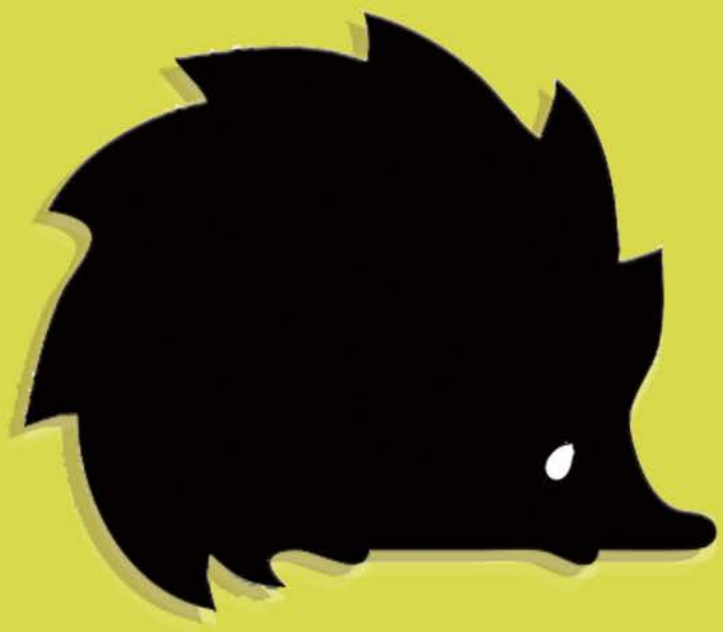


OURSIN

ÍNDICE

1.- Introducción.

Oursin es una empresa que nace con el deseo y la ilusión de tender un puente entre los juegos infantiles clásicos y los niños invidentes, de manera que estos últimos no solo puedan disfrutar, sino también aprender y desarrollar capacidades sobre su entorno. Es por esto, que la empresa ha creado Rayo McBlind, el primer coche teledirigido para niños invidentes.

Rayo McBlind está provisto de una serie de sensores que permiten detectar objetos cercanos al vehículo. De este modo, en caso de que surja una posibilidad inminente de colisión, el coche frenará y emitirá un sonido que se transmitirá por Bluetooth y que el usuario recibirá en sus auriculares. El sonido recibido es un sonido en 8D, lo que quiere decir que, la persona que lo escucha, es capaz de ubicar el sonido en el espacio, asemejándose al sonido natural: si vamos por la calle podemos distinguir si pasa un coche a nuestra izquierda o nuestra derecha por el sonido, sin embargo, si escuchamos una canción, no se produce esta posibilidad. Con el sonido recibido, el usuario podrá ubicar el coche en el espacio, de modo que podrá ir rectificando su trayectoria hasta llegar a un punto final. Por esta necesidad de ubicar el coche en el espacio, Rayo McBlind no se trata solo de un coche teledirigido, es también una herramienta para desarrollar la inteligencia espacial de quien haga uso de él.

El coche se dirige mediante un mando, como el de PlayStation, conectado por Bluetooth a la Raspberry Pi, que es el microcontrolador integrado en el dispositivo. Este microcontrolador ha sido programado con el lenguaje Python y desarrollado en Ros.

Por otro lado, cabe destacar, que la empresa Oursin surge en un contexto universitario y su creación ha sido promovida por la asignatura Electrónica de Consumo del cuarto curso del Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación de la Escuela de Ingenieros de Telecomunicación (ETSIT) de la Universidad Politécnica de Madrid (UPM).

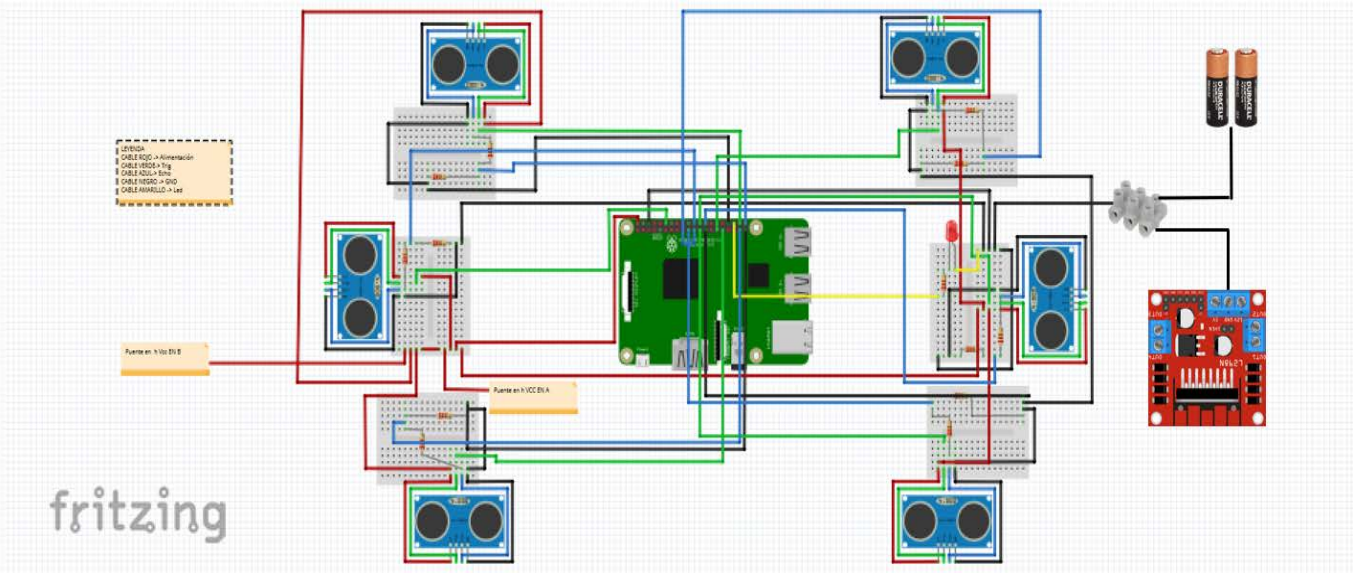
Se procede en los siguientes apartados, a detallar el funcionamiento, creación y dinámica del producto.

2.- Prototipo Hardware.

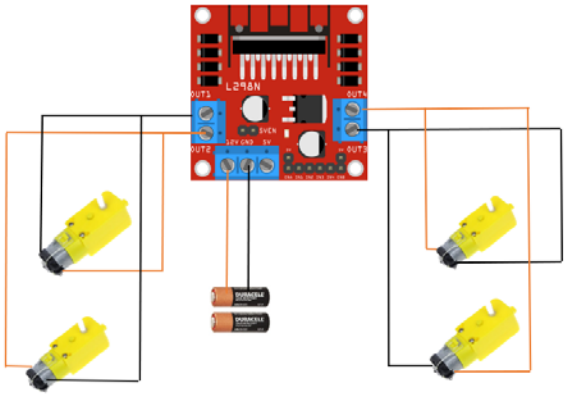
2.1.- Esquema eléctrico.

Para el prototipo hemos usado una Raspberry que hemos alimentado a través de una batería externa con una autonomía de 12 horas. También tiene incorporado 6 sensores de movimiento, 3 que colocaremos en la parte delantera y otros 3 que colocaremos en la parte trasera del coche. Como la Raspberry trabaja a 3.3 V y los sensores de movimiento a 5 V, hemos creado un divisor de tension a traves de 2 resistencias, de 1 y 2 k Ω.

