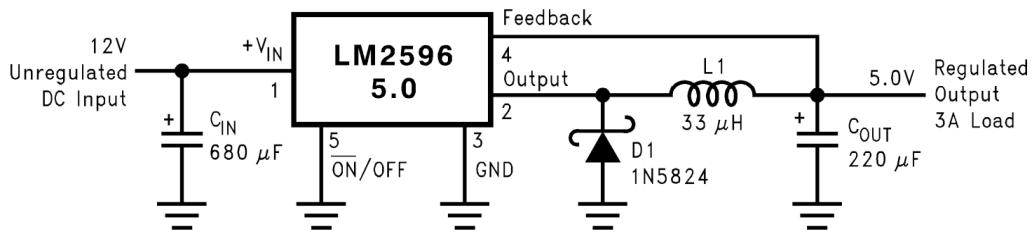


Figura 4.11: Circuito regulador Buck 5V

El componente elegido ha sido el LM2596 [?], un regulador capaz de ofrecer hasta 3A. Su elección se ha basado en el tipo de componentes que requiere, su facilidad de uso, su coste total reducido y la experiencia previa con este regulador. Para la selección de componentes, la mayoría vienen definidos por el propio fabricante, excepto la bobina, la cual para obtener el máximo rendimiento del regulador su inductancia se ha de elegir según la figura 4.12. Puesto que las baterías oscilan de 6 a 8.4V y que el consumo varía de 1 a 2A, se ha coloreado de verde claro aquellas zonas en la que el regulador puede llegar a trabajar y de color más oscuro la zona más típica dónde se puede encontrar. Por esto motivo se ha elegido el valor de 22  $\mu$ H.

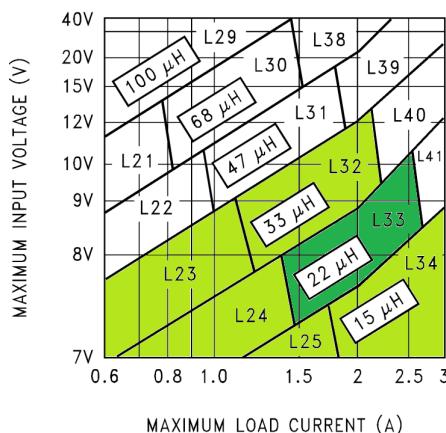


Figura 4.12: Guia de selección de inductancia

#### 4.1.11. Control de motores

Para el control de motores se utilizará el mismo componente L298 usado en el robot original, un doble puente en H. Este dispositivo permite un control independiente de dos motores de continua hasta un consumo de 2 amperios por motor.

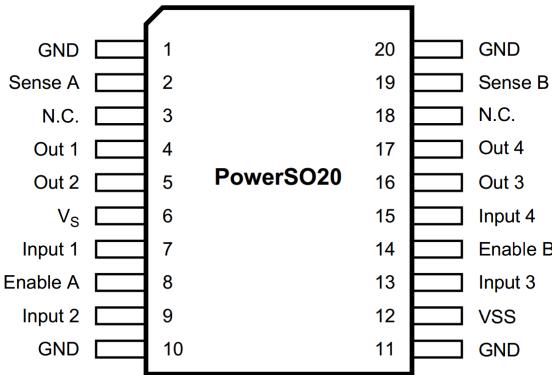


Figura 4.13: Diagrama de pines L298N

El sentido de rotación se dirige a partir de dos pines indicados como "Inputs" y la velocidad mediante una señal PWM en el pin de "Enable". Aunque dispone de una salida para conocer la intensidad que circula por cada canal (se conecta en serie una resistencia Shunt y se mide la diferencia de potencial), no dispone de ningún sensor para saber la velocidad del motor.

### 4.2. Integración de los componentes

Los componentes descritos anteriormente desde los puntos 4.1.6 hasta 4.1.11 requieren de ser montados en una PCB. Para ello se han diseñado un conjunto de placas, se han enviado a producir y finalmente se han ensamblado los componentes manualmente.

Para la producción de las PCB se ha utilizado un servicio de prototipado de bajo coste pero con la limitación que los diseños han de estar contenidos en rectángulos de 5x5, 5x10 o 10x10cm. Aunque posiblemente se pudiera incluir todos los componentes en un solo circuito impreso de 10x5cm, se ha preferido crear 3 diseños de 5x5cm. Así se reduce el riesgo de deshacerse de toda una placa por culpa de un imperfección en un componente y se da modularidad al sistema. Las placas creadas son: SecurePi, SensorPi y MotorPi.

#### 4.2.1. SecurePi

Esta placa tiene dos objetivos: Ofrecer una etapa de regulación para la alimentación de todos aquellos elementos que requieran una alimentación de 5V y proteger todos los

pines GPIO de la Raspberry Pi (ofreciendo bidireccionalidad y convirtiendo el voltaje de 3.3 a 5V).

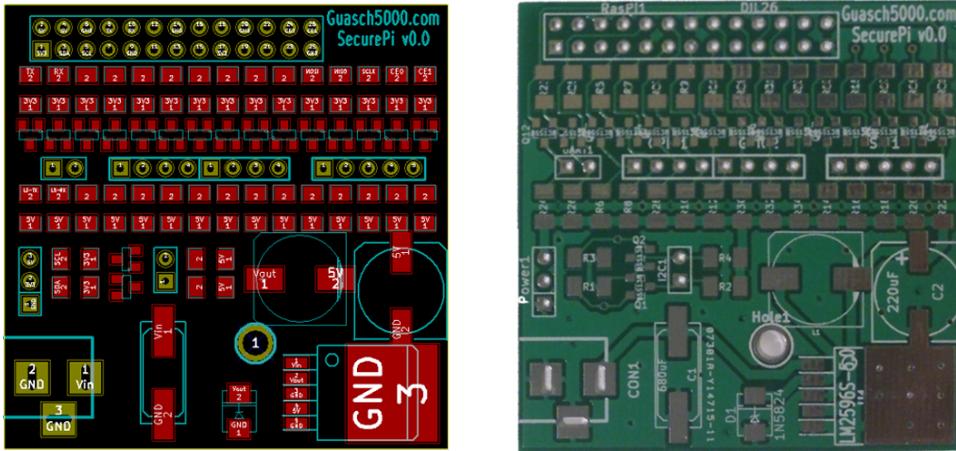


Figura 4.14: Layout y PCB de la placa SecurePi

Otra característica de esta placa es la redistribución de los pines, separados según la funcionalidad de cada uno. Podemos observar como el conector de 26 pines se ha distribuido en varios: UART(x2), GPIO(x8), SPI(x5), I2C(x2) y alimentación (GND, 3.3V y 5V).

Tal y como se aprecia en la figura 4.15, una vez montada la SecurePi encima de la Raspberry, el usuario puede seguir accediendo a los pines directos del GPIO, usar las nuevas salidas elevadas a 5V y decidir si alimentar el dispositivo mediante el puerto MicroUSB o con una fuente de mayor tensión mediante el conector Jack de 5.1mm.



Figura 4.15: SecurePi colocada con la Raspberry Pi

#### 4.2.2. SensorPi

Sobre la placa SecurePi se coloca la SensorPi, esta placa dispone de los descodificadores de cuadratura, los conversores analógicos y un conjunto de leds (uno verde para indicar la correcta alimentación y ocho ámbar para los pines GPIO destinador a al control de motores).

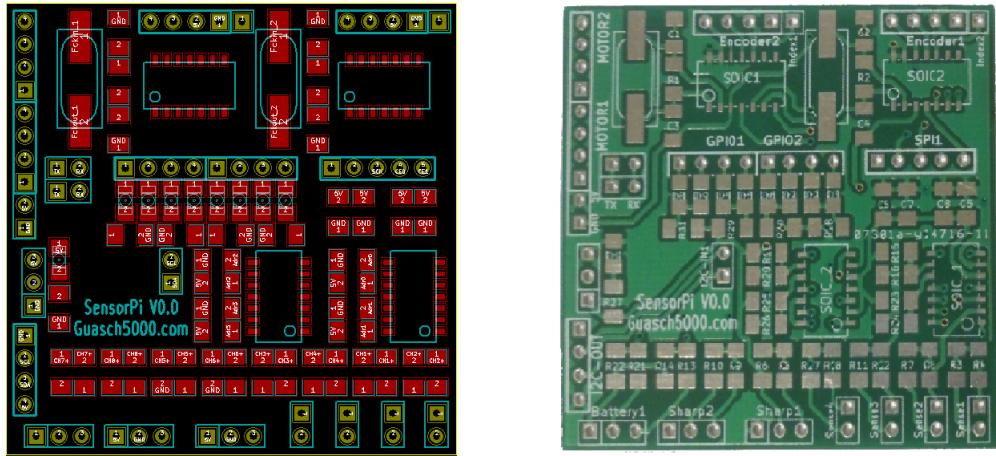


Figura 4.16: Layout y PCB de la placa SensorPi

Para cada uno de los descodificadores se han dispuesto de dos tiras de tres pines que disponen de alimentación (GND y 5V), dos canales de cuadratura y un pin opcional para el índice. En el caso de las entradas analógicas, se han adaptado mediante divisores de tensiones para ajustar al límite de 2.048V. Para medir las baterías se han dispuesto de dos lectura para cada una de las celdas en serie, la primera lectura tiene el fondo de escala a 5V y la segunda a 10V. Para los sensores infrarrojos se han configurado a 3.5V como máximo y a 2.2V las diseñadas para la lectura de la intensidad (midiendo la caída de potencial entre los extremos de una resistencia Shunt).

Finalmente las ocho salidas del GPIO se han colocado en un extremo para facilitar su acceso y se ha dispuesto una tira de 4 pines extra para la alimentación y comunicación de los dispositivos que utilizan I<sup>2</sup>C.

#### 4.2.3. MotorPi

Esta placa controla la gestión de motores e incluye el circuito de monitorización de la batería. Al contrario que las dos placas anteriormente descritas, la MotorPi no está diseñada para ser encajada encima de otra placa.

