# Navegación de un robot con láser y sonar

Adrià García Martín

**Resumen**—En este proyecto se realizará la programación de un robot de manera que navegue autónomamente por un espacio determinado sin colisionar con los objetos del entorno. El robot contará con una Raspberry Pi donde irán conectados todos los componentes: los motores de las ruedas, el sonar y el láser. El sistema operativo que usará el robot será ROS (Robot Operating System), lo que nos permitirá, además, explorar diversas ampliaciones posibles al proyecto, gracias a las posibilidades que ofrece su estructura. Se explicarán los objetivos planteados, el desarrollo y la metodología utilizada, así como los resultados obtenidos y las conclusiones.

**Palabras clave**—Robot, navegación, autónomo, ROS, Raspberry Pi, láser, sonar, Python.

**Abstract**—This project will talk about how to programe a robot so it can navigate autonomously around a limited space without colliding with surrounding objects. The robot would have a Raspberry Pi where all components would be connected: wheel motors, the sonar and the laser. ROS (Robot Operating System) would be the operation system which would be used, so it would allow us to explore some possible extensions to this project, because of ROS structure. This document would explain the project objectives, development and methodology as far as the results and conclusions obtained.

**Index Terms**—Robot, navigation, autonomous, ROS, Raspberry Pi, laser, sonar, Python.

## 1 Introducción

En este documento se explicará cómo se ha llevado a cabo el proyecto "Navegación de un robot con láser y sonar". Para ello se planteará a continuación tanto la motivación de este proyecto como los objetivos marcados. Seguidamente se explicarán los conceptos generales sobre el estado del arte, los conceptos necesarios para entender cómo y porqué se ha desarrollado de esta manera el robot, y posteriormente se hablará del desarrollo del proyecto, explicando de qué componentes consta el robot, haciendo hincapié en la metodología utilizada así como mostrando los resultados obtenidos. Finalmente se expresarán las conclusiones finales del proyecto además de comentar posibles extensiones de cara al futuro.

### 1.1 Motivación

Hoy en día la informática está presente en cualquier aspecto cotidiano de la vida de una persona de nuestra sociedad. Y dentro del campo de la informática, considero que la robótica es una de las vertientes más interesantes e imaginativas. La robótica es sinónimo de progreso y desarrollo tecnológico y la mayoría de robots tienen como finalidad el beneficio de los ciudadanos: desde robots de asistencia personal a robots de seguridad, médicos, de mantenimiento, etc.

¿Quién no se ha imaginado alguna vez viviendo en una sociedad donde los robots facilitan la vida a los humanos? La presencia de robots en una sociedad se puede considerar un avance y un salto cualitativo del nivel de vida.

No obstante, pueden verse muchas clases de robots como he mencionado, por lo que en este proyecto simplemente intenta dar un primer paso dentro de la robótica, tratando con un robot simple que puede servir como base para posteriores ampliaciones de funcionalidades. Este robot simple se basará en realizar ciertas acciones según una percepción, en base a un objetivo determinado.

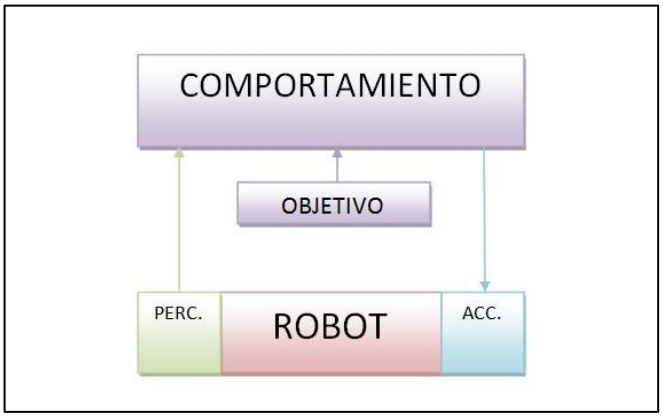


Fig. 1. Esquema a alto nivel del funcionamiento del robot: al recibir una percepción realiza una acción según un comportamiento determinado que viene dado por un objetivo a conseguir.


- *E-mail de contacto: adria.garcia.ma@gmail.com*
- *Mención realizada: Computación.*
- *Trabajo tutorizado por: Ricardo Toledo Morales (Ciencias de la Computación)*
- *Curso 2013/14*

### 1.2 Objetivos del proyecto

Los objetivos principales del proyecto son los siguientes:

- Conseguir que el robot sea capaz de navegar autónomamente por un espacio cerrado determinado sin colisionar con ningún objeto, evitando los posibles obstáculos utilizando un sonar y un láser.
- Programar el control del robot sobre el sistema operativo ROS.

## 2 Estado del Arte

A continuación se explicarán los conceptos teóricos necesarios para entender qué estructura tiene el sistema operativo que controla al robot, así como saber qué es una placa Raspberry Pi, que procesará todo nuestro sistema de control. También se verán algunos de los dispositivos que pueden usarse en el robot.

### 2.1 ROS (Robot Operating System)

ROS [1] es un meta-sistema operativo que ofrece las facilidades que un sistema operativo real proporcionaría, tales como abstracción de hardware, implementación de funcionalidades usadas comúnmente, envio de mensajes entre procesos, etc. ofreciendo para ello diferentes herramientas y librerías.