Podemos ver un esquema general creado a través de fritzing en la siguiente imagen:



Una parte muy importante del coche es la intervención del puente en h. Este será el encargado de gestionar la velocidad y el movimiento de los motores. Estará conectado a los cuatro motores del coche y además a cuatro pilas de 1,5 voltios cada una.

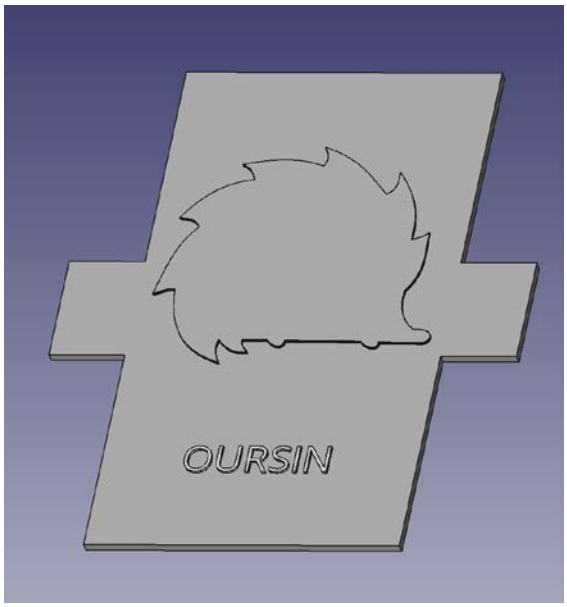


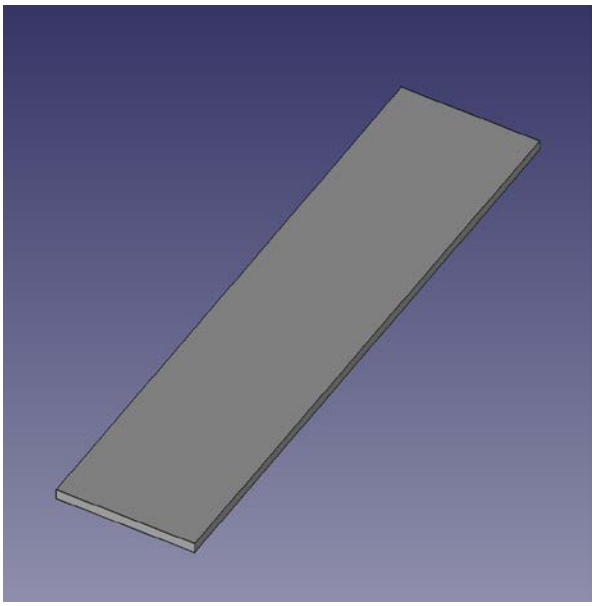
2.2.- Piezas y medidas

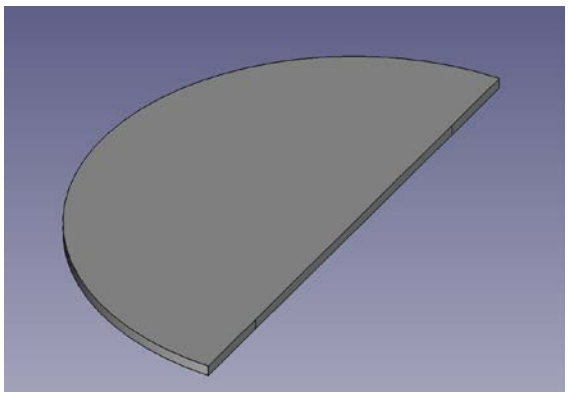
El coche consta de cuatro motores, con sus cuatro ruedas y dos placas transparentes que conforman la estructura de este. Podemos ver en el gráfico siguiente algunas de sus medidas, así como las de la caja donde irán las pilas que alimentan al puente en h:

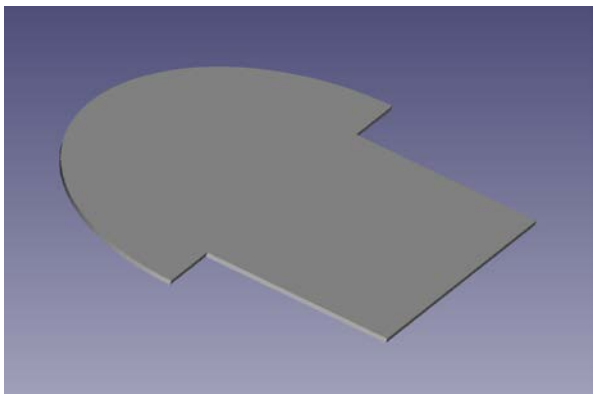
Para evitar daños en el coche hemos diseñado una carcasa parachoques. Sirve para que, si fallase algún sensor, en vez de chocar el coche con la pared, con los posibles daños que esto puede ocasionar, impacte con esta carcasa.

Para el diseño hemos utilizado la aplicación de Freecad y posteriormente lo hemos impreso con una impresora 3D. Como el diseño era muy grande para imprimirlo de una vez, lo hemos dividido en cuatro partes como las que tenemos a continuación:

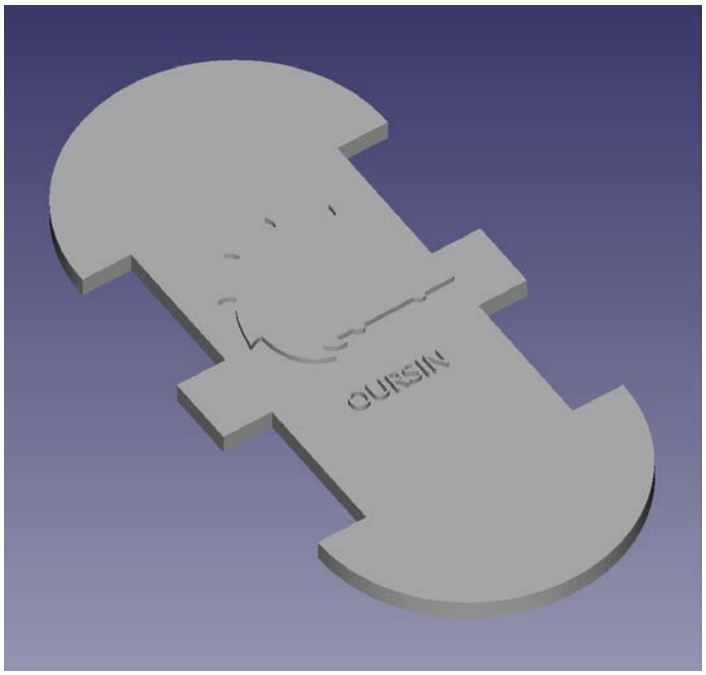








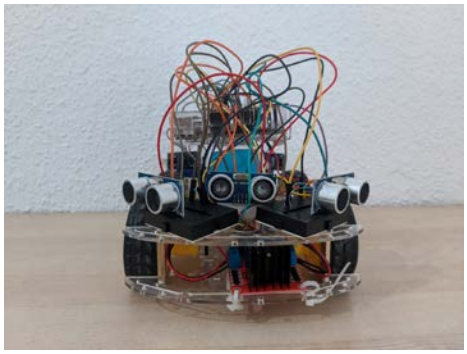
Posteriormente las hemos pegado, quedando el siguiente resultado:

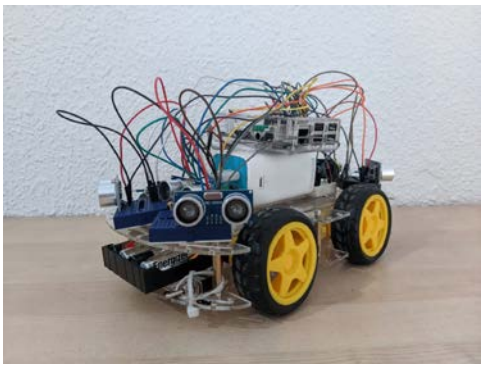


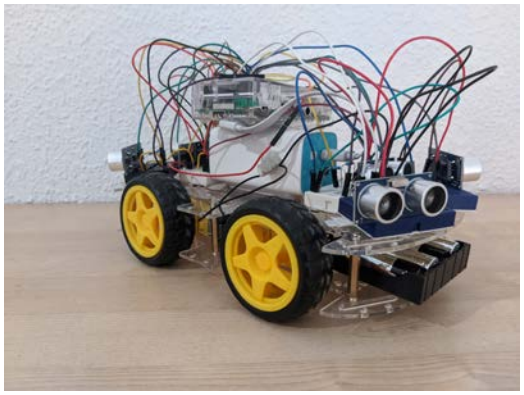
La carcasa tiene un grosor de 5mm, la parte de los semicírculos es más larga que el coche, así impactarían estas partes con la pared, y, como se ve en la imagen, en el medio tiene el logo y nombre de nuestra empresa.

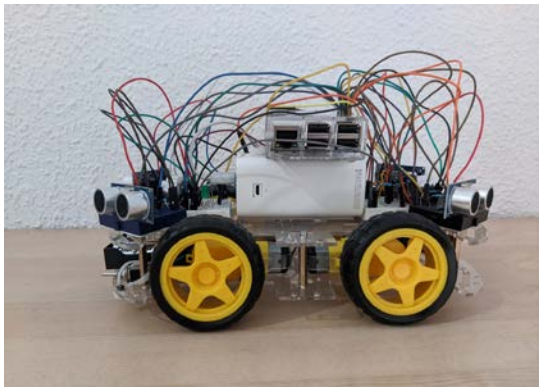
Una vez montado, el prototipo tendría el siguiente aspecto:

SIN CARCASA

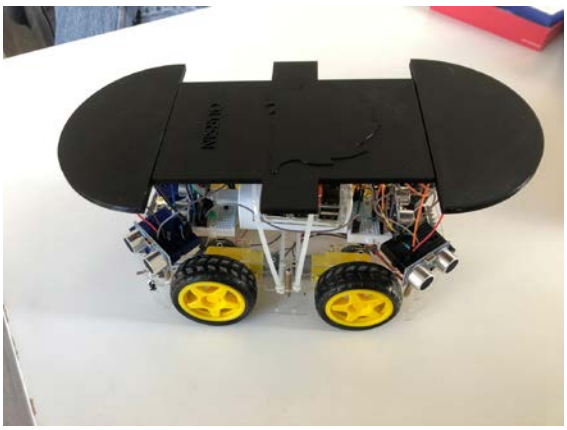


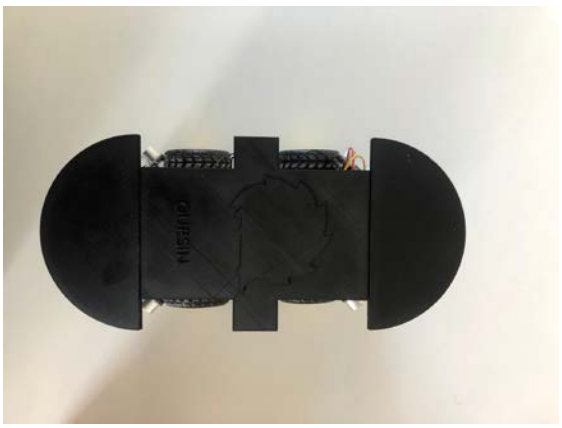






CON CARCASA





3.- Prototipo Software.

3.1.- Implementación.

3.1.1.- ROS Noetic.

Para todo el funcionamiento del coche hemos utilizado como base software ROS, para aprender su sistemática y funcionamiento decidimos realizar un curso de introducción a ROS desde la plataforma online Udemy.

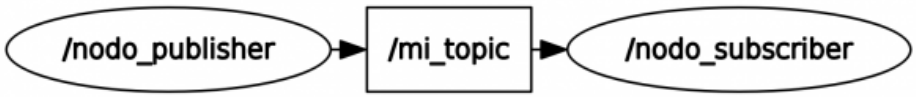
Robot Operating System (ROS) es un entorno de código libre utilizado para el desarrollo de robots. No es un sistema operativo, pero posibilita la comunicación con el hardware como hace un sistema operativo. Con ello, nos dota de la capacidad de comunicar diferentes procesos entre sí mediante el paso de mensajes.

Además, una de las principales ventajas de ROS frente a otros programas para el desarrollo de software, es que, provee un sistema de gestión de paquetes llamado catkin. Este sistema, permite la creación de paquetes en un mismo espacio de trabajo y dispone de herramientas de compilación de paquetes.

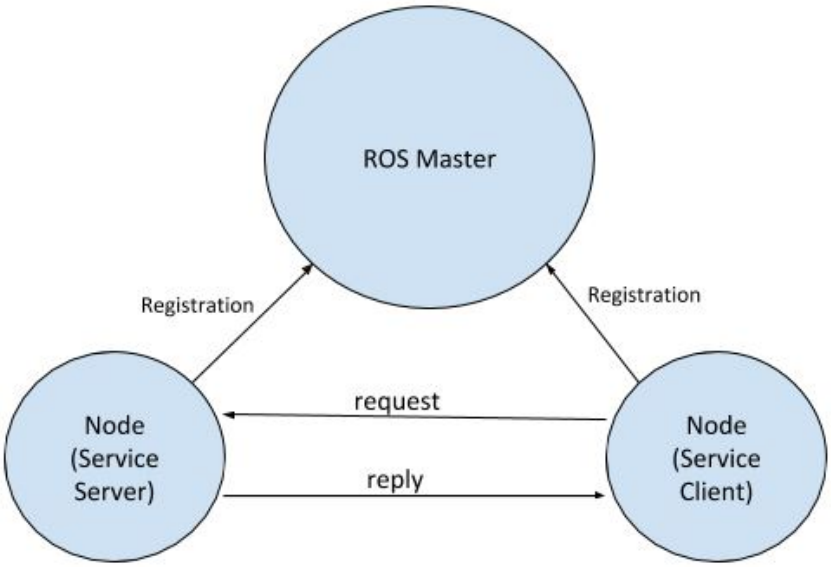
ROS está basado en una arquitectura de nodos. Los nodos se comunican entre sí y comparten datos entre ellos, a través de los denominados “topics”. Cada nodo es responsable de una pequeña porción específica de la funcionalidad completa del robot.

Cada nodo, puede ser “publisher” o “subscriber”. Un nodo “publisher” publica la información correspondiente a su funcionalidad en un “topic”. El nodo “subscriber” recibe dicha información.

Para clarificar el concepto, el funcionamiento es como una pequeña tubería, por la que circula la información.