Se han utilizado bloques de terminales para el cableado de potencia (batería y motores) y tiras de pines para el resto de conexiones separadas en 3 bloques: El primero consiste en

una tira de 11 pines pensado para el visualizador led de 10 barras. Un segundo bloque de 10 terminales incorpora los cuatro pines de indicación del sentido de giro de los motores, dos para la habilitación de los motores delanteros y cuatro más de lectura de corriente. Finalmente el tercer bloque proporciona la alimentación de lógica para los L298, dos pines de habilitación para los motores traseros y sus 4 correspondientes para la lectura de consumo.

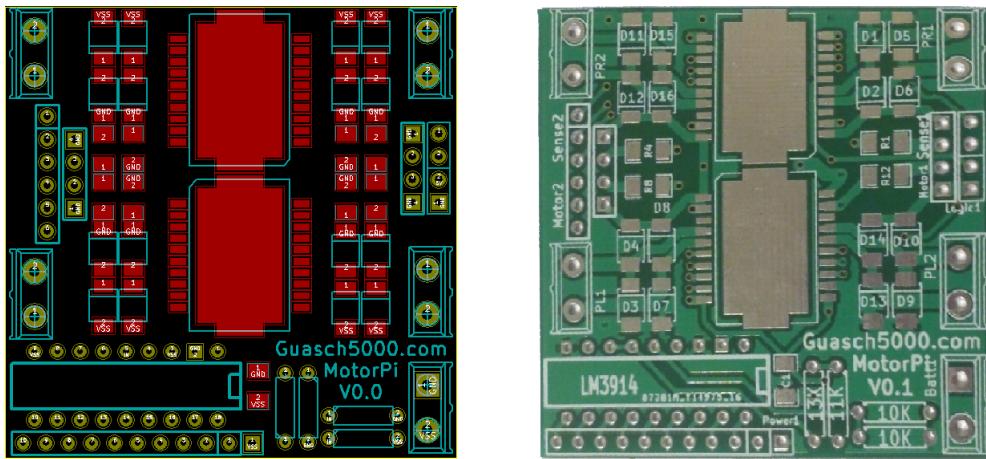


Figura 4.17: Layout y PCB de la placa MotorPi

## 4.3. Conexiónado

Una vez definidos todos los elementos que participan en la modificación de la plataforma robótica, es necesario de indicar cómo se conectan entre ellos. Debido a que cada componente debe ser alimentado a un voltaje específico, se han proyectado diferentes conexiónados según la función de sus elementos:

### 4.3.1. Potencia

Este cableado se encarga de la distribución de energía a todos aquellos componentes que estén diseñados para aceptar voltajes dentro del rango de la batería. Si nos fijamos en la figura 4.18 estos componentes forman dos grupos que se han enmarcado: El grupo encargado de la recarga, formado por el conector "Power Jack" y un fusible para proteger la fuente de alimentación exterior, y el grupo encargado del consumo, donde encontramos un interruptor, el switch y los dos circuitos impresos MotorPi y SecurePi.

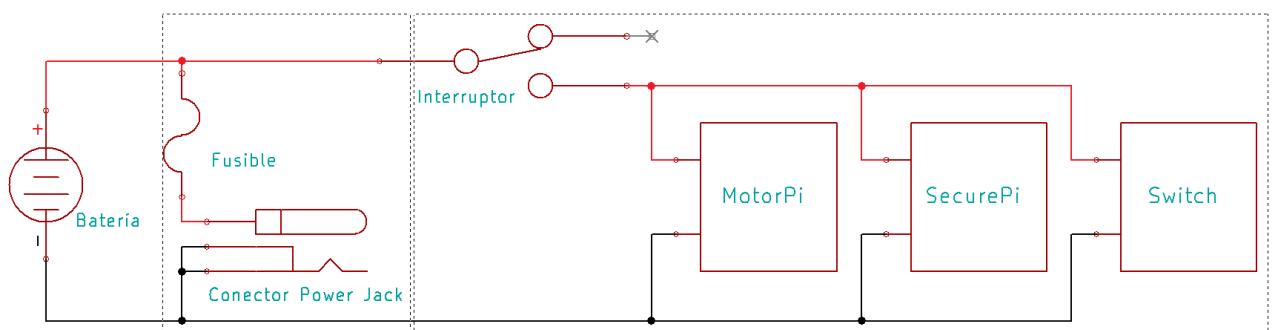


Figura 4.18: Esquema eléctrico del conexionado de potencia

Aunque el circuito de recarga dispone de fusible, se puede observar que el segundo no. Este hecho se debe a que en la situación de funcionamiento normal, el robot se encuentra en movimiento, el circuito de recarga está abierto, por lo que toda la energía la subministra la batería. Y en el caso que una sobrecarga o cortocircuito el propio módulo PCM actuaría, con lo que no hace necesaria la instalación de un fusible para el circuito de consumo. En cambio, si por casualidad nos encontramos en la situación que tenemos el robot recargándose con poca batería y decidimos activar los motores, debido al pico de consumo que aparece se puede dar el caso que sea la fuente quién alimente el grupo motor y la acabemos dañando.

### 4.3.2. Alimentación de lógica

Una vez la placa SecurePi ha sido alimentada, ofrece una tensión de 5V estable. Los consumidores de esta salida son la propia Raspberry Pi, la placa SensorPi y el Router

WiFi.

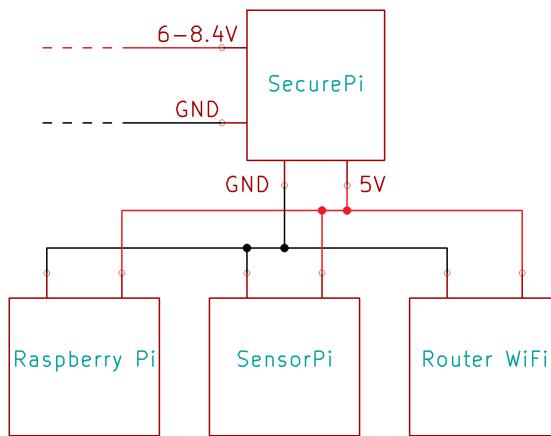


Figura 4.19: Esquema eléctrico de la alimentación de lógica

Tanto la Raspberry Pi como la SensorPi no requieren de cableado puesto que su contacto se efectúa mediante los pines previamente soldados. Por el otro lado, se ha modificado un cable USB tipo A a MicroUSB para ser conectado al GPIO y subministrar energía al Router.

#### 4.3.3. Cableado SensorPi

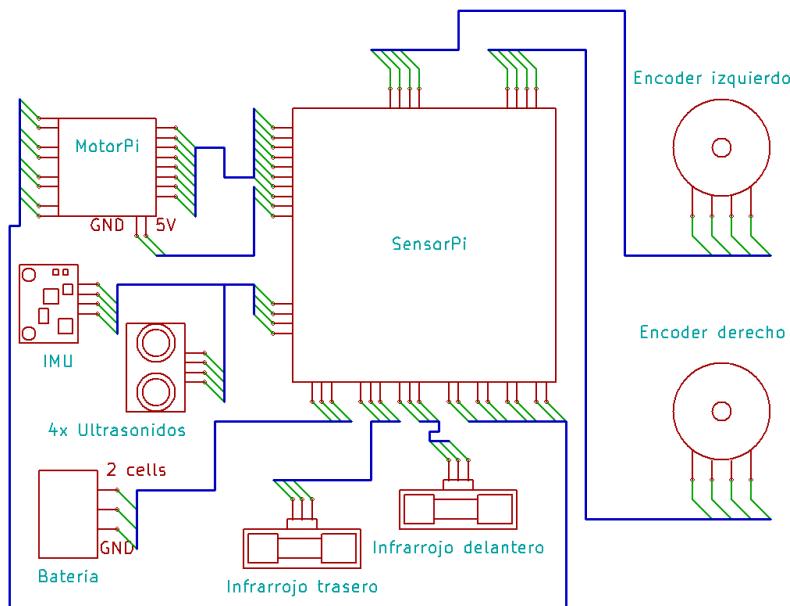


Figura 4.20: Esquema eléctrico del cableado de la placa SensorPi

Del mismo modo que la placa SecurePi alimenta a un conjunto de elementos, la placa SensorPi hace lo mismo. Podemos observar la distribución del cableado en la figura 4.20 y los elementos que dependen de esta placa para funcionar.

El conexiónado con la placa MotorPi se separa en 3 grupos: Alimentación, comunicación con los controladores y lectura de consumo. El resto de componentes también son alimentados excepto los destinados a la lectura del estado de las celdas de la batería.

#### 4.3.4. Comunicación

Llegados a este punto tenemos todo lo necesario para que el robot sea capaz de moverse y obtener información de su entorno, solamente nos queda poder comunicarnos con él. Para ello se ha conectado la Raspberry Pi, Router, Switch y la clavija de red externa mediante cables Ethernet. Por otra parte se han extendido las salidas USB de la Raspberry Pi hacia el exterior del robot para facilitar su acceso.

## 4.4. Distribución interna

Una vez que los elementos internos del robot han sido definidos, es hora de organizar su distribución. Para ello se han considerados las especificaciones previas, donde se define a grandes rasgos su ubicación preferida. Durante esta fase se ha ido corrigiendo la disposición de los elementos hasta encontrar uno que satisface todas las restricciones. Para poder llevar a cabo esta tarea se han modelado todos los componentes en SolidWorks®, de tal manera que se puede obtener una representación gráfica de su colocación y además poder usar herramientas propias del programa para detectar colisiones o interferencias entre sólidos.

### 4.4.1. Diseño preliminar

Puesto que se desea que la parte de procesamiento y comunicación se encuentran en el mismo hemisferio, la colocación de la clavija de Ethernet externa y los dos conectores USB también deberán estar en el mismo bloque. Siguiendo con el proceso de descarte, el único sensor que se puede colocar en el panel trasero derecho es el infrarrojo, ya que el ultrasonidos ocuparía demasiado y haciendo imposible una distribución acorde de los conectores externos. A causa de esta elección quedan fijados en que lugares se distribuyen el resto de sensores.