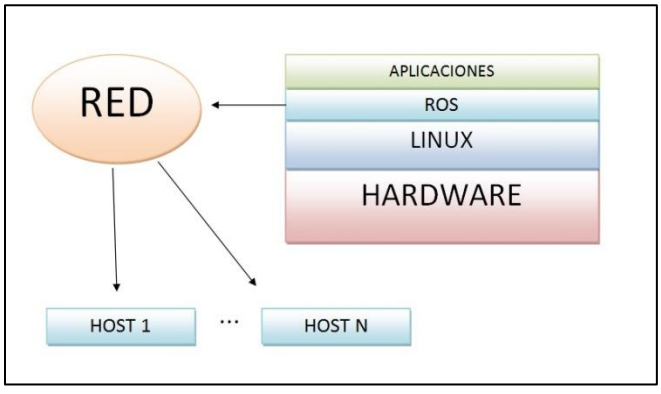


Fig. 2. ROS actúa por encima del sistema operativo como un meta-sistema operativo, nos permite abstracción y además permite la fácil comunicación entre diferentes hosts.

El funcionamiento de ROS es una red de procesos en forma de grafo, donde cada nodo es un proceso y se intercambian mensajes entre ellos utilizando un modelo de subscripción y publicación. Los diferentes elementos del grafo computacional de ROS son:

- Nodos: Cada nodo es un proceso que realiza operaciones computacionales, cada uno asignado al control de las diferentes partes del robot. Así un nodo estaría asignado por ejemplo a un láser, recibiendo sus lecturas, otros nodos podrían controlar las ruedas, algún otro se encargaría de realizar el cálculo y la interpretación de las lecturas, etc.
- "Master": Es el nodo principal y controla el correcto funcionamiento del grafo computacional de ROS, registrando los nombres de los diferentes nodos. Es necesario para que los nodos se encuentren el uno al otro y puedan recibir mensajes. Además también guarda los diferentes parámetros a los que pueden acceder los nodos.
- Mensajes: Los nodos se comunican entre ellos mediante mensajes, que son estructuras de datos simples.
- Tópicos: Un tópico es un nombre que identifica el contenido de un mensaje. Un nodo interesado en un cierto tipo de datos se suscribirá al tópico apropiado asociado a esos datos. Puede haber varios subscriptores y publicadores para un mismo tópico y de la misma manera un nodo puede ser subscriptor y publicador de varios tópicos a la vez.
- Servicios: Debido a que el modelo de subscripción y publicación es útil para la comunicación entre varios nodos, para poder realizar comunicación eficiente de "request/reply" son necesarios los servicios, utilizados para establecer conexiones con dos tipos de mensaje, uno de solicitud (request) y otro de respuesta (reply). Un nodo que ofrece un servicio esperará a recibir una solicitud de un nodo cliente, el cual recibirá una respuesta y podrá utilizar dicho servicio.
- Bags: Es un mecanismo para poder guardar los datos de los mensajes.

El ROS Master contiene todos los datos de los tópicos y de los servicios ofrecidos por los nodos, así como el nombre de éstos. Cuando un nodo y otro requieren una conexión, el ROS Master actúa del mismo modo que un servidor DNS. Cuando un nodo subscrito a un tópico requiere una conexión con otro nodo que publica en él, se establece una conexión, normalmente con TCPROS (utiliza sockets de TCP/IP).

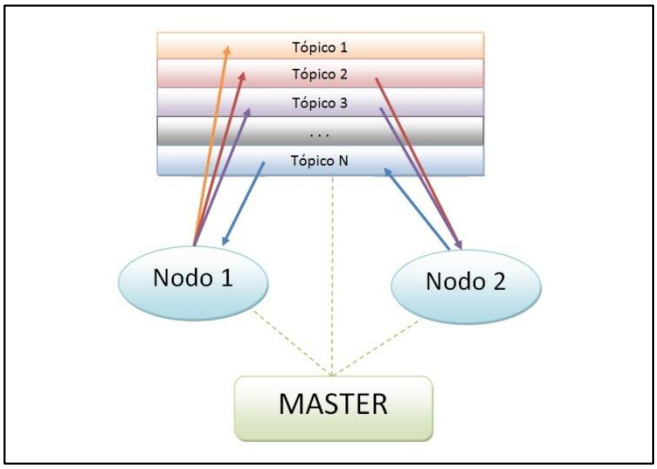


Fig. 3. Esquema de cómo se comunican dos nodos en ROS. El nodo 1 publica al tópico X, mientras que el nodo 2 subscriptor del tópico X recibe los datos publicados por el nodo 1, todo ello gracias al Master. Un mismo nodo puede ser subscriptor o publicador de varios tópicos a la vez.

Algunas de las herramientas más útiles que nos ofrece ROS son las siguientes:

- Rosnode list: nos ofrece una lista de todos los nodos en funcionamiento en el actual grafo computacional, incluyendo el nodo Master.
- Rospack find [20]: nos permite encontrar rápidamente un paquete por su nombre, útil si se dispone de muchos.
- Roscd [20]: nos permite acceder a cualquier directorio dentro del espacio de trabajo actual, indicando el nombre del directorio.
- Roslaunch: permite iniciar diversos nodos con solo un fichero. Útil si se necesitan iniciar varios nodos a la vez para conseguir una funcionalidad completa, se inicializa el launch en lugar de iniciar nodo a nodo.
- Rostopic: nos permite acceder a los diferentes tópicos de ROS indicando el nombre del tópico.
- Rqt_graph: nos muestra por pantalla un grafo con todos los nodos acutales en funcionamiento, así como los tópicos a los que publican y se subscriben, indicando la conexión entre un nodo y otro. Con esta herramienta se puede observar de forma sencilla el grafo computacional en funcionamiento.