Además, también tenemos la posibilidad de crear “services”, con sus correspondientes “topics”, “clients” y “servers”. Sin embargo, esta funcionalidad no ha sido necesaria para la implementación de nuestro coche, dado que los services se utilizan principalmente para procesos remotos que deben ejecutar una acción rápidamente, como por ejemplo consultar el estado de un nodo, apagar el sistema o realizar cálculos rápidos

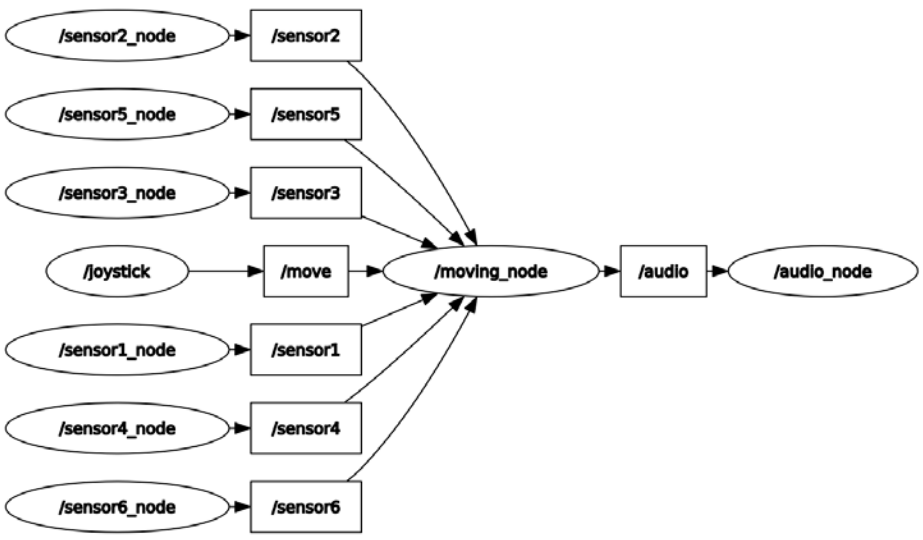


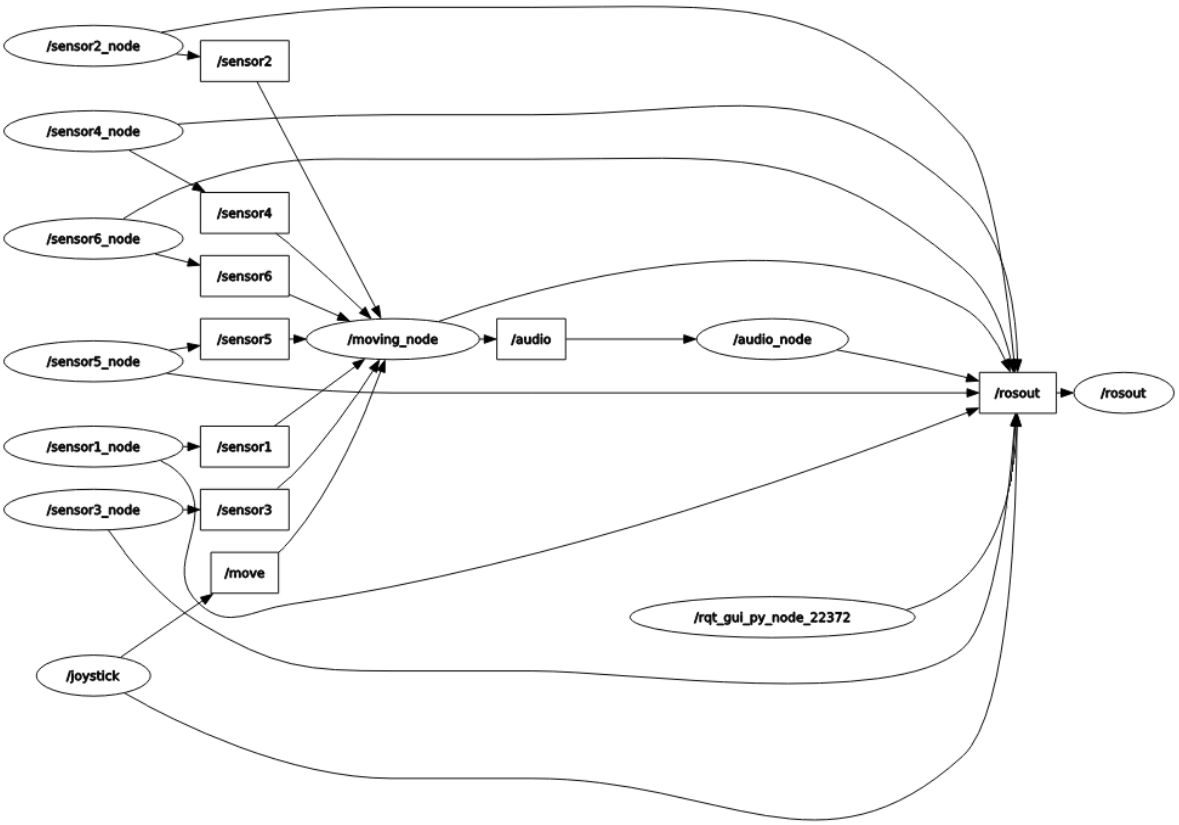
3.1.2.- Programación en ROS.

Para la implementación hemos utilizado 9 programas, cada uno es un nodo, y como ROS permite la programación en distintos lenguajes para cada uno de los nodos, hemos empleado:

- 8 programas escritos en Python: sensor1.py, sensor2.py, sensor3.py, sensor4.py, sensor5.py, sensor6.py, audio.py, joystick.py.
- 1 programa escrito en C++: moving.cpp

Estos son los grafos finales del programa, donde los rectángulos representan los “topics” y los círculos los nodos, así como las flechas representan el intercambio de la información tanto de un “publisher” al “topic” (la flecha sale desde el publisher), como de un “topic” al “subscriber ” (la flecha llega al subscriber):





Se han utilizado 9 nodos:

- 6 nodos de sensores.
- 1 nodo para el joystick .
- 1 nodo para el movimiento.
- 1 nodo para el audio.

Los sensores publican continuamente la distancia a los obstáculos en su topic “sensor” correspondiente. A su vez, el nodo joystick detecta la pulsación de cada tecla, activa el flag correspondiente a esa tecla y lo publica en el topic “move”.

El nodo “moving” se suscribe a todos los nodos anteriores, por lo que maneja toda la información procedente del joystick para iniciar el movimiento mediante el callback correspondiente, y la información de los sensores para detener el coche y activar el flag que indica que se ha superado la distancia mínima permitida cuando ocurra este evento.

Esta información es, a su vez, publicada en el topic “audio”.

Por último, el nodo audio, se suscribe al topic audio e inicia la reproducción del mismo por los auriculares.

Para cada nodo, hemos escrito un programa distinto, siendo el programa ”moving” en lenguaje C++ y los 8 restantes en Python.

Como añadido, también hemos creado un programa íntegramente en lenguaje C, para utilizar en el entorno Eclipse como alternativa al ROS.