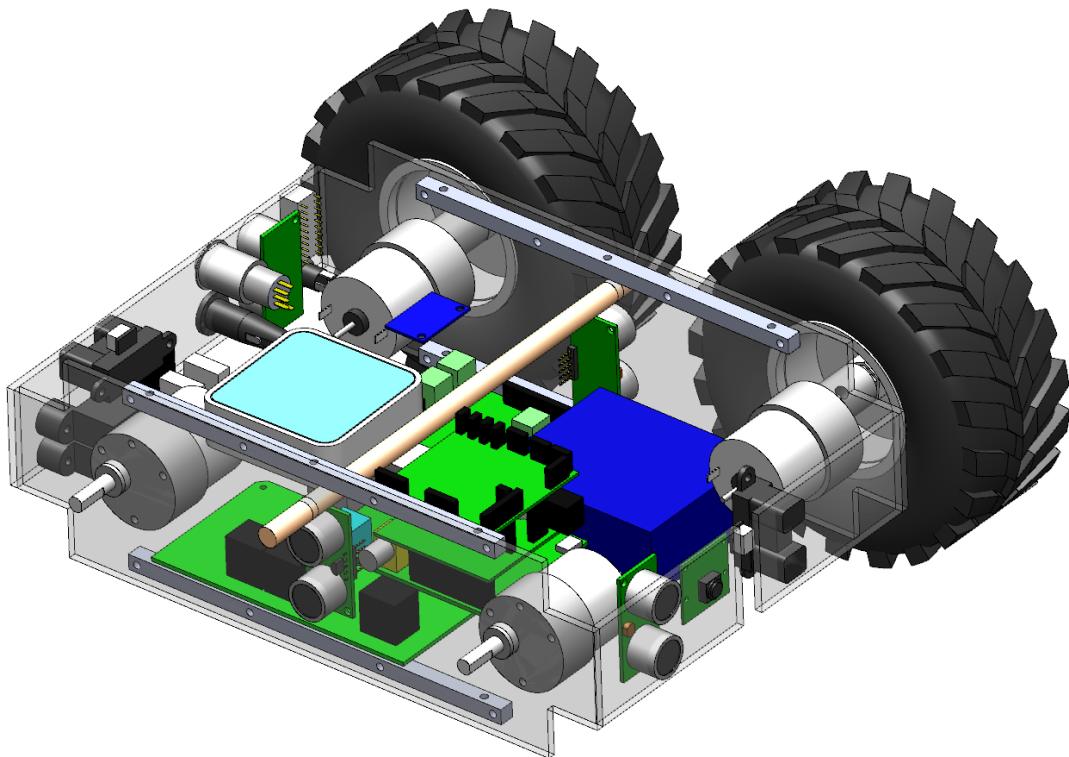


Figura 4.21: Diseño preliminar con la ubicación de todos los componentes

Con los conectores y sensores colocados, es el turno de el Switch y la Raspberry Pi.

En este caso habían muy pocas combinaciones posibles, haciendo que la preferida fuese aquella que permitiese una fácil manipulación del cableado más frágil (los procedentes de la placa SensorPi) y que en caso de problema procedente del sistema operativo, la extracción de la SD fuese cómoda. Finalmente se introdujo el Router, la cámara se ubicó en el punto más centrado y se definió la rotación de los motores (los ejes no son colineales al eje central).

A continuación se siguió con el hemisferio izquierdo. En el panel trasero se ubicaron el interruptor vandálico, el portafusibles, el encapsulado de 10 leds, el conector para la carga y el sensor de ultrasonidos. Puesto que el uno de los motores se encuentra cerca, la distribución se ha basado en la profundidad a la que llegan estos componentes, de tal manera que se ha evitado cualquier tipo de colisión y se ha dejado espacio para el cableado correspondiente. Para terminar se colocaron el placa MotorPi y la batería en la parte inferior de robot y se dejó la IMU en la parte superior para acceder fácilmente durante el ensamblado, el resultado final se puede apreciar en la figura 4.21

#### 4.4.2. Soportes

Con todos los elementos ubicados, llegamos al punto de ofrecer un soporte a cada uno. El objetivo de estos soportes es que sus elementos se unan a los paneles más cercanos y posteriormente unir estos grupos al ensamblaje principal.

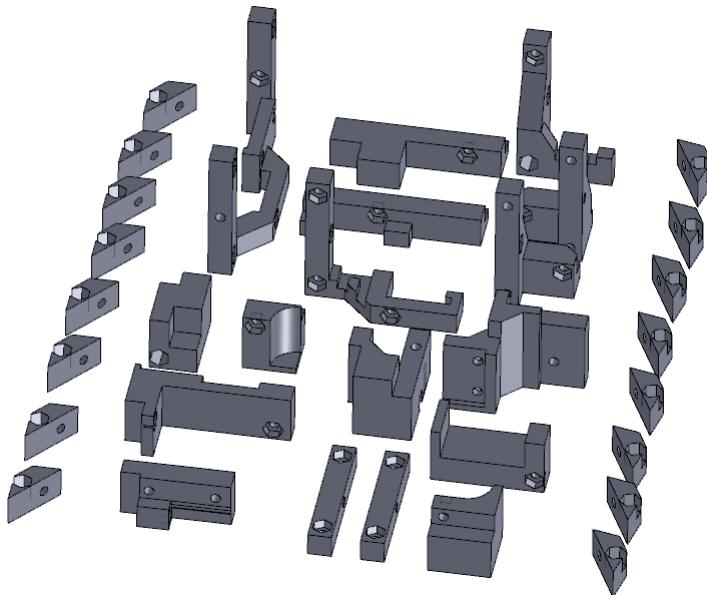


Figura 4.22: Conjunto de soportes orientados para impresión 3D

Debido a las intrincadas formas que cada uno de los soportes tenía que disponer, se optó por la impresión 3D basada en la superposición de material para la fabricación de estos. Este hecho afectó al diseño final de los soportes, haciendo que estos se pudiesen

fabricar. Un aspecto que se tuvo en cuenta fue la disposición de ranuras para las tuercas, haciendo que estas se colocarán fácilmente y que no requieran del uso de alicates o llaves para su posterior sujeción a la hora de insertar el tornillo.

Para darle un acabado menos rectangular se diseñaron unos chaflanes, para la colocación de estos se crearon piezas auxiliares con el único objetivo de ser unidas al cuerpo principal. Estas piezas auxiliares corresponden a las que se encuentran en los extremos de la figura 4.22 formando dos filas.

#### 4.4.3. Diseño Final

En esta última fase se crearon todos los paneles y sus correspondientes orificios para tornillería, ejes de motores y paso de cableado. Para mantener la estética original, todos los paneles que forman el robot se ha decidido que sean negros excepto los dos superiores.



Figura 4.23: Acabado final delantero del robot

Para los paneles superiores se ha optado por que sean transparentes, con este cambio se pretende que el usuario tenga una idea de que contiene el robot a primera vista, además de permitir encontrar rápidamente incidencias por de la falta de indicadores encendidos.



Figura 4.24: Acabado final trasero del robot

## 4.5. Componentes mecánicos

El trabajo de ofrecer una estructura donde poder incorporar toda la electrónica y que permita al robot desplazarse por su entorno recae en la mecánica. De entre los varios diseños posibles se ha elegido aquel que conserve el espíritu del robot original, es decir que siga siendo una plataforma constituida por dos partes, que mantenga el concepto de estructura una externa y que su tamaño se respete. A continuación se detallan los diferentes trabajos realizados en este proyecto.

### 4.5.1. Mecanizados

Como se ha comentado anteriormente, todos los roscados habían sido elaborados con una métrica no estándar. En un primer lugar tenemos los roscados correspondiente a las ocho barras longitudinales de aluminio, unos siete orificios por barra (cuatro en una dirección y el resto en su perpendicular). Por suerte el diámetro exterior del roscado era casi idéntico al diámetro interior de una rosca típico de M3. Gracias a este hecho no se requería la necesidad de repasar los orificios con una broca y volver a roscar. Simplemente se volvieron a mecanizar utilizando un macho de roscar.

Otro elemento que también requería de una modificación era los acopladores (couplings) entre la salida del eje del motor y de las ruedas. Estos acopladores, que aparecen en la

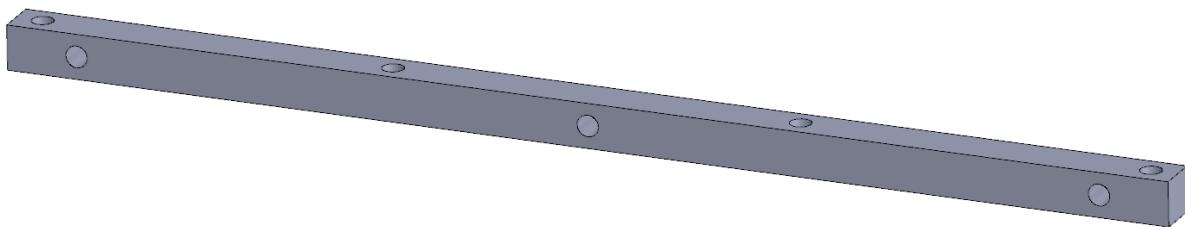


Figura 4.25: Modelo por ordenador de una barra longitudinal de aluminio.

figura 4.26, disponen en un extremo un orificio de 6mm de diámetro pensado para insertar el eje del motor, en el otro extremo se ha mecanizado el cilindro hasta obtener un prisma de base hexagonal (para ofrecer un giro soldiario con la rueda) y en su interior se ha mecanizado una rosca cercana al M4. Perpendicular al eje principal el acoplador dispone de un tercer orificio diseñado para utilizar un espárrago prisionero. Debido al degradado estado de los tornillos y prisioneros, se tuvo que utilizar métodos poco convencionales para poder extraerlos. En los tornillos que unían la rueda se les practicó una ranura para poder insertar un destornillador plano, ya que el óxido y su cabeza con hueco hexagonal no estandarizado imposibilitaba el uso de llaves Allen. En el caso de los prisioneros, no hubo más remedio que realizar un corte con sierra de disco para obtener una ranura, pero para ello se tuvo que eliminar parte del aluminio del acoplador.

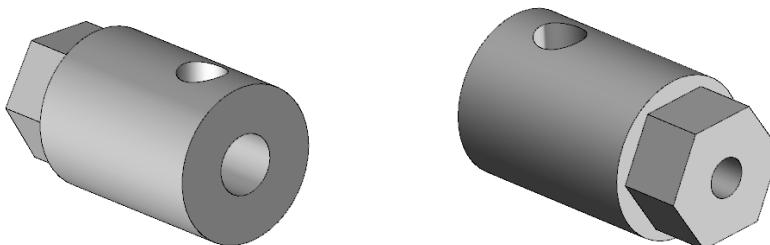


Figura 4.26: Modelo por ordenador de un acoplador de aluminio.