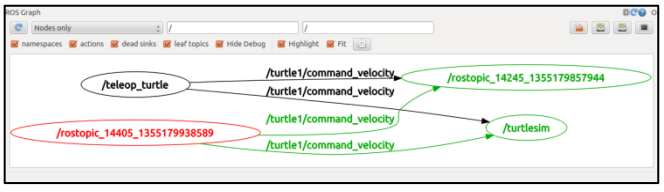


Fig. 4. Ejemplo de grafo mostrado por la herramienta rqt_graph.

Los lenguajes de programación utilizados en ROS son Python y C++ principalmente, aunque en este proyecto se centrará en Python.

## 2.2 Raspberry Pi

Raspberry Pi es un ordenador de placa reducida de bajo coste. Consta de una CPU de 700 MHz y 512 MB de memoria RAM. No dispone de disco duro y en su lugar usa una tarjeta SD para el almacenamiento permanente, además de tener un muy reducido consumo energético (sobre los 3 W). También dispone de conexión Ethernet y soporta sistemas operativos basados en Linux.

Raspberry Pi ha sido creada por la Raspberry Pi Foundation, dedicada a avanzar en la educación en el campo de los ordenadores y la ciencia de la computación.

Se podría decir que Raspberry Pi es prácticamente un ordenador de bajas prestaciones y de coste muy reducido, premitiendo aumentar las prestaciones juntando varias de estas placas (posible gracias a su bajo coste).

Debido a que necesita un tiempo muy alto para compilar ciertos procesos por si sola, la Raspberry Pi nos permite conectar diversos host, que ayudan a esta compilación de forma totalmente transparente, utilizando una "cross compilation", lo que reduce muchísimo el tiempo de compilación mientras esté conectada al host (o hosts). Esta conexión entre diferentes máquinas también ayuda a repartir el cálculo computacional, por lo que se puede conseguir una red de ordenadores trabajando juntos con una mayor potencia y de forma sencilla.

## 2.3 Sonar

El sonar es un sensor necesario para llevar a cabo el proyecto y conseguir el objetivo propuesto. Este componente lanza ondas sónicas y espera recibir el eco de éstas. Una vez recibido obtiene el tiempo de respuesta y puede así calcular el espacio entre el sonar y el obstáculo. Por lo tanto, los datos que transfiere este dispositivo son de distancias.

Actualmente existen diversos modelos de sonares en el mercado. Algunos ejemplos de ellos son el SRF02 [9], el SRF05 [8] y el SRF08 [7]. El primero es el más pequeño y de menor coste, aunque solo dispone de un sensor a diferencia de los otros dos, que disponen de dos sensores. El SRF05 al disponer de dos sensores es más preciso, y además el coste no se eleva demasiado. Por último, el SRF08 es el más preciso de los tres, además de contar también con un sensor de luz, pudiendo enviar así datos sobre la iluminación. Los precios de estos dispositivos oscilan entre los 12 y los 30 euros, por lo que son costes asumibles.

## 2.4 Láser

Otro sensor de distancia necesario para la realización del proyecto es el láser. Parecido al sonar, este dispositivo envía señales de láser y espera recibir el reflejo. A partir de aquí calcula la distancia a la que se encuentra el obstáculo y la transmite. Aunque pueda parecer que la utilización de láser y sonar es redundante, no es así: son dispositivos complementarios. Por una parte el láser es mucho mas preciso que el sonar y tiene un rango mas amplio de observación. Por otro lado, el sonar es capaz de detectar objetos que el láser no podría: este último no puede detectar cristales debido a que la luz lo atraviesa y no rebota; por tanto las ondas sónicas del sonar sí lo detectan al ser un objeto sólido.

El coste de este componente es el más elevado. Existen láseres como el láser Parallax [10], cuyo precio es de 100 dólares, el láser S300 de Sick [11] o el láser HOKUYO [12], rondando un precio de unos 1500 euros. Más adelante se explica qué modelo se ha elegido y por qué.

### 2.5 Motores

El sistema locomotor del robot que le proporciona la movilidad está compuesta por dos partes: los motores de las ruedas y la placa controladora de estos motores. Las ruedas están controladas por los motores, que hacen que éstas giren según un cierto corriente eléctrico que circula en uno u otro sentido del motor. Este circuito está controlado por una placa, mediante la cual se pueden dar órdenes de movilidad a dos motores.

En el mercado podemos ver motores como los EMG30 [13] o los EMG49 [14], los segundos evolución de los primeros. Además, para controlar estos motores existen placas como la MD25 [15] o la MD49 [16]. La primera se utiliza para el control de los motores EMG30, mientras que la segunda se usa para controlar los EMG49. Tanto los motores como la placa se pueden encontrar en el mercado en forma de pack, contando con los motores EMG30 y la placa MD25 por 120 euros, o por el doble de precio la placa MD49 con los motores EMG49. Más adelante se darán las especificaciones más técnicas.