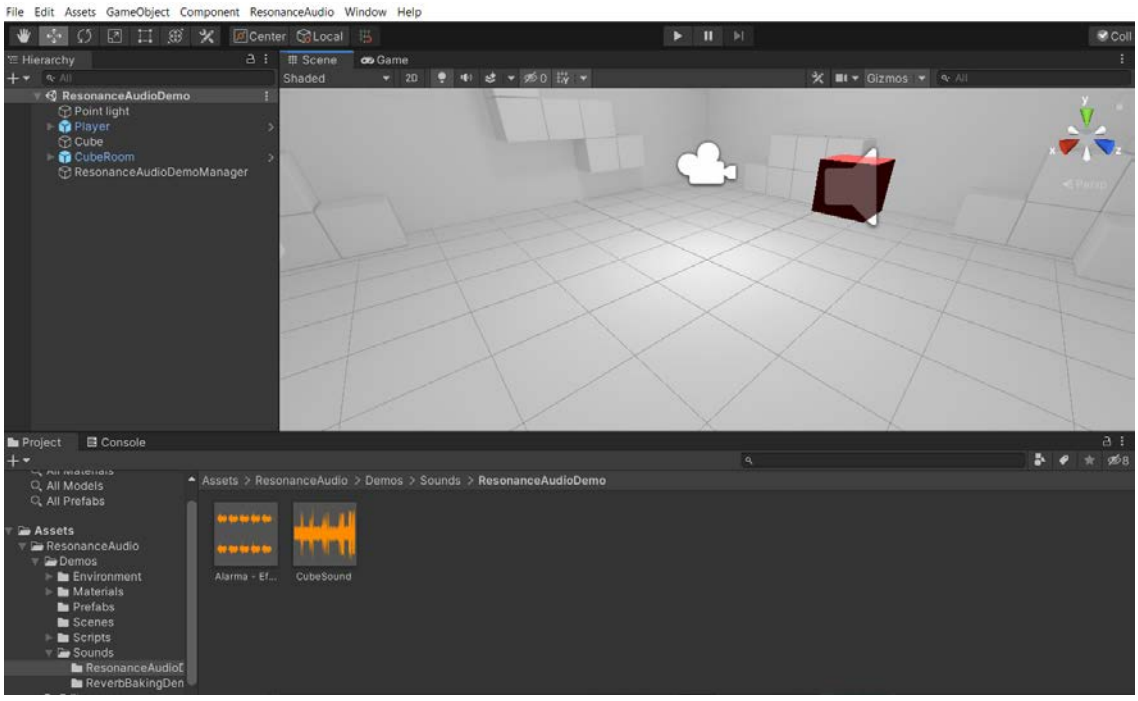
Todo el código implementado se encuentra publicado en el apartado de Anexos.

3.2.- Resonance Audio.

Uno de los objetivos de nuestro proyecto era que, a partir de la detección de obstáculos por parte del coche, el usuario fuera capaz de saber en qué momento y lugar del espacio se producía dicha situación. Para ello, utilizamos un proyecto de software denominado Resonance Audio. El proyecto se trataba de un código escrito en lenguaje C que permitía mediante ciertas fórmulas matemáticas conseguir un mapeo del sonido por el cual, el usuario sería capaz distinguir, por medio de unos auriculares, de qué lugar del espacio procedía el sonido del posible obstáculo.

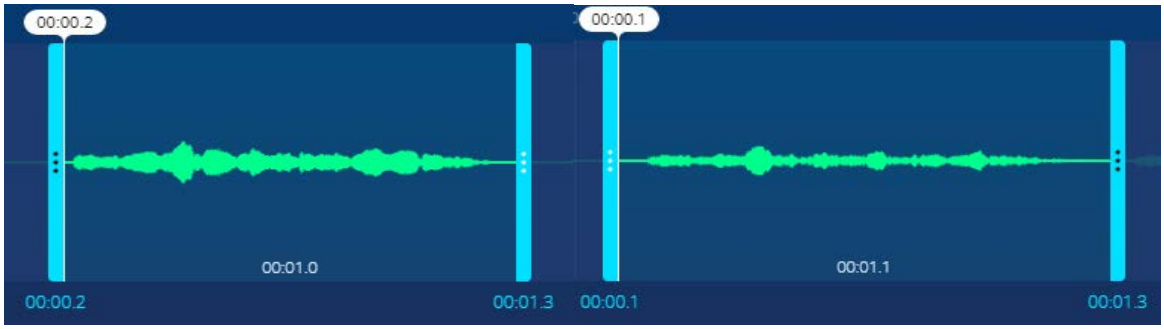
Una vez entendido el software, adquirimos el programa Unity. La plataforma básica de **Unity** es un centro creativo para artistas, diseñadores y programadores. Permite editar e iterar rápidamente tus ciclos de desarrollo con vistas previas de tu trabajo en tiempo real. Puedes crear escenas 2D o 3D, animaciones o cinemáticas directamente en **Unity Editor**.

Por ello, utilizamos Unity para crear una escena 3d donde importamos un objeto. Se trataba de un **cubo** fijo en el espacio que emitía un **sonido constante**. A través de este objeto, su emisión de sonido y la herramienta de movimiento de Unity, pudimos movernos en el espacio para elaborar las pistas de audio y así incorporarlas al coche.



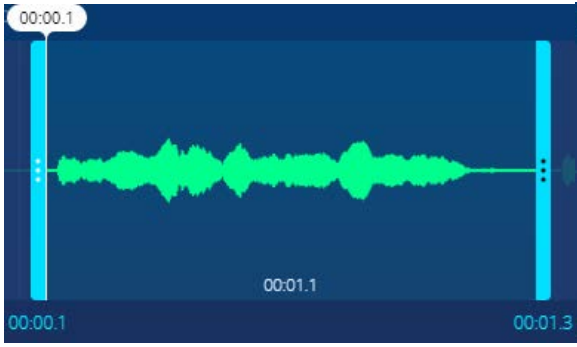
Con esto, elaboramos pistas para apreciar un obstáculo que se presenta delante, uno que está por delante a la derecha, otro delante a la izquierda, también uno que se aproxime por atrás, atrás derecha y atrás izquierda. Haciendo un total de 6 pistas de audio que serán incorporadas al software de cada uno de los sensores.

Podemos observar en las siguientes gráficas el resultado obtenido de las 6 pistas de audio:

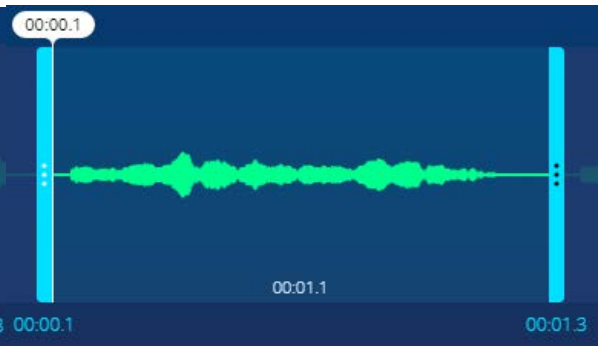


Delante

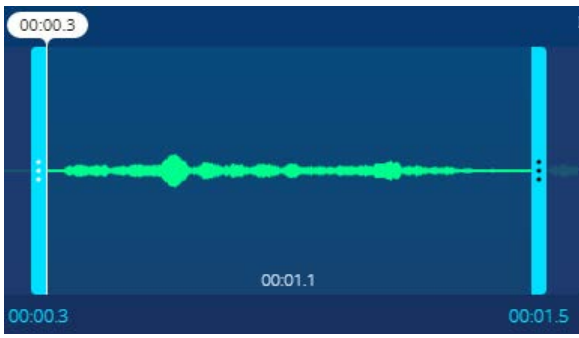
Delante derecha



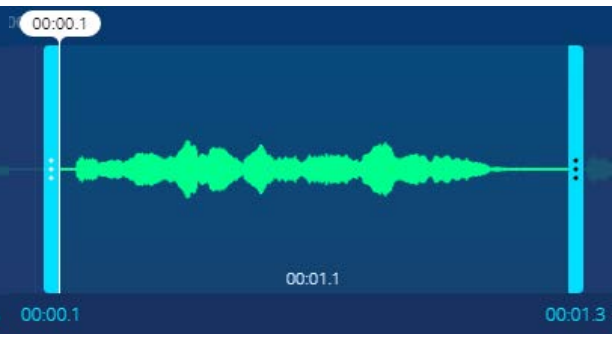
Delante izquierda



Detrás



Detrás derecha



Detrás izquierda

4.- Finanzas.