#### 4.5.2. Paneles

Una vez diseñados los paneles que conformarían la estructura del robot, se procedió a su fabricación. Debido a que en un principio se desconocía que material se utilizaría (PVC, Metacrilato, Policarbonato, Acrilico, etc.) se decidió usar un grosor típico, fácil de adquirir y que aportase una resistencia aceptable.

Finalmente el material elegido fue el metacrilato, el motivo que hizo decantarse la balanza fue saber que era el único que se podía cortar a un precio razonable. Mientras

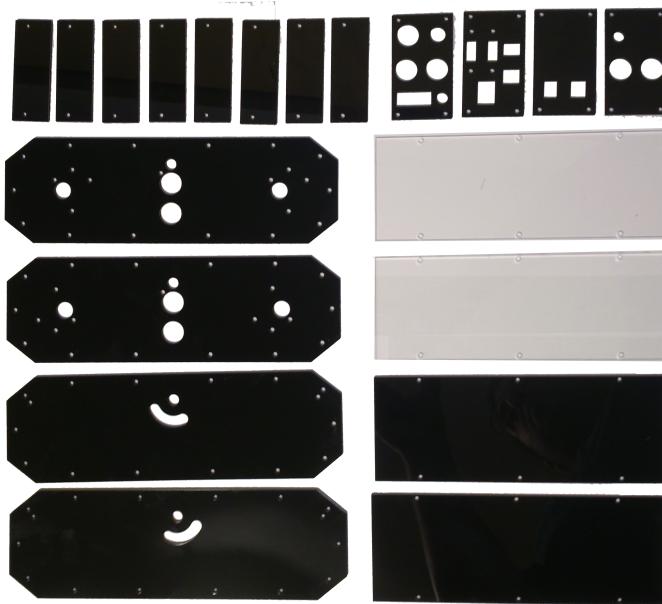


Figura 4.27: Conjunto de paneles obtenidos por corte láser.

que el resto requerían del uso de una fresadora CNC, el metacrilato podía ser trabajado mediante corte láser.

#### 4.5.3. Tornillería

Además de soportar la estructura el objetivo de la nueva tornillería que se ha utilizado es que resista mejor las inclemencias de su entorno, por este motivo se ha querido que todos los tornillos sean inoxidables. Además para evitar posibles enredos a la hora de realizar algún mantenimiento se han restringido los tipos de tornillos utilizados. De tal manera que para la sujeción de los paneles, elementos internos y motores se ha utilizado un tornillo de calidad A2, de cabeza alomada o DIN 7985 y de M3x10 o M3x20 según requiera la ocasión. Para sujetar la cámara se han usado el mismo DIN7985 pero de M2x10. Para la unión entre el acoplador y la llanta unos DIN912 A2 de M4x10 y para sujetar el acoplador al eje unos espárragos DIN 916 de M5x5, estos últimos han tenido que ser de Acero Negro ya que no se han localizado de inoxidables.

## 4.6. Montaje

En esta parte se realiza el ensamblado final del robot. El objetivo es crear un sándwich donde los paneles laterales son el continente y los paneles traseros, delanteros, superiores, inferiores y chaflanes el contenido.

### 4.6.1. Grupos individuales

Para ello se unen los paneles interiores verticales con sus tapas inferiores, de esta manera se crea una escuadra de referencia. En una de ellas se colocan el Switch y la Raspberry con las placas SecurePi y SensorPi ya insertadas, el la otra la batería y la placa MotorPi. Por el contrario, en los paneles exteriores verticales se instalan los motores y el sensor de ultrasonidos.

En el caso de los paneles delanteros y traseros se unen sus respectivos componentes aprovechando los soportes creados. Excepto la clavija de red, ya que esta forma parte del primer grupo definido. Una vez montados todos los grupos individualmente se realiza un primer cableado de aquellos componentes que su posterior acceso resulte reducido o nulo, concretamente estamos hablando del encapsulado de 10 leds, el conjunto de carga (conector y portafusibles), el interruptor antivandálico, los sensores de infrarrojos y ultrasonidos y el módulo de la cámara. Llegados a este punto se obtiene el resultado de la figura 4.28

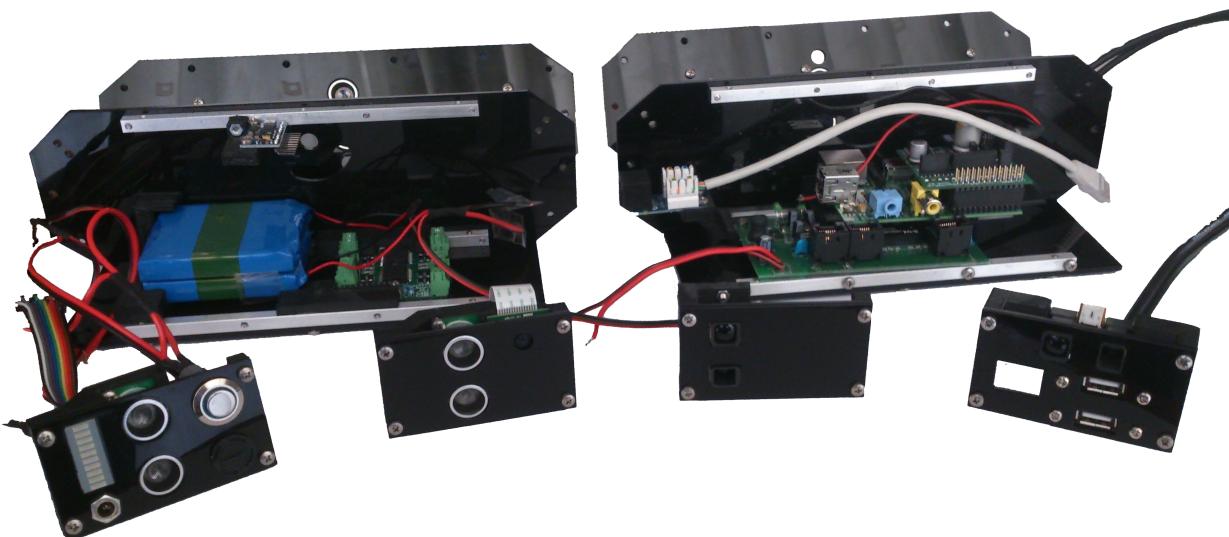


Figura 4.28: Conjunto de paneles montados individualmente.

#### 4.6.2. Conjunto

Finalmente se van acoplando los paneles traseros a cada uno de las escuadras creadas y se hace un primer cableado. A continuación se hace pasar la barra roscada que será la que permita la rotación de cada uno de los hemisferios y se coloca un separador de unos 30mm en el medio. Seguidamente se coloca el panel con los motores con cuidado de respetar las tolerancias y evitando enredos con el cableado. Llegados a este punto colocamos los chaflanes, las tuercas de cabeza redondeada en cada extremo de la barra roscada y las tapas superiores. Para acabar colocamos los acopladores en cada uno de los cuatro ejes e insertamos las llantas en su ranura respetando el sentido del perfil del neumático.



Figura 4.29: Acabado final de la plataforma RospiBot.

## 4.7. Configuración Router WiFi

El router TP-Link WR702N permite operar en vario modos tal y como están definidos en el punto 4.1.4. Debido a la necesidad de que tanto la red inalámbrica como la cableada se puedan comunicar, el único modo posible es que la clavija de red RJ45 esté configurada como LAN. Además el dispositivo ha de ser capaz de crear su red propia y no depender de otras redes. Por este motivo el modo elegido ha sido el de punto de acceso (o Access Point).

La peculiaridad de este punto de acceso es que se ha habilitado el servidor DCHP, lo que permite que cualquier dispositivo que entre en la red se le asigne un IP. Definido el modo de trabajo, se configura la red inalámbrica, para ello se ha elegido el nombre de “**rosplibot**” como SSID. Para la contraseña se ha elegido el mismo nombre que la red y protegida mediante el protocolo WPA2.

MAC Address:	F8-D1-11-9B-1D-23
IP Address:	192.168.100.254
Subnet Mask:	255.255.255.0

**Save**

Figura 4.30: Configuración IP del router.

La IP del router asignada ha sido la **192.168.100.254**, se ha elegido esta numeración para evitar conflictos con las numeraciones más utilizadas (192.168.0.XXX y 192.168.1.XXX). Para la asignación del resto de IPs, se ha establecido un rango, desde 192.168.100.101 hasta 192.168.100.199.

DHCP Server:	<input checked="" type="radio"/> Enable
Start IP Address:	192.168.100.101
End IP Address:	192.168.100.199
Address Lease Time:	2880 minutes (1~2880 minutes, the default value is 120)
Default Gateway:	0.0.0.0 (optional)
Default Domain:	(optional)
Primary DNS:	0.0.0.0 (optional)
Secondary DNS:	0.0.0.0 (optional)

**Save**

Figura 4.31: Configuración servidor DCHP.

## 4.8. Configuración Raspberry Pi

Debido a que el disco duro de la Raspberry Pi reside en la SD, y esta puede ser extraída sin inconvenientes, la configuración que a continuación se explica se ha efectuado mediante una segunda Raspberry Pi conectada a internet. Una vez finalizada la configuración, se ha insertado la SD de nuevo en el robot (Para tener acceso a la SD se ha de retirar el panel inferior del hemisferio derecho).

### 4.8.1. Grabación Raspbian

El primer paso realizado ha sido la grabación del sistema operativo **Raspbian**, el más extendido en la comunidad (concretamente se ha utilizado la versión más actual, la de Enero 2014). La imagen de este sistema ocupa un espacio de unos 2.75GB, pero para un correcto funcionamiento se requiere un SD de 4GB mínimo, 8GB recomendado. De los diferentes procedimientos para la obtención de nuestra SD, se ha optado por el programa **Win32 Disk Imager**. Una vez grabada la SD, en Windows solo aparecerá la partición NTFS de 57MB.