### 2.6 Bus de comunicaciones

Para conectar todos los dispositivos a la Raspberry Pi y que se puedan comunicar entre ellos es necesario que se disponga de un bus de comunicaciones en el robot. Este componente hará que se pueda acceder fácilmente a los dispositivos.

Los buses más utilizados actualmente son el bus I2C [17], desarrollado por Philips, y el bus CAN [18], el bus que más se utiliza en los vehículos.

## 3 Desarrollo del Proyecto

Como se ha comentado anteriormente, este proyecto consiste en programar un robot que utilice ROS y que sea capaz de navegar autónomamente sin colisionar con los objetos del entorno. Para ello será necesario desarrollar los nodos de control de cada elemento así como de juntarlos en el grafo computacional que nos proporciona el sistema operativo utilizado. El lenguaje que se utilizará para desarrollar estos nodos será Python. La versión de ROS utilizada será la versión HYDRO.

A continuación se explicarán las características del robot y la metodología utilizada para el desarrollo del proyecto, así como los resultados obtenidos.

### 3.1 Características del robot

Como se ha dicho, el robot está compuesto por diversos componentes y dispositivos. De los diferentes tipos de dispositivos hemos escogido uno dentro de las posibilidades comentadas. Los componentes y dispositivos del robot son los siguientes:

- Meta-sistema operativo ROS.
- Raspberry Pi con un host.
- Sonar SRF08.
- Láser HOKUYO
- Motores EMG30 y placa MD25.
- Bus I2C.

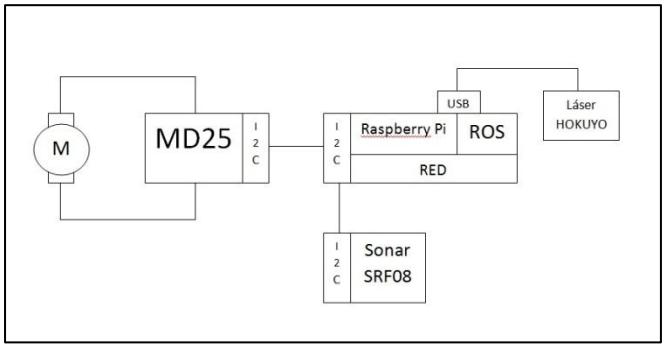


Fig. 5. Esquema del robot con los diferentes componentes y dispositivos.

En los siguientes apartados se detallarán las características de cada uno y su influencia dentro del robot.

### 3.2 ROS en el robot

Como se ha hecho referencia anteriormente, la versión utilizada de ROS es la HYDRO. Debido a que tenemos una Raspberry Pi y un host, ROS debe estar instalado en ambos dispositivos. De esta manera, al comunicarse se comportará como si solo se tratara de un único núcleo de cálculo.

ROS nos permite la abstracción del harware y nos proporciona herramientas permitiendo trabajar a muy alto nivel, como se ha comentado en el apartado 2.1.

### 3.3 Raspberry Pi y host

Nuestro robot cuenta con una Raspberry Pi, que necesita un voltaje de 5V de alimentación para funcionar [19], y un host para poder compilar rápidamente utilizando "cross compilation" en el caso de la instalación de ROS, ya que usaremos Python, que no requiere compilación.

La Raspberry Pi cuenta con dos puertos USB, un puerto Ethernet, por lo que en los USB dispondremos de teclado y ratón para poder trabajar con la Raspberry mientras tengamos al robot sin dependencia, conectado al host mediante un cable Ethernet. Más adelante, cuando el robot se comunique con el host mediante WiFi, uno de los puertos USB será utilizado para conectar el láser y otro para la targeta WiFi.

Por otro lado, el host del que disponemos se trata de un ordenador con procesador 8 cores y cuenta con 16 GB de memoria RAM, suficiente para llevar a cabo todos los procesos necesarios para la ejecución de los nodos del grafo computacional de ROS de nuestro robot.

Ambos dispositivos tienen Linux como sistema operativo, sobre el que va instalado ROS.

### 3.4 Sonar SRF08

El sonar escogido para ser usado en el robot es el sonar SRF08. Necesita una alimentación de 5V y se conecta mediante un bus I2C a la Raspberry Pi.

Cuenta con 36 registros en los cuales se puede escribir o leer de ellos. El primer registro sirve para escribir en él y mandar comandos al sonar. El segundo registro indica la lectura del sensor de luz. Y por último, los 34 registros restantes indican la lectura de cada uno de los 17 ecos que envia el sonar. Los ecos tienen un tamaño de 2 Bytes, por lo que un eco queda guardado en dos registros, de un Byte cada uno.

**TABLA 1 - Registros SRF08**

| Registro | Lectura | Escritura |
|---|---|---|
| 0 | Revisión del Software | Registro de comandos |
| 1 | Sensor de luz | Registro de máxima ganancia |
| 2 | Primer Byte de Eco 1 | Registro de rango |
| 3 | Segundo Byte de Eco 1 | |
| ~~~~ | ~~~~ | ~~~~ |
| 34 | Primer Byte de Eco 17 | |
| 35 | Segundo Byte de Eco 17 | |

El orden de los ecos va determinado por defecto por la distancia de éstos: el primer eco corresponde a la menor distancia detectada dentro del rango del sonar, mientras que el último corresponde a la más lejana.