A continuación, se procede a detallar el coste que ha tenido la realización del prototipo así como una posterior deducción de su precio final de venta al público.

La siguiente tabla detalla el coste por unidad de cada uno de los componentes de Rayo McBlind.

Componente	Precio (ud)	Origen
Raspberry	28,90 €	Amazon
Sensor	1,99 €	Amazon
Motor+chasis	7,01 €	AliExpress
MicroSD	2,84 €	AliExpress
Puente en H	5,90 €	Amazon
Alimentación	4,99 €	Amazon
Mini Protoboard	0,32 €	AliExpress
Embalaje	3,00 €	Amazon

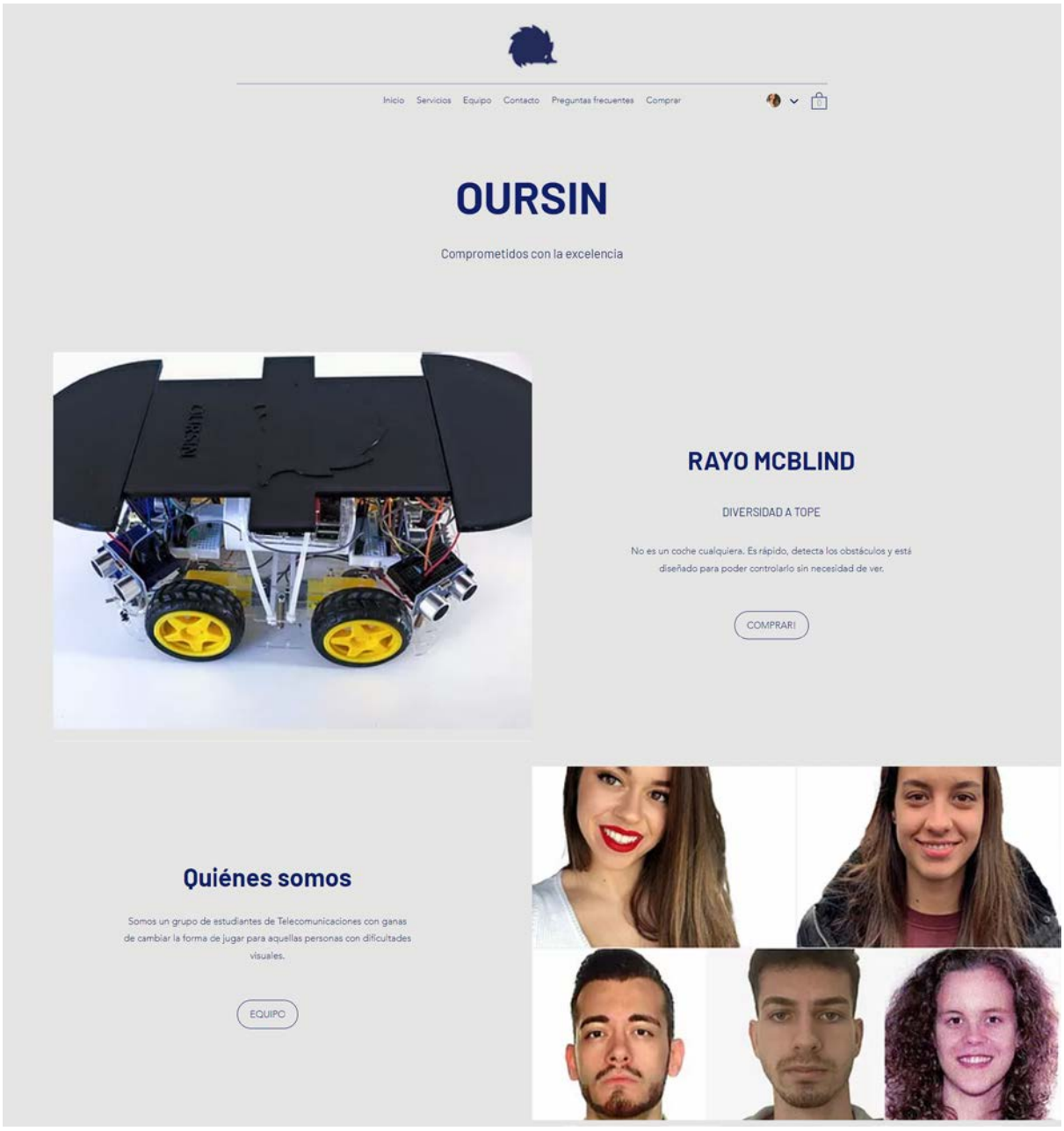
Con estos precios, el coste total de producción de una unidad sería de 66,5 €. Sin embargo, se ha hecho un cálculo estimado de cómo quedaría este precio con una producción masiva, pues se podrán abaratar costes al comprar muchas unidades y se podrán reemplazar algunos componentes por otros más baratos. Según esto, el coste de producción podría rebajarse en un 40%, quedando reducido a 39,90 €.