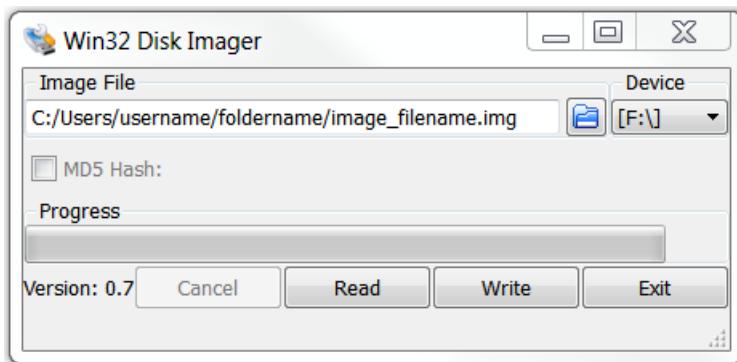


Figura 4.32: Interfície del programa Win32 Disk Imager.

El sistema operativo ha obtenido de la página oficial <http://www.raspberrypi.org/downloads/> y viene con una cuenta creada con el usuario **pi** y contraseña **raspberry**. Durante toda la realización del proyecto se ha utilizado esta cuenta de usuario manteniendo su contraseña. Además de esta distribución podemos encontrar otros sistemas compatibles con la Raspberry Pi.

### 4.8.2. Configuración inicial

Con la imagen del sistema cargada en la SD, introducimos la tarjeta de memoria en la Raspberry Pi, conectamos el cable de red y alimentamos el conjunto. Al haber activado el servidor DHCP, por defecto el router le asignará una IP aleatoria, en este caso según la figura 4.33, (192.168.1.105). Con esta IP nos conectamos mediante SSH. Para ello podemos utilizar diversos métodos, en este proyecto se ha usado el comando **ssh**

**pi@192.168.1.105** en entorno Linux y el programa **Putty** en Windows. En el caso de Linux, solamente nos pedirán la contraseña, en cambio en Windows hay que especificar el puerto 22 y luego el usuario y contraseña. Una vez hemos accedido vía ssh, nos aparecerá un terminal idéntico al de la figura 4.34.

DHCP Clients List			
ID	Client Name	MAC Address	Assigned IP
1	raspberrypi	B8-27-EB-08-09-3B	192.168.1.105

Figura 4.33: Assignación de IP por DHCP.

Una vez dentro de la Raspberry Pi, procedemos a su configuración inicial. Para ello tecleamos el comando **sudo raspi-config**. A continuación apareceremos en un menú con diversas opciones como el de la figura 4.35. Lo primero que haremos será aprovechar el máximo la capacidad de nuestra SD con la opción **Expand Filesystem**. Indicaremos que al iniciar no active el modo gráfico, habilitaremos la cámara y por último, en **Advanced Options** modificaremos el nombre del dispositivo (o hostname) a **"rosplibot"**.

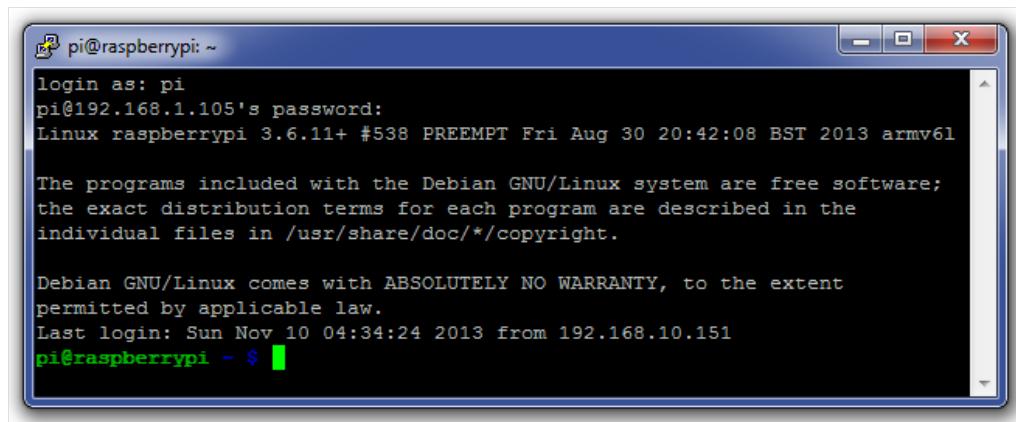


Figura 4.34: Consola de acceso remotor por SSH.

Aunque también existe la opción de aumentar la frecuencia de la CPU hasta 1GHz, de momento se dejará con el valor predefinido de 700MHz. En el caso de necesitar una mejora se procederá a aumentar progresivamente la frecuencia usando la opción **Overclock**. Una vez definidos todos estos cambios se reinicia la placa, iniciando el sistema con la nueva configuración pero teniendo en cuenta que posiblemente la IP sea diferente.

#### 4.8.3. Instalación de librerías

Con el S.O. Raspbian debidamente configurado procedemos a la instalación de herramientas, módulos y librerías. Primero de todo actualizaremos nuestras fuentes con

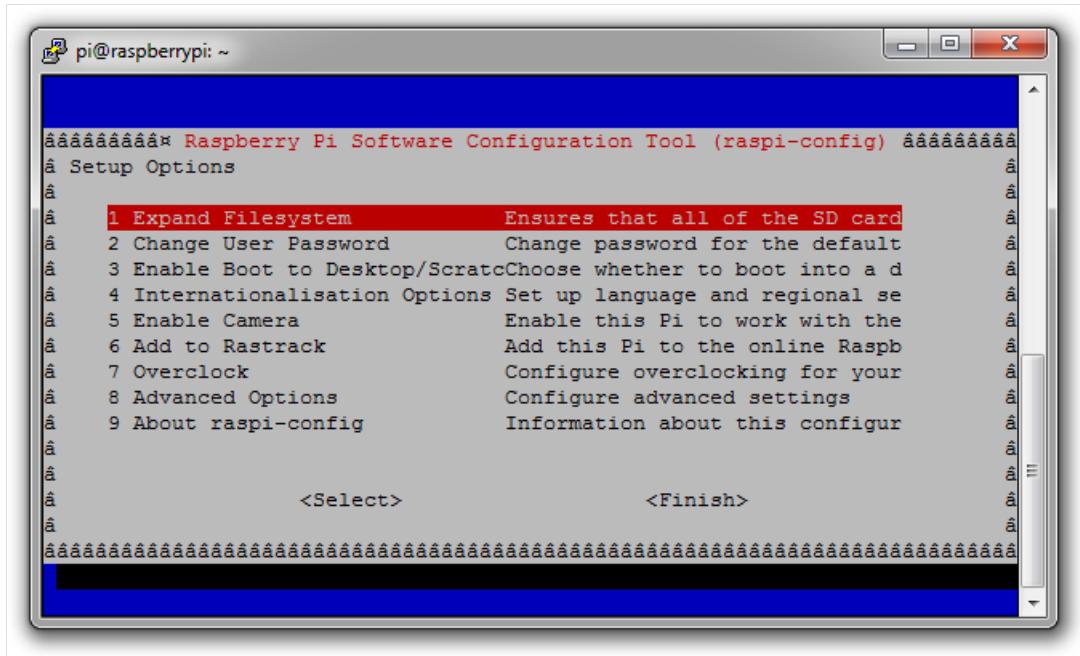


Figura 4.35: Menú de opciones de configuración de Raspbian.

el comando **`sudo apt-get update`**, de esta manera sabremos dónde encontrar aquello que busquemos. A continuación se irán detallando aquellas herramientas instaladas.

- **`sudo apt-get install xrdp`** Permite acceder por entorno gráfico mediante el uso del Escritorio Remoto de Windows.
- **`sudo apt-get install python-numpy`** Módulo de Python, ofrece un mayor soporte para vectores y matrices.
- **`sudo apt-get install python-scipy`** Módulo de Python, orientado a optimización, álgebra lineal y procesamiento de señales y de imagen entre otros.
- **`sudo apt-get install python-matplotlib`** Módulo de Python, permite generar gráficos a partir de listas o vectores.
- **`sudo apt-get install python-imaging`** Módulo de Python, añade herramientas para procesamiento de imágenes.
- **`sudo apt-get install memcached python-memcached`** Programa y módulo de Python, permite crear y compartir variables en la memoria caché.

Para el uso de bus I2C y SPI que dispone el componente BCM2708, se han de seguir los siguientes pasos: Primero modificamos el archivo **`raspi-blacklist.conf`**, para ello ejecutamos el comando **`sudo nano /etc/modprobe.d/raspi-blacklist.conf`** y comentamos con una almohadilla los dos dispositivos de la lista. Tal y como aparece en la figura 4.36.

```
pi@rosplibot: ~
GNU nano 2.2.6  File: /etc/modprobe.d/raspi-blacklist.conf

# blacklist spi and i2c by default (many users don't need them)

#blacklist spi-bcm2708
#blacklist i2c-bcm2708
```

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos  
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Tex ^T To Spell

Figura 4.36: Archivo de configuración de los dispositivos I<sup>2</sup>C y SPI.

```
pi@rosplibot: ~
GNU nano 2.2.6          File: /etc/modules          Modified

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd-bcm2835
cuse
i2c-bcm2708
i2c-dev

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Tex ^T To Spell
```

Figura 4.37: Archivo de configuración de los módulos.

El siguiente paso consiste en modificar el archivo **modules**, abrimos el archivo con **sudo nano /etc/modules** y añadimos al final **i2c-bcm2708** y **i2c-dev**.

A continuación instalamos un nuevo grupo de herramientas para el uso de estos dos buses de comunicación:

- **sudo apt-get install i2c-tools** Herramientas y comandos para utilizar el bus I<sup>2</sup>C.
- **sudo apt-get install python-smbus** Módulo de Python, ofrece funciones para trabajar más cómodamente con el bus I<sup>2</sup>C.
- **sudo apt-get install python-dev** Módulo de Python, conjunto de librerías estáticas orientadas al desarrollo.

Ahora sólo nos hace falta el módulo orientado para la comunicación por SPI, en este caso se va a usar un módulo creado por un desarrollador independiente. Para ello crea-

remos una carpeta donde almacenar su código, lo descargaremos y los instalaremos. Los comandos a ejecutar son:

- `mkdir python-spi`
- `cd python-spi`
- `wget https://raw.githubusercontent.com/doceme/py-spidev/master/setup.py`
- `wget https://raw.githubusercontent.com/doceme/py-spidev/master/spidev_module.c`
- `sudo python setup.py install`

Una vez tenemos nuestra Raspberry Pi ya configurada y con las librerías instaladas, reiniciamos para que los cambios surtan efecto.