Además existen diversos comandos que podemos enviar al sonar a través del primer registro. Estos comandos, que pueden ser enviados en decimal o en hexadecimal, sirven principalmente para cambiar la unidad en la que mide la distancia el sonar (pulgadas, centímetros o microsegundos). También podemos cambiar el modo de lectura del sonar de manera que ordene los ecos leídos de manera diferente, obteniendo una secuencia de lecturas según la posición, de izquierda a derecha del rango del sonar, que puede ser usada en redes ANN.

### 3.5 Láser HOKUYO

El láser que se usará en este proyecto será un láser HOKUYO, concretamente el modelo URG-04LX. Se necesita un voltaje de 5V de alimantación, igual que el sonar SRF08. Su ángulo de visión es de 240º.

La gran ventaja de este láser es que en ROS ya existe un nodo que gestiona el laser HOKUYO (hokuyo_node). De esta manera, este nodo realiza todos los cálculos necesarios y publica la lectura del láser en el tópico "/LaserScan" para ser posteriormente utilizada por cualquier otro nodo.

### 3.6 Motores EMG30 y placa MD25

Los motores elegidos para el robot son los EMG30 junto con la placa MD25 que los controla y son necesarios 12V de alimentación. Así la placa nos permite tratar los motores a un más alto nivel mediante registros y comandos transmitidos por el bus I2C con el que se conectará a la Raspberry Pi.

| Register | Name | Read/Write | Description |
|---|---|---|---|
| 0 | Speed1 | R/W | Motor1 speed (mode 0,1) or speed (mode 2,3) |
| 1 | Speed2/Turn | R/W | Motor2 speed (mode 0,1) or turn (mode 2,3) |
| 2 | Enc1a | Read only | Encoder 1 position, 1st byte (highest), capture count when read |
| 3 | Enc1b | Read only | Encoder 1 position, 2nd byte |
| 4 | Enc1c | Read only | Encoder 1 position, 3rd byte |
| 5 | Enc1d | Read only | Encoder 1 position, 4th (lowest byte) |
| 6 | Enc2a | Read only | Encoder 2 position, 1st byte (highest), capture count when read |
| 7 | Enc2b | Read only | Encoder 2 position, 2nd byte |
| 8 | Enc2c | Read only | Encoder 2 position, 3rd byte |
| 9 | Enc2d | Read only | Encoder 2 position, 4th byte (lowest byte) |
| 10 | Battery volts | Read only | The supply battery voltage |
| 11 | Motor 1 current | Read only | The current through motor 1 |
| 12 | Motor 2 current | Read only | The current through motor 2 |
| 13 | Software Revision | Read only | Software Revision Number |
| 14 | Acceleration rate | R/W | Optional Acceleration register |
| 15 | Mode | R/W | Mode of operation (see below) |
| 16 | Command | Write only | Used for reset of encoder counts and module address changes |

Fig. 6. Registros de la placa MD25 y su descripción.

Existen 17 registros para controlar la placa MD25. Los dos primeros sirven para aplicar movimiento a los motores, ya sea ajustando la velocidad o el giro que se desea, dependiendo del modo. Si el modo es 0 o 1 el primer registro sirve para la velocidad de un motor y el segundo para la velocidad del otro motor, mientras que si es 2 o 3 el primer registro sirve para la velocidad y el segundo para el giro. Este modo puede ser cambiado mediante otro de los registros de la placa. Además de la diferencia de los modos que se ha mencionado anteriormente tenemos, dependiendo del modo también cambia el rango de las velocidades a aplicar:

- Modo 0 y 2: velocidades entre 0 y 255 (0 hacia atrás, 128 parado y 255 hacia adelante).
- Modo 1 y 3: velocidades entre -128 y 127 (-128 hacia atrás, 0 parado y 127 hacia alante)

También podemos encontrar 8 registros (encoders) dedicados a guardar el movimiento de las ruedas (4 registros para cada motor). Estos datos se calculan mediante el giro de las ruedas e indican cuanta distancia se ha movido ésta, muy útil para calcular en un módulo a parte la odometría del robot (posición y movimiento).

Además también existe el registro para enviar comandos a través del bus I2C. Como en el caso del sonar, estos comandos pueden ser en formato decimal o hexadecimal, y se pueden usar, por ejemplo, para resetear los valores de los encoders a 0.

Por otra parte, los motores EMG30 son accionados a bajo nivel por el sentido de la corriente eléctrica. El sentido de giro de las ruedas viene dado por el sentido en el que circula la corriente por el motor. Para ello, la placa MD25

controla transistores a forma de interruptor para poder cambiar el sentido de la corriente.

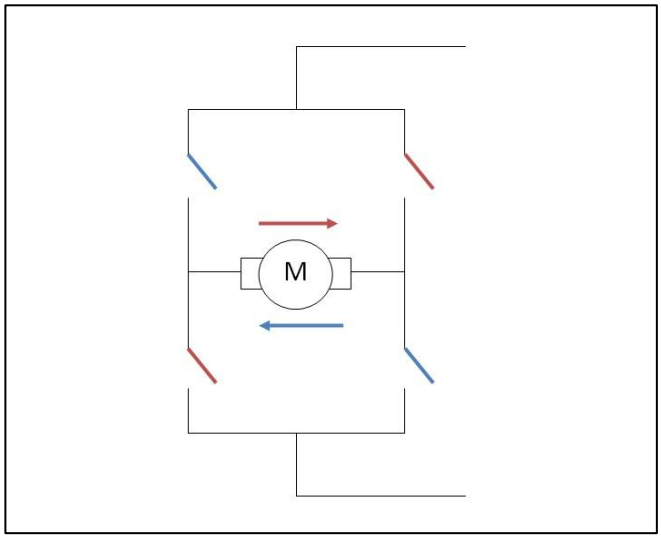


Fig. 7. Funcionamiento a bajo nivel de los motores EMG30. Cuando los interruptores de color azul están cerrados el sentido de la corriente es el contrario al de la generada cuando están cerrados los rojos.