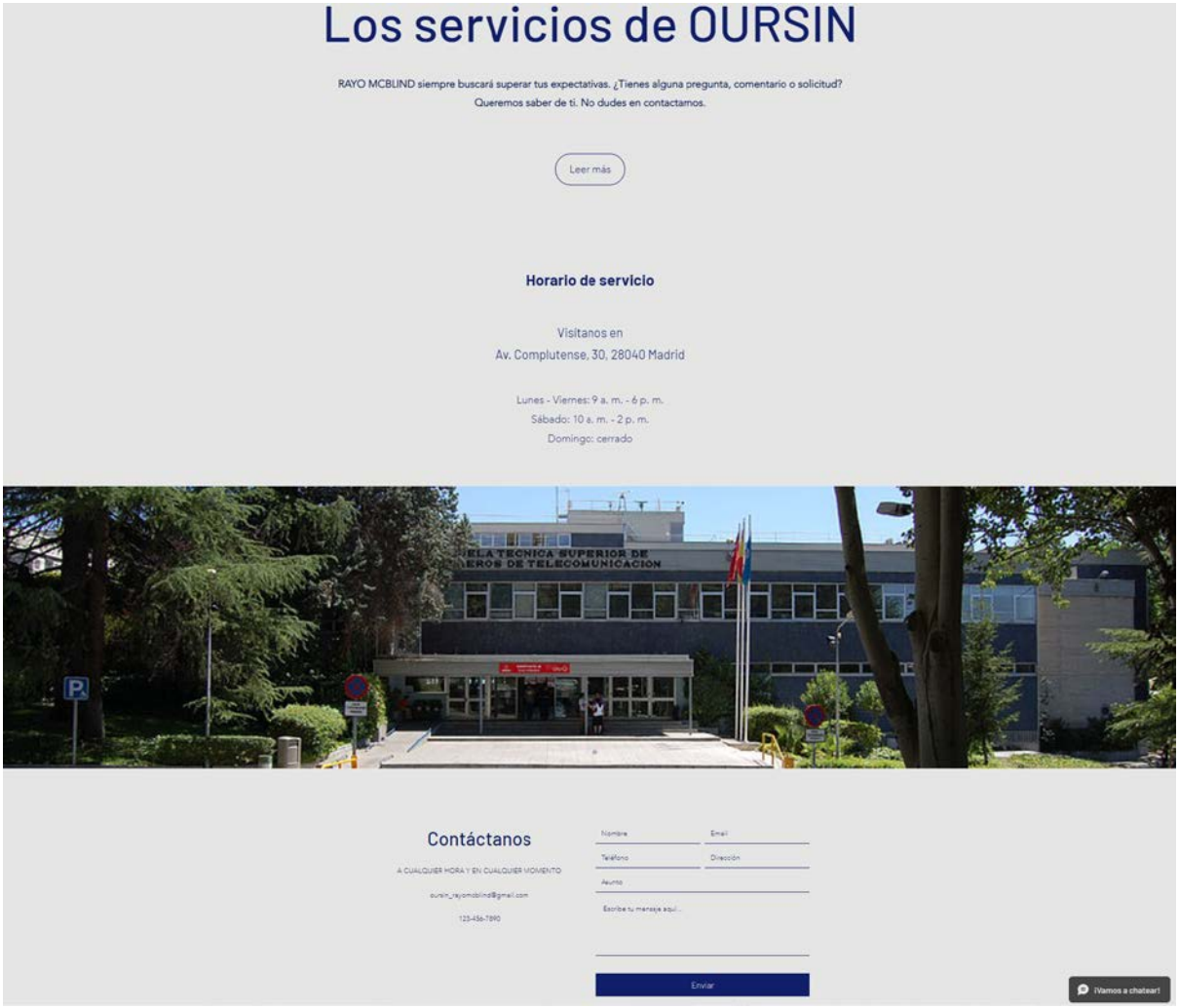
Haciendo un breve estudio de mercado, se analiza que el precio de un coche teledirigido suele tener un rango de variación entre cuarenta y noventa euros. Como la empresa pretende acercar el producto como un juguete más a las personas invidentes, quiere que el precio sea equitativo respecto a los coches teledirigidos convencionales. Es por esto, que se ha decidido que el precio final de venta del producto sea de 79,80€, obteniendo entonces un 50% de beneficios para la empresa.

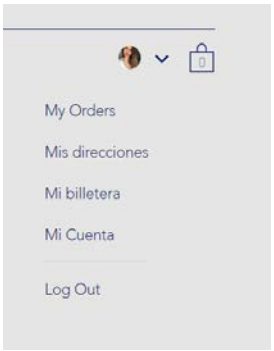
5.- Marketing y venta.

5.1.- Página web

Para la venta y publicidad de nuestro prototipo hemos creado una página web en la que encontraremos todo lo necesario para saber más de Rayo McBlind y poder comprarlo.
En la página principal, nada más entrar en la web, podremos ver un breve resumen de esta. Qué vendemos, quiénes somos, servicios que ofrecemos, e incluso cómo contactarnos.Contamos además con un menú con el que podremos navegar a través de las distintas partes de la página web.







Podremos loguearnos en la página haciendo click en "log in" (parte superior derecha de la página principal) y de esta manera tener acceso a mis pedidos, mis direcciones de envió, mi billetera y mi cuenta.

Si continuamos explorando la web y clicamos sobre "servicios" (situado en el menú superior), podremos ver un breve resumen de nuestro servicio al cliente.

Llamada de atención al cliente

Queremos que todos nuestros clientes puedan experimentar el nivel de profesionalismo de nuestro equipo al colaborar con Coche. La razón de ser de nuestros servicios, especialmente de este, es mejorar tu vida. Tenemos los mejores productos y nuestra atención al cliente es incomparable.

Entrega programada

Elige dónde y cuándo quieres recibir tu paquete. Llegaremos justo a tiempo para entregarte el pedido.

Consultas sobre compras

La mayoría de nuestros clientes utiliza este servicio, que contribuye a nuestro éxito. Con nuestros servicios buscamos cubrir todas tus necesidades. Nos enorgullece nuestro servicio al cliente y sabemos que nuestro equipo te atenderá con eficiencia. ¿Qué podemos hacer por ti?

¿Tienes preguntas sobre nuestros servicios? Contáctanos con confianza.

Contáctanos

Si el usuario tiene alguna duda puede clicar el boton de “Contáctanos” o en Contacto dentro del menú superior. Enviarnos un correo electrónico o consultar el horario de visitas.

Si todavía no hemos resuelto sus dudas, o estas son muy generales, puede consultar el apartado de “Preguntas frecuentes” (situado en el menú superior) que suelen hacernos los usuarios.

Algunas de ellas las podemos ver en la siguiente imagen:

¿Cómo llegan los envíos de RAYO MCBLIND?

Los envíos correrán a cargo de OURSIN, pero se realizarán por medio de empresas de logística privada.

¿Cuánto tarda en llegar el paquete?

Ofrecemos varias posibilidades.

- A un día hábil después de la compra (+5 €).
- A dos o tres días hábiles después de la compra (+3 €).
- A tres o cinco días hábiles después de la compra (gratuito).

¿Qué pasa si no estoy en casa para recibir el paquete?

Si no hay nadie en la dirección para recibir tu entrega, OURSINLogistics dejará un aviso de «No hemos podido encontrarte». Haremos tres intentos de entrega en días consecutivos.

¿Cuánto tarda en reflejarse un compra con tarjeta de crédito?

Las compras en las Tarjetas de Crédito se descuentan inmediatamente del cupo disponible, sin embargo, se debe esperar para que la transacción se vea reflejada en sus movimientos. Si realizó una compra nacional, esta se refleja en 1 o 2 días hábiles después de efectuar la transacción.

¿Cómo realizo la devolución de mi paquete por correo?

Es muy sencillo, simplemente tienes que volver a empaquetar tu articulo, de manera que no pueda dañarse en el transporte. Puede ser en la misma caja en la que te llegó o en otra. Descarga el pdf de devolución, imprímelo y pégalo en el paquete. Por ultimo llévalo a correos.

¿Hay algún coste al realizar una devolución por correo?

No, es totalmente gratuito.

¿Cuánto tarda en reflejarse la devolución por correo en mi cuenta?