## 4.9. Programas básicos

Para que el usuario pueda trabajar cómodamente con el robot, se han creado un conjunto de programas encargados de comunicarse directamente con los dispositivos. Estos programas se han distribuido en dos carpetas dentro del directorio **/home/pi: Modules**, incluye todas las librerías auxiliares creadas por otros miembros de la comunidad Raspberry Pi, y **Publishers**, dónde encontramos los programas que se comunican los dispositivos y almacenan la información en la memoria caché.

Todos los códigos que aquí se mencionan pueden allarse en el repositorio **GitHub** de este proyecto, <https://github.com/Guasch5000/RosPiBot>. A continuación se detallan los programas ubicados en la carpeta **Publishers**, cual es su función, que parámetros intervienen y de que manera almacena la información.

- **readsensors.py** Este módulo se encarga de la lectura de los sensores que se comunican por el bus I<sup>2</sup>C, donde encontramos la lectura del estado de la batería, la de corriente de cada motor, los dos sensores de distancia por infrarrojos, los cuatro por ultrasonidos y la imu. Debido a que cada sensor requiere de ser muestrado a frecuencias diferentes, lo primero que encontramos en el código es una tabla para definir estos valores en segundos. Seguidamente se definen las direcciones de cada dispositivo y se declaran las funciones de lectura.

La lectura de los sonars se publica en milímetros debido a que esa es su resolución, con un máximo de 5192mm. Por el contrario los infrarrojos se publican en centímetros, siendo su máximo valor de 1,51cm. En el caso de la batería, se ha pasado su voltaje a un porcentaje, siendo 6V el 0 % y 8,3V el 100 %. Para los sensores de corriente, se almacena el valor en miliamperios, con un máximo de 4000mA. Finalmente, los valores procedentes de la imu se han almacenado según se han recibido, las aceleraciones en G ( $9,8\text{m}\cdot\text{s}^{-2}$ ), las velocidades angulares en  $\text{rad}\cdot\text{s}^{-1}$  y la orientación en radianes.

- **readencoders.py** Diseñado para comunicarse con los descodificadores de cuadratura. Puesto que almacenar el recorrido de cada rueda no es útil para sistemas no holónomos, este programa recoge el recuento de cada codificador y mediante una diferencia se extrae la velocidad de cada motor delantero. Para estandarizar los valores, estos se almacenarán en radianes por segundo, el sentido positivo indica que la rueda ejerce una movimiento hacia adelante.

- **movemotors.py** A partir de dos variables almacenadas en la memoria cache (speed\_1,

speed\_r) con un rango de valores de -100 hasta 100, se genera señales PWM para cada motor. El sentido de giro de los motores es el mismo para cada lado, debido que hacer girar las ruedas de un mismo hemisferio en sentidos contrarios implica contraponer los dos motores y desaprovechar la energía.

- **readjoystick.py** Este último programa se espera a que el mando inalámbrico F710 de Logitech esté conectando y envié señal. Una vez conectado el programa lee los eventos generados por el mando y modifica los registros de la cache relacionados con la velocidad de los motores. Para poder conducir el robot mediante el mando es necesario mantener el botón **LB**, luego con el joystick izquierdo se controla el sentido de ir adelante o atrás, por el contrario, con el derecho se controla el sentido de rotación.

## 4.10. Módulo Rospibot

Puesto que el objetivo de este proyecto es ofrecer una plataforma robótica con las herramientas necesarias para trabajar con ella, se ha creado el módulo **rospibot** de python para poder interaccionar con todas las variables del sistema. Dicho módulo se encuentra alojado en el directorio **Modules** (junto otras librerías).

Para importar este módulo desde cualquier ubicación se ha modificado la variable de entorno **PYTHONPATH**, añadiendo el directorio **/home/pi/Modules** a dicha variable. Para ello se ha ejecutado el comando **export PYTHONPATH=\$PYTHONPATH:/home/pi/Modules**. Con este cambio cualquier usuario puede importar los nuevos módulos, excepto el usuario **root**. Para que el usuario root también pueda hacer uso, se ha editado el archivo **/etc/sudoers** y se ha incorporado la línea **Define env\_keep += \$PYTHONPATH**. A continuación se detallan todas la funciones que forman el módulo rospibot, estas funciones simplemente sirven de enlace con la memoria cache:

- **get\_sonar\_front()** Devuelve en milímetros la lectura del sensor de distancia por ultrasonidos orientado hacia adelante. Del mismo modo, para el sensor trasero debemos cambiar **\_front()** por **\_back()**, **\_left()** para el izquierdo y **\_right()** para el derecho.
- **get\_orientation()** Devuelve en radianes un vector de tres posiciones correspondiente a la orientación respecto el suelo. El primer valor corresponde al "pitch" o rotación respecto el eje *y*, el segundo al "roll", rotación respecto *x* y el último al "yaw", eje *z*.
- **get\_acce()** Devuelve en G ( $9.8\text{m}\cdot\text{s}^{-2}$ ) las aceleraciones de cada eje, siguiendo el orden *x*, *y* y *z*.
- **get\_acce()** Devuelve en radianes por segundo la velocidad angular de cada eje siguiendo el mismo orden que la función anterior.
- **get\_infrared\_front()** Devuelve en centímetros la distancia de un obtáculo a partir del sensor de infrarrojos. Del mismo modo que los sonars, para la lectura del sensor trasero únicamente se deberá modificar **\_front()** por **\_back()**.

- **get\_battery()** Devuelve en tanto por ciento (un número entero) el estado de la batería.
- **get\_current\_left\_front()** Devuelve en miliamperios el consumo de corriente del motor delantero izquierdo. Para el motor delantero derecho **\_right\_front()** y para los traseros **\_left\_back()** y **\_right\_back()**.
- **get\_rads\_left()**: Devuelve en radianes por segundo la velocidad de la rueda delantera izquierda. En condiciones de experimentación se ha observado que el valor máximo en carga es de aproximadamente unos  $7\text{rad}\cdot\text{s}^{-1}$ . En el caso de querer conocer la velocidad de la otra rueda, deberemos cambiar **\_left()** por **\_right()**.
- **set\_speed(left\_speed, right\_speed)** Al contrario que el resto de funciones, esta se encarga de modificar el registro. para ello será obligatorio definir las dos velocidad con un valor de entre -100 y 100. La velocidad aplicada por el usuario puede no corresponder con la real puesto que se trata de un control de lazo abierto.

## 4.11. Instalación ROS Groovy

La peculiaridad de este robot respecto otros es el hecho que en su interior puede ejecutarse un núcleo de ROS (roscore). Otras plataformas recurren a un "intérprete" entre la información que procedente de la plataforma y el formato aceptado por ROS. Esta característica permite que el robot funcione independientemente de un ordenador externo a la hora de trabajar con ROS.

De las versiones existentes en la realización del proyecto, sólo habían dos posibilidades, **Groovy** o **Hydro** (la distribución de Indigo aún se encontraba en desarrollo). De entre las dos versiones disponibles se eligió Groovy debido a que ofrecía soporte directo a Raspbian.

Este proyecto finaliza con la instalación y configuración de ROS Groovy en la plataforma RosPiBot. Puesto que las variables del robot se pueden acceder con el módulo creado, no aparece la necesidad de crear tópicos que publiquen esta información. Aún así, puede convertirse en un primer contacto con ROS para futuros usuarios. A continuación se detallan los pasos realizados, aunque estos pasos ya quedan explicados en la página <http://wiki.ros.org/groovy/Installation/Raspbian>, existe una diferencia que puede conllevar problemas si se desea realizar los tutoriales. Por ello se explica todo el proceso hasta la creación del paquete "beginner\_tutorials" incluido.

### 4.11.1. Instalación básica

- `sudo sh -c 'echo "deb http://64.91.227.57/repos/rospbian wheezy main" > /etc/apt/sources.list.d/rospbian.list'`
- `wget http://64.91.227.57/repos/rospbian.key -O - | sudo apt-key add -`
- `sudo apt-get update`
- `sudo apt-get install ros-groovy-ros-comm`

### 4.11.2. Inicializar dependencias y librerías

- `sudo rosdep init`
- `rosdep update`
- `echo "source /opt/ros/groovy/setup.bash" >> /.bashrc`
- `source /.bashrc`

- `sudo apt-get install python-rosinstall`
- `sudo apt-get install ros-groovy-ros-tutorials`

#### 4.11.3. Creación directorio de trabajo

- `mkdir -p /ROS/catkin_ws/src`
- `cd /ROS/catkin_ws/src`
- `catkin_init_workspace`
- `cd /ROS/catkin_ws`
- `catkin_make`
- `source devel/setup.bash`

#### 4.11.4. Creación paquete "beginner\_tutorials"

- `cd /ROS/catkin_ws/src`
- `catkin_create_pkg beginner_tutorials std_msgs rospy roscpp`
- `catkin_make -DCMAKE_BUILD_TYPE=Release`
- `catkin_make install`
- `catkin_make install -DCMAKE_INSTALL_PREFIX=/opt/ros/groovy`
- `echo "source /home/pi/ROS/catkin_ws/devel/setup.bash" >> /.bashrc`
- `source /.bashrc`

## 5. Presupuesto

El coste total de este proyecto asciende hasta los **5139,69 euros**. El material utilizado implica unos **523,69 euros** (un 10,2 % del coste total) y el coste personal unos **4616,00 euros** (el 89,8 % restante). A continuación se detallan los costes del proyecto.

### 5.1. Coste Material

De los 523,69 euros invertidos en este proyecto para la adquisición de material, se han distribuido según la tabla 5.1. En los siguientes sub-apartados se define los componentes.