### 3.7 Bus I2C

Debido a que la mayoría de dispositivos seleccionados trabajan con el bus I2C es éste el tipo de bus de comunicaciones que tendrá instalado el robot. Este bus nos permitirá añadir una capa de abstracción entre el hardware y el sistema operativo, facilitando el acceso a todos los dispositivos conectados a él. El acceso mediante I2C se realiza con comandos de lectura y escitura como si se tratara de un fichero, donde cada dispositivo está asociado a una dirección de memoria.

### 3.8 Metodología

La metodología utilizada para este proyecto es la de cascada o "Waterfall", marcando unos determinados hitos o "milestones", los cuales se deberán cumplir realizando las tareas necesarias para conseguirlos. De esta manera, para cada hito existirá un proceso previo de aprendizaje (requerido para llevar a cabo la tarea), la fase de implementar el código necesario para conseguir el objetivo marcado y la prueba final para confirmar que se ha logrado correctamente y se ha alcanzado el hito correspondiente.

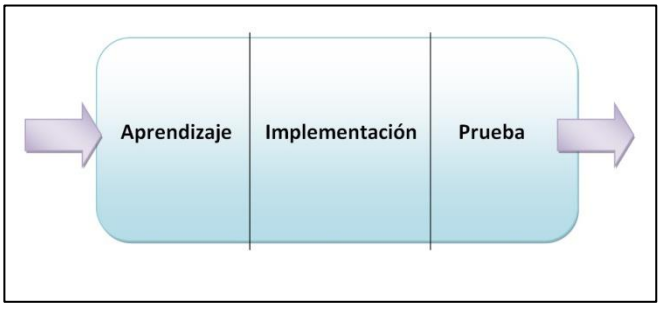


Fig. 8. Composición de una tarea, primero un proceso de aprendizaje, después la implementación y finalmente la prueba.

Durante todo el proceso se llevará a cabo la anotación y documentación del proyecto, manteniendo así un registro de todo el trabajo realizado. Con esta información se llevará a cabo un control de las fechas para saber la duración de cada tarea y poder comprobar si se cumplen los plazos correctos para llegar al objetivo final.

Las tareas o hitos que se realizarán en este proyecto serán las siguientes:

1. Realizar la instalación del sistema tanto en la Raspberry Pi como en la máquina donde se llevarán a cabo las simulaciones.

2. Realizar los tutoriales de ROS disponibles en la wiki para asimilar los conceptos de programación para ROS.

3. Conseguir mover las ruedas del robot mediante un nodo de control y conseguir leer la odometría en un nodo a parte.

4. Instalar el sonar y recibir las lecturas de éste en un nodo.

5. Acoplar el movimiento de las ruedas según la lectura del sonar.

6. Instalar el láser y conseguir recibir las lecturas de éste en un nodo diferente para su tratamiento.

7. Acoplar el movimiento de las ruedas según la lectura del láser para hacer que el robot no choque con obstáculos.

8. Investigar posibles usos de esta tecnología y posibles ampliaciones del proyecto.

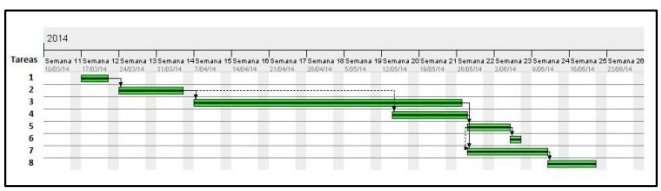


Fig. 9. Diagrama de Gantt de las tareas realizadas en el proyecto

Dado que las tareas 1 y 2 tratan de seguir manuales de instalacion y tutoriales respectivamente, el proceso de aprendizaje de estas tareas se realiza mientras éstas se llevan a cabo. En el proceso de prueba se comprueba que se han realizado correctamente si el sistema operativo funciona bien y si todos los ejemplos generados de la tarea 2 dan los resultados esperados por el tutorial.

Para las siguientes tareas de programación de nodos en ROS, parte del proceso de aprendizaje queda implícito en la realización de la tarea 2.

Además, para las tareas 3 y 4 la fase de aprendizaje consta

de la conexión, habilitación y utilización del bus i2c, tanto escribir como leer de él. Posteriormente este aprendizaje difiere en los registros y parámetros utilizados por las placas md25 de los motores y el sonar SRF08. A continuación se lleva a cabo la implementación del código de las tareas y se comprueba que funciona correctamente.

Por otra parte, una vez realizadas estas tareas es necesario entender los datos leídos como parte inicial antes de implementar el código de la tarea 5 que realice el movimiento de las ruedas según la lectura del sonar.