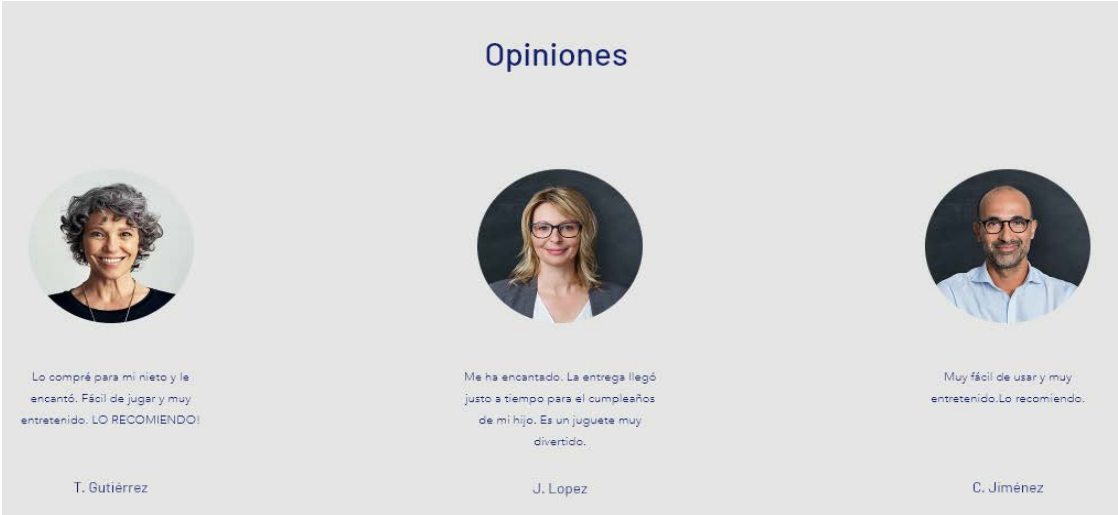
Para las devoluciones de los productos enviados por OURSIN, recibirás un reembolso en un plazo máximo de 14 días desde la fecha en la que enviaste la devolución. Podrás verlo en tu extracto bancario en un plazo máximo de 5-7 días laborables desde la emisión del reembolso.

¿Cómo rastrear el envió de mi pedido?

Cundo tu pedido salga de nuestros almacenes, recibirás un correo con los datos de la compra y el numero de seguimiento del pedido con el que podrás ver por donde va.

Podremos ver el precio, características y diseño final, además de poder comprar el producto clicando en “Comprar” (en el menú superior):

Podremos ver comentario de otros usuarios que ya compraron el producto, así como agregar nosotros mismos uno después de realizar la compra:



El enlace a la página es el siguiente:

https://mariaciffu.wixsite.com/website

5.2.- Empaquetado

Para el envío de nuestros pedidos hemos diseñado una caja. Se trata de una caja de cartón que tiene en dos de sus caras el nombre del producto, Rayo McBlind, y en las otras dos el logo de la empresa con el nombre y además el nombre en relieve, para que las personas con discapacidad visual lo puedan leer (braille).

Las medidas de la caja son de 35x35x20 y para que el producto no se deteriore en el transporte va rellena de una estructura de poliespan.

En el siguiente enlace se puede ver el prototipo de la caja en 3D:
https://packlane.com/share/8d6670bf-6e81-4ba6-a607-31a4ce290a1f/embed

6.- Conclusiones.

Tras la revisión del prototipo presentado, se cree que hay diversas líneas de mejora de cara a un prototipo final que se enumeran a continuación.

1. Mejorar la carcasa del vehículo de modo que sea más compacto y permita un mejor agarre por parte del usuario.
2. Abaratamiento y reemplazo de algunos componentes que encarecen el producto, como puede ser la sustitución de la Raspberry Pi o el diseño propio del chasis del coche para no tener que comprarlo a empresas externas.
3. Desarrollar un mando propio que se pueda ofrecer dentro de la venta para que el usuario no tenga que disponer de un mando de PlayStation.
4. Mejora del producto basándonos en testeos de personas invidentes.
5. Dar a conocer el producto en fundaciones como la ONCE o en colegios especializados en este ámbito.

Con todas las líneas de mejora puestas en la mesa, se cree que Rayo McBlind es un producto de futuro que, más allá de producir beneficios económicos a la empresa, puede hacer una gran labor social y humanitaria, convirtiéndose en un proyecto ilusionante y de futuro.

7.- Bibliografía.

1.- Funcionamiento automático con detección de obstáculos

https://solectroshop.blogspot.com/p/funcionamiento-automatico-con-deteccion.html

2.- Manejo de un robot por infrarrojo

https://solectroshop.blogspot.com/p/manejo-del-robot-por-infrarrojo.html

3.- Arduino + Raspberry Pi - RaspDuino

https://geekytheory.com/arduino-raspberry-pi-raspduino

4.- Arduino IDE on Raspberry Pi Zero W

https://www.raspberrypi.org/forums/viewtopic.php?t=251194

5.- How to use Bluetooth Controllers with Python on Raspberry Pi

https://www.youtube.com/watch?v=F5-dV6ULeg8

6.- Bluetooth Enabled Joystick Controller

https://www.instructables.com/Bluetooth-Enabled-Joystick-Controller/

7.- Bluetooth Joystick Pan/Tilt Controller

https://www.instructables.com/DIY-Pan-Tilt-ProjectAndroid-and-Arduino/

8.- Joystick + 10x LED iluminado 5V Botones pulsadores

https://www.amazon.es/EG-STARTS-iluminado-pulsadores-Raspberry/dp/B075DFNK24/ref=sr_1_16?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=2Q1Q2W8OZXHV2&dchild=1&keywords=joystick+raspberry+pi&qid=1615546894&sprefix=joystick+ras%2Caps%2C183&sr=8-16

9.- Adding Audio Output to the Raspberry Pi Zero

(Son lo mismo)

https://www.raspberrypi.org/blog/tinkernut-diy-pi-zero-audio/

https://www.youtube.com/watch?v=3pXB90IDNoY

10.- Raspberry Pi Zero, audio output via I2S (para un altavoz)

http://www.lucadentella.it/2017/04/26/raspberry-pi-zero-audio-output-via-i2s/

11.- Raspberry Pi and Ultrasonic HC-SR04 distance sensor (Mirar funcionamiento en 1.-)

https://www.youtube.com/watch?v=L90WS-ptnvI

12.- HC-SR04 distance sensor (enlace de compra)

https://www.amazon.es/dp/B06W9JD4X2/ref=sspa_dk_detail_0?psc=1&pd_rd_i=B06W9JD4X2&pd_rd_w=3IMCb&pf_rd_p=c4c9c78d-928d-4cf2-ba41-9873ab24035f&pd_rd_wg=IKdC9&pf_rd_r=5ZRTXW2M7R6ABHXMRASW&pd_rd_r=91152893-9d89-4b62-b805-8bfa47ff59cb&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzQk41QzAwODE1V1BJJmVuY3J5cHRlZElkPUExMDIwNTM5MTcwMVBMR1VGOVpZNCZlbmNyeXB0ZWRBZElkPUEwOTg0MDEzM0pXSlJSNkk2NDM5MyZ3aWRnZXROYW1lPXNwX2RldGFpbCZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=

Mejor opción: 5 sensores 7,99€

https://www.amazon.es/ARCELI-transductor-medici%C3%B3n-Distancia-ultras%C3%B3nico/dp/B07MPZR59P/ref=pd_sbs_2?pd_rd_w=An6lo&pf_rd_p=c289ec33-e7ac-4087-b63e-bf352f1cf145&pf_rd_r=33Y29CN58AN7JXJ28C80&pd_rd_r=736dae42-510d-4d63-894f-02395beb7516&pd_rd_wg=RZGOc&pd_rd_i=B07MPZR59P&psc=1

13.- Receptor Bluetooth 5.0 Jack 3.5mm

https://www.amazon.es/Bluetooth-AGPTEK-Inal%C3%A1mbrico-Dispositivos-Conectados