Concepto	Coste [€]	Porcentaje [%]
Dispositivos	234.05	44.7
Electrónica	112.59	21.5
Mecánica	106.63	20.4
SecurePi	25.49	4.9
SensorPi	24.10	4.6
MotorPi	20.83	4.0
<b>Total</b>	<b>523,69</b>	<b>100,0</b>

Tabla 5.1: Distribución del presupuesto de material

#### 5.1.1. Coste de Dispositivos

Concepto	Coste unitario [€/u]	Unidades	Total [€]
Raspberry Pi tipo B	40,96	1	40,96
Módulo cámara	28,92	1	28,92
SD 8GB	7,19	1	7,19
Mando Logitech F710	60,44	1	60,44
Router TL-WR720N	21,68	1	21,68
Cable Ethernet Cat5	2,30	3	6,90
Extensor Ethernet	3,96	1	3,96
Extensor USB	2.12	2	4,24
Sonar KS103	11,50	4	46,00
IMU GY-80	13,79	1	13,79
<b>Total</b>			<b>234,05</b>

Tabla 5.2: Coste detallado de cada dispositivo

### 5.1.2. Coste de la Electrónica

Concepto	Coste unitario [€/u]	Unidades	Total [€]
Batería LiPo 7,4V 11.000mAh	78,59	1	78,59
Interruptor antivandálico	11,49	1	11,49
Barra 10 leds	8,05	1	8,05
Portafusible	3,87	1	3,87
Cableado	10,59	1	10,59
<b>Total</b>			<b>112,59</b>

Tabla 5.3: Coste detallado de la electrónica

### 5.1.3. Coste placa SecurePi

Concepto	Coste unitario [€/u]	Unidades	Total [€]
PCB SecurePi	1,94	1	1,94
Regulador LM2596S-5.0	4,22	1	4,22
Diodo Schottky 50V 4A	0,52	1	0,52
Bobina 22uH	1,11	1	1,11
Condensador 680uF electrolítico	0,92	1	0,92
Condensador 220uF electrolítico	0,73	1	0,73
Resisténcia 10Kohms SMD 1206	0,005	32	0,15
Mosfet BSS138	0,21	16	3,36
Conecotor barrel Jack 5.2mm	2,51	1	2,51
Conecotor Hembra 18x2 pines	4,84	1	4,84
Conecotor Hembra 8 pines	1,14	1	1,14
Conecotor Hembra 5 pines	0,91	1	0,91
Conecotor Hembra 4 pines	0,82	1	0,82
Conecotor Hembra 3 pines	0,77	1	0,77
Conecotor Hembra 2 pines	0,73	1	0,73
Conecotor Macho 13x2 pines	0,53	1	0,53
<b>Total</b>			<b>25,49</b>

Tabla 5.4: Coste detallado placa SecurePi

### 5.1.4. Coste de la Mecánica

Concepto	Coste unitario [€/u]	Unidades	Total [€]
Paneles Metacrilato	99,75	1	99,75
DIN 912 INOX A4 4x10	0,23	4	0,92
DIN 913 5x5	0,10	4	0,40
DIN 934 M2	0,07	2	0,14
DIN 934 M3	0,04	80	3,2
DIN 1587 M8	0,91	2	1,82
DIN 7985 A2 3x10	0,03	102	3,06
DIN 7985 A4 3x20	0,05	11	0,55
<b>Total</b>			<b>106,63</b>

Tabla 5.5: Coste detallado de la mecánica

### 5.1.5. Coste placa SensorPi

Concepto	Coste unitario [€/u]	Unidades	Total [€]
PCB SensorPi	1,94	1	1,94
Led ámbar	0,11	8	0,88
Led verde	0,05	1	0,05
Resisténcia 1Mohms 1/4W SMD 1206	0,01	2	0,02
Resisténcia 10Kohms 1/4W SMD 1206	0,005	12	0,06
Resisténcia 6,9Kohms 1/4W SMD 1206	0,007	4	0,03
Resisténcia 2,49Kohms 1/4W SMD 1206	0,008	2	0,02
Resisténcia 1Mohms 1/4W SMD 1206	0,01	2	0,02
Resisténcia 1Mohms 1/4W SMD 1206	0,01	2	0,02
Condensador 10uF 10V cerámico SMD 1206	0,19	2	0,38
Condensador 0,1uF 10V cerámico SMD 1206	0,03	2	0,06
Condensador 18pF 50V cerámico SMD 1206	0,06	2	0,12
Integrado MCP3424 ADC Delta-Sigma SOIC	3,79	2	7,58
Cristal 40MHz 30ppm	0,53	2	1,06
Integrado LS7366S descodificador SOIC	4,23	2	8,46
Conector macho 8 pin largo	1,05	3	3,15
Conector macho 40 pin	0,24	1	0,24
<b>Total</b>			<b>24,10</b>

Tabla 5.6: Coste detallado placa SensorPi

### 5.1.6. Coste placa MotorPi

Concepto	Coste unitario [€/u]	Unidades	Total [€]
PCB MotorPi	1,94	1	1,94
Integrado LM3914	2,34	1	2,34
Resisténcia 10Kohms 1 % DIP	0,03	2	0,06
Resisténcia 11Kohms 1 % DIP	0,03	1	0,03
Resisténcia 13Kohms 1 % DIP	0,10	1	0,10
Resisténcia 1Kohms SMD 1206	0,007	4	0,03
Resisténcia Shunt 0,5Ohms 2W SMD	1,91	4	4,74
Condensador 1uF 16V Ceramico	0,06	1	0,06
Bloque Terminal doble	0,38	5	1,90
Diodo Schottky 40V 5A	0,29	4	1,17
Driver L298P	4,24	2	8,48
<b>Total</b>			<b>20,83</b>

Tabla 5.7: Coste detallado placa MotorPi

## 5.2. Coste Personal

Aunque en un proyecto de este tipo el presupuesto en personal no influyen en el gasto total, es recomendable conocer en que cantidades rondaría el proyecto si fuese de ámbito comercial. Para realizar dicha contabilidad se ha segmentado el coste personal según su labor

Labor	Coste horario [€/h]	Horas	Total [€]
Estudio de viabilidad	40	53	2.120
Diseño circuito impreso	26	12	312
Diseño asistido por ordenador	24	22	528
Soldadura PCB	18	4	72
Montaje y cableado	16	14	224
programación	20	68	1.360
<b>Total</b>			<b>4.616</b>

Tabla 5.8: Coste detallado del personal necesario

### 5.2.1. Descripción de las labores

- **Estudio de viabilidad** Analizar el producto WiFiBot, buscar mejoras, localizar nuevo componentes que ofrezcan mayor utilidad y estudiar su compatibilidad entre ellos.
- **Diseño Circuito Impreso** Encontrar la mejor distribución de los componentes encontrados en el estudio de viabilidad, de tal manera que su producción sea fácil para sus distribuidores y montadores.
- **Diseño Asistido por ordenador** Ubicar los diferentes componentes dentro del robot, realizar un modelo por ordenador de todos los componentes y Diseñar el resultado final.
- **Soldadura de PCB** Montaje de las placas que conforman el proyecto y comprobación de su funcionamiento.
- **Montaje y Cableado** Unificación de todos los elementos que forman el robot, rectificación de roscas y acabado final.
- **Programación** Configuración del sistema operativo, creación de programas para la comunicación con los diferentes dispositivos y de un conjunto de librerías para mejorar la comodidad al usuario.

### 5.3. Planificación

Este proyecto se ha realizando en un periodo de 15 semanas. En el diagrama de la figura 5.1 se aprecia como determinadas labores se han realizado al mismo tiempo. El motivo de dicha situación se debe al continuo desarrollo de los diferentes sectores y del hecho que decisiones en un sector pueden afectar a otros directamente.

Para la programación final del robot se esperó a disponer de la plataforma totalmente ensamblada, con el fin de poder realizar las pruebas directamente sobre el robot. De esta manera se evitaba repetir las comprobaciones ya que su elaboración previa no implicaría un ahorro de tiempo.

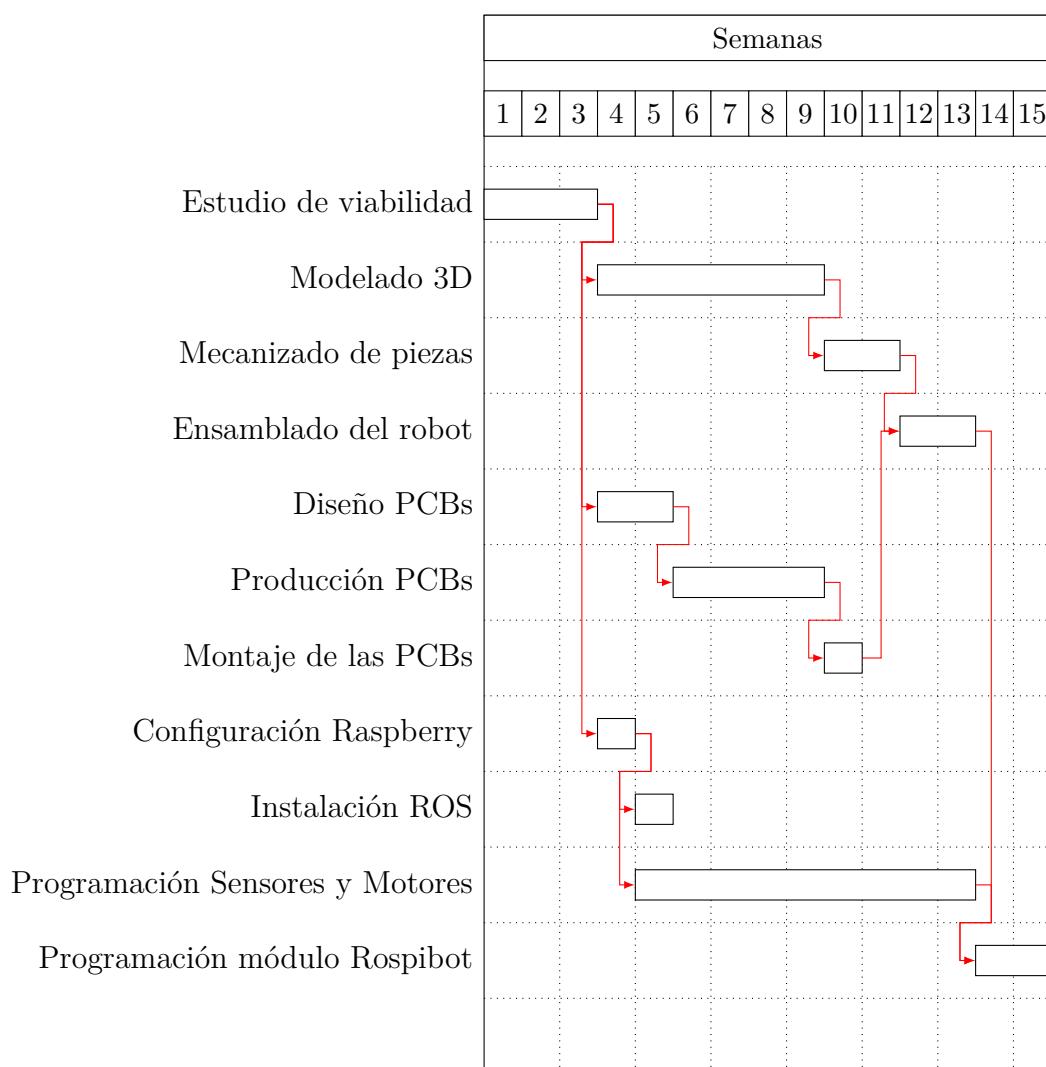


Figura 5.1: Diagrama de Gantt

## 6. Impacto Ambiental

Aunque este proyecto trate de la restauración de un plataforma móvil que estaba en desuso, hay que tener en cuenta que existe la posibilidad que su mejora suponga un empeoramiento del medioambiente. Para ello se ha tenido en cuenta aquello de lo que nos deshacemos y de lo que añadimos.