La tarea 6 es similar a la tarea 4, ya que tanto el sonar como el láser son componentes parecidos. La única variación son las conexiones y los parámetros utilizados por cada uno. De la misma manera, la tarea 7 es complementaria a la tarea 5, dado que las ruedas se comportarán de manera muy similar según las lecturas recibidas.

Por último la tarea 8 es de investigación en su totalidad.

Una vez realizada todas las tareas propuestas en la metodología a utilizar, se habrá llegado al objetivo propuesto inicialmente: el robot será capaz de moverse autónomamente y aleatoriamente por un espacio determinado sin colisionar con los objetos del entorno.

### 3.9 Resultados

El resultado obtenido en este proyecto es el robot con la Raspberry Pi como dispositivo de computación donde van conectadas las placas md25 que controlan los motres y el sonar, ambos mediante el bus i2c, y el laser.

Además se han obtenido los nodos necesarios para el movimiento de las ruedas, la lectura de la odometría y la lectura del sonar y del láser, así como el nodo de control sobre la velocidad de las ruedas según las lecturas tanto del láser como del sonar. Dado que la estructura del sistema operativo ROS tiene forma de grafo, como se había estudiado anteriormente, cada nodo corresponde a un archivo escrito en Python que realiza una función correcta dentro del sistema. Estos archivos son:

- SonarPub.py: Nodo publicador del tópico "sonar", que publica las lecturas realizadas por el sonar leyéndolas del bus i2c al tópico mencionado. Este nodo debe estar ubicado en el robot, debido a que necesita acceder al sonar.

- SonarSub.py: Nodo subscriptor del tópico "sonar", encargado de recibir los datos del sonar y mostrarlos en la pantalla. Este nodo puede estar en cualquier host, ya que no requiere de elementos de hardware específicos.

- OdomPub.py: Nodo publicador del tópico "odometry", que se encarga de publicar las lecturas de los odómetros de las ruedas al tópico anterior. Las lecturas las obtiene del bus i2c. Este nodo necesariamente debe estar en el robot para poder acceder a los encoders de los motores.

- OdomSub.py: Nodo subscriptor del tópico "odometry", que recibe los datos del movimiento de las ruedas y las muestra por pantalla. Este nodo puede estar situado en cualquier host.

- LaserSub.py: Nodo subscriptor del tópico "laser" que recibe los datos del láser y los muestra por pantalla. Puede estar ubicado en cualquier host al no necesitar hardware específico.

- WheelSub.py: Nodo subscriptor del tópico "movement", que recibe los datos de velocidad del robot y transforma la velocidad linear y angular de los ejes x, y, z en la velocidad apropiada para cada rueda. Esta velocidad se le aplica a los motores mediante el bus i2c. Es imprescindible que esté situado en el robot para poder acceder al bus I2C y dar órdenes de movimiento a las ruedas.

- ControlRobot.py: Nodo subscriptor de los tópicos "sonar" y "laser" y a la vez publicador de "movement". Se encarga de enviar señales de velocidad al tópico "movement" para las ruedas a partir de las señales recibidas de los tópicos "sonar" y "laser". Este nodo puede estar tanto en el robot como en un host externo, ya que no necesita acceder directamente a los dispositivos, sino que obtiene los datos de la subscripción a los tópicos. Preferiblemente debería estar en un host externo para aligerar la carga de cálculo de la Raspberry.

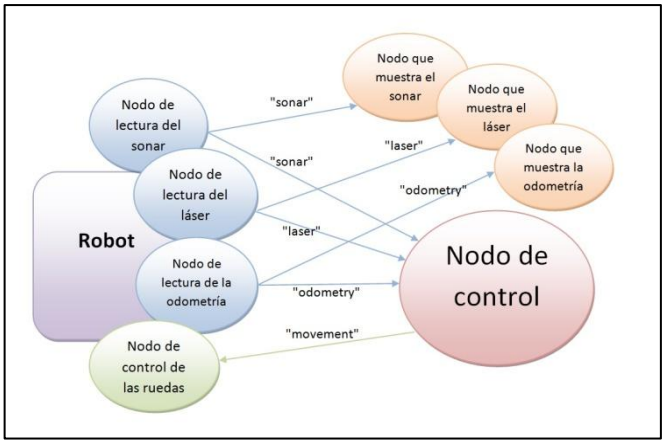


Fig. 10. Esquema del grafo computacional del robot del resultado. En azul los nodos de lectura de laser, sonar y odometría. En rojo el nodo de control del robot. En verde el nodo que ordena el movimiento de las ruedas. En naranja los nodos que muestran por pantalla.

El funcionamiento del grafo de control es el siguiente: los nodos publicadores del láser y el sonar publican a través de sus tópicos. A partir de aquí el nodo de control recibe estas lecturas, realiza los cálculos necesarios y publica una señal de velocidad al tópico de movimiento, que será

recibido por el nodo de accionamiento de los motores de las ruedas y aplicará el movimiento al robot. De esta manera el robot puede moverse autónomamente dependiendo de las lecturas del sonar y el láser, por lo que no colisionará con ningún objeto del entorno. Con esto conseguimos alcanzar el objetivo inicialmente planteado.

## 4 CONCLUSIÓN

Mi aportación a este proyecto ha sido la de crear los nodos correspondientes para obtener un grafo de control con el fin de conseguir el objetivo propuesto inicialmente: hacer que el robot navegue por un espacio determinado evitando colisionar con los diferentes objetos que pueda haber en el entorno.