Los paneles originales del WiFiBot están constituidos de Policarbonato, este tipo de plástico se recicla alrededor del 90 % en países europeos. Su impacto en el ambiente es reducido y se puede encontrar en aplicaciones sanitarias. Aun así, desde hace unos años se ha comprobado que en condiciones de degradación este plástico libera moléculas de Bisfenol A, que puede causar anomalías durante el desarrollo de embriones.

Otro elemento del que nos desharemos será la circuitería original. Debido al uso de estaño con plomo hasta 2006 por la directiva RoHS, muy probablemente que contenga este metal pesado. Además de plomo, a causa de la complejidad de la circuitería empleada es posible que también disponga de otros metales nocivos como el zinc, cobalto o níquel entre otros. Para minimizar su impacto, los circuitos serán entregados en un punto especial de recogida de este tipo de material para su reciclaje.

Además de la circuitería, las baterías también son otro elemento a tener en cuenta. Aunque la eliminación inadecuada de las baterías de NiMH representa un gran peligro ambiental menor que la de NiCd debido a la ausencia de cadmio tóxico. La extracción y su procesamiento puede plantear otros tipos de impacto ambiental, sin embargo, la mayoría de níquel industrial es reciclado, debido a la relativamente fácil recuperación mediante electroimanes. Sobre el níquel, se ha de tener en cuenta que tiene efectos sobre la piel y que puede producir cáncer de pulmón si se inhala en altas cantidades.

Sobre los componentes introducidos, la gran mayoría cumplen la directiva RoHS actual. El único componente que puede introducir un poco de sospecha son las baterías. Estas baterías están compuestas por litio, un elemento neurotóxico y dañino para el riñón. Aunque el litio puede conllevar riesgo, su impacto ambiental es mínimo. En Estados Unidos se ha declarado que se pueden desechar de forma segura como residuo normal, por el contrario, los gobiernos europeos advierten del potencial para contaminar aguas subterráneas debido a que contienen metales base como el cobalto. Para extender el uso de las baterías, contamos con los módulos PCM. Estos módulos nos ayudarán a alargar la vida de nuestras celdas de energía.

## 7. Conclusiones

Aunque restaurar una plataforma y volverla a poner al día ha sido un labor que conllevado varios quebraderos de cabeza, este proyecto demuestra que es realizable. Durante la ejecución de este proyecto se ha observado que las plataformas robóticas se quedan obsoletas a causa de sus unidades de procesamiento desfasadas, mientras que los componentes más básicos como los controladores de motores o los sensores siguen estando operativos. Todo esto hace ver que quizá hace falta un cambio en el razonamiento de la robótica, de tal manera que las unidades de procesamiento puedan ser actualizadas sin afectar al resto de los elementos de un robot.

A medida que se ha ido realizando el proyecto, trabajando con el robot, pensando mejoras y aplicándolas, uno se da cuenta de las deficiencias del producto original y comprende la evolución que ha seguido la plataforma WifiBot hasta el presente. El cambio más notorio es el hecho que las nuevas plataformas creadas por la empresa francesa están constituidas por un único bloque (sin la necesidad de crear dos hemisferios). Durante el diseño del RosPiBot, en el momento de la ubicar sus componentes, se destino gran parte del tiempo a encontrar la manera que el switch, la Raspberry Pi y el router pudiesen convivir en el mismo espacio. Además, disponer de una barra metálica que atraviesa transversalmente el robot, dificultó enormemente su desarrollo.

Por último, el hecho de "Rosificar" cualquier plataforma robótica es una apuesta segura hacia su desarrollo. La gran comunidad que hay detrás de ROS y el nivel de conocimientos del que dispone, son motivos suficientes utilizar esta herramienta. Incluso con plataformas desactualizadas, añadir ROS puede originar un resurgimiento.

## 8. Agradecimientos

Un proyecto como este no se podría haber realizado sin el apoyo de familiares ni amigos. Ya que son ellos los que me han prestado ayuda y me han dado ánimos durante toda la realización de este trabajo.

Agradecer la oportunidad que los profesores **Cecilio Angulo y Manuel Velasco** me han ofrecido al permitirme poder realizar todas las modificaciones deseadas en el WiFiBot. También por el apoyo a la hora de resolver dudas o aportar ideas al proyecto. Además agradecer a los miembros de la asociación **AESS Estudiants** todo el tiempo que me han dedicado para que este proyecto fuese posible. Compartiendo su experiencia para solucionar los problemas que aparecían dia a dia.

## Referencias

- [1] MARTÍNEZ, A. Y FERNÁNDEZ, E., *Learning ROS for Robotics Programming*. PACKT Publishing, 2013
- [2] NEXTER ROBOTICS, *Wifibot4G Quick Start Guide* Francia, 2006 <http://rbouraoui.free.fr/wifibot/wifibot4Genglishguide.pdf>
- [3] LYNXMOTION, *Data sheet for: GHM-04 7.2vdc 50:1 175rpm 6mm shaft* Estados Unidos, 2006 [http://vallejo.cc/proyectos/cellbot/motor\\_lynxmotion\\_7\\_2V.pdf](http://vallejo.cc/proyectos/cellbot/motor_lynxmotion_7_2V.pdf)
- [4] US DIGITAL, *E4P OEM Miniature Optical Kit Encoder* Estados Unidos, 2005 [http://www.usdigital.com/assets/datasheets/E4P\\_datasheet.pdf?k=635367475934568087](http://www.usdigital.com/assets/datasheets/E4P_datasheet.pdf?k=635367475934568087)
- [5] KINGSIN TECHNOLOGIES Co., *KS101B/KS103/KS103S Ultra Sonic Range Finder Technical Specification* China, 2011 [http://www.dauxi.com/KS10X-V110\\_EN.pdf](http://www.dauxi.com/KS10X-V110_EN.pdf)
- [6] TP-LINK, *TL-WR702N 150Mbps Wireless N Nano Router User Guide* China, 2014 [http://www.tp-link.com/resources/document/TL-WR702N\\_V1\\_User\\_Guide\\_1910010880.pdf](http://www.tp-link.com/resources/document/TL-WR702N_V1_User_Guide_1910010880.pdf)
- [7] LSI COMPUTER SYSTEMS, INC., *32-Bit Quadrature Counter With Serial Interface* Estados Unidos, 2012 [http://www.lsicsi.com/pdfs/Data\\_Sheets/LS7366R.pdf](http://www.lsicsi.com/pdfs/Data_Sheets/LS7366R.pdf)
- [8] PHILIPS SEMICONDUCTORS N.V., *Bi-directional level shifter for I<sup>2</sup>C-bus and other systems. Application Note AN97055* Países Bajos, 1997 <http://www.adafruit.com/datasheets/an97055.pdf>
- [9] FAIRCHILD SEMICONDUCTORS, *BSS138 N-Channel Logic Level Enhancement Mode Field Effect Transistor* Estados Unidos, 2005 <http://www.adafruit.com/datasheets/BSS138.pdf>
- [10] MICROCHIP TECHNOLOGY INC., *16-Bit, Multi-Channel ΔΣ Analog-to-Digital Converter with I<sup>2</sup>C Interface and On-Board Reference* Estados Unidos, 2009 <http://ww1.microchip.com/downloads/en/DeviceDoc/22226a.pdf>
- [11] TEXAS INSTRUMENTS INC., *LM3914 Dot/Bar Display Driver* Estados Unidos, 2000 <http://www.ti.com/lit/ds/symlink/lm3914.pdf>
- [12] TEXAS INSTRUMENTS INC., *LM3914 Dot/Bar Display Driver* Estados Unidos, 2000 <http://www.ti.com/lit/ds/symlink/lm3914.pdf>
- [13] TEXAS INSTRUMENTS INC., *LM2596 SIMPLE SWITCHER Power Converter 150 kHz 3A Step-Down Voltage Regulator* Estados Unidos, 1999 <http://www.ti.com/lit/ds/symlink/lm2596.pdf>

- [14] STMICROELECTRONICS N.V., *L298 DUAL FULL-BRIDGE DRIVER* Países Bajos, 2000 [https://www.sparkfun.com/datasheets/Robotics/L298\\_H\\_Bridge.pdf](https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf)

## Soporte infromático

**Filezilla** Linux, Mac, Windows

Cliente FTP multiplataforma de código abierto y software libre. Permite la transferencia de archivos entre ordenadores.

**LATEX** Linux, Mac y Windows

Sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas.

**KiCad** Linux y Windows

Entorno de software usado para el diseño de circuitos eléctricos y de código abierto.

**Photoshop CS2** Windows

Editor de gráficos rasterizados desarrollado para el retoque de fotografías y gráficos.

**Putty** Linux, Mac y Windows

Cliente SSH, Telnet, rlogin, y TCP raw con licencia libre. Permite conexión remota con otro ordenador.

**Solidworks 2010** Windows

Programa de diseño asistido por computadora para modelado mecánico.