Una vez finalizado, se ha obtenido un robot simple de movimiento autónomo y aleatorio, con un láser y un sonar. Los componentes seleccionados han sido éstos debido a que para detectar los objetos del entorno son complementarios (el láser es más preciso mientras que el sonar detecta objetos que el láser no puede, como cristales).

Pero sobretodo, más importante que lo que se ha hecho es el cómo se ha hecho: la estructura en forma de grafo de ROS. Esta estructura nos permite añadir nodos fácilmente a un grafo sin modificar nada de los nodos ya implementados, lo que supone que podemos crear tantos nodos como queramos y añadir diferentes componentes a estos nodos. Esto se traduce a que podemos añadir componentes y aumentar la funcionalidad de nuestro robot base de manera fácil y rápida.

De esta manera, el proyecto puede ser ampliado por muchas vertientes. Si se quiere mejorar el movimiento del robot de forma que no sea aleatorio puede investigarse el uso de SLAM, con nodos que generen el mapa y se encarguen de la navegación del robot sobre él. O si por otro lado se le quiere añadir una cámara que nos muestre el recorrido del robot sería tan fácil como conectarla en la Raspberry Pi y realizar el nodo que trate las imágenes de la cámara. Además éstas imágenes pueden ser tratadas por procesos de visión por computador en otros nodos, por lo que aún se aumentan más las posibilidades (por ejemplo para robots de seguridad). Incluso se pueden añadir electrodomésticos caseros: con una aspiradora sería una versión más sofisticada del robot aspiradora roomba, por lo que también se puede destinar el robot al uso doméstico. Como vemos, a partir de esta base y de cómo está hecha se abre un abanico de posibilidades muy amplio para extender este proyecto.

Además este proyecto me ha servido para aprender los conceptos básicos del movimiento de un robot y cómo funciona el sistema operativo ROS, muy útil para toda clase de robots.

## AGRADECIMIENTOS

Me gustaría agradecer a mi tutor Ricardo Toledo, a Joan Oliver y a Jorge Ramírez por toda la ayuda prestada durante el proyecto, por el robot y los materiales prestados y por el espacio del que he dispuesto para trabajar.

## BIBLIOGRAFÍA

[1] "ROS: Getting Started". Internet: http://wiki.ros.org/ROS/StartGuide, [Marzo 12, 2014]

[2] "ROS: Tutorials". Internet: http://wiki.ros.org/ROS/Tutorials, [Abril 23, 2014]

[3] "Stanford Artificial Intelligence Robot Web". Internet: stair.stanford.edu, [Marzo 12, 2014]

[4] Morgan Quigley, Eric Berger, Andrew Y. Ng, "STAIR: Hardware and Software Architecture", Internet: http://robotics.cs.brown.edu/aaai07/materials/stanford_paper.pdf, [Marzo 13, 2014]

[5] Raspberry Pi Foundation, "Raspberry Pi Documentation". Internet: http://www.raspberrypi.org/documentation, [Junio 21, 2014]

[6] Robot Electronics, "MD25", Internet: http://www.robot-electronics.co.uk/htm/md25ser.htm, [Abril 23, 2014]

[7] Robot Electronics, "SRF08 Technical Specification", Internet: http://www.robot-electronics.co.uk/htm/srf08tech.shtml, [Junio 20, 2014]

[8] Robot Electronics, "SRF05 Technical Specification", Internet: http://www.robot-electronics.co.uk/htm/srf05tech.htm [Junio 20, 2014]

[9] Robot Electronics, "SRF02 Technical Specification", Internet: http://www.robot-electronics.co.uk/htm/srf02tech.htm [Junio 20, 2014]

[10] Parallax Inc, "Laser Range Finder", Internet: http://www.parallax.com/product/28044, [Junio 20, 2014]

[11] Sick Sensor Intelligence, "S300 Laser Scanner", Internet: http://www.sick.com/group/EN/home/products/product_portfolio/optoelectronic_protective_devices/Pages/safetylaserscanners_S300.aspx, [Junio 20, 2014]

[12] HOKUYO, "URG-04LX", Internet: http://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx.html [Junio 20, 2014]

[13] Robot Electronics, "EMG30 Specification", Internet: http://www.robot-electronics.co.uk/htm/emg30.htm, [Junio 20, 2014]

[14] Robot Electronics, "EMG49 Specification", Internet: http://www.robot-electronics.co.uk/htm/emg49.htm [Junio 20, 2014]

[15] Robot Electronics, "MD25 Technical Specification", Internet: http://www.robot-electronics.co.uk/htm/md25tech.htm [Junio 20, 2014]

[16] Robot Electronics, "MD49 Technical Specification", Internet: http://www.robot-electronics.co.uk/htm/md49tech.htm, [Junio 20, 2014]

[17] I2C, "I2C Bus", Internet: http://www.i2c-bus.org/ [Junio 20, 2014]

[18] CIA, "Controller Area Network", Internet: http://www.can-cia.org/index.php?id=can [Junio 20, 2014]

[19] Raspberry Pi Foundation, "Raspberry Pi Hardware Specifications", Internet: http://www.raspberrypi.org/documentation/hardware/rasp

berrypi/README.md, [Junio 21, 2014]
[20] "ROS: Navigating the ROS Filesystem", Internet: http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem, [Junio 21, 2014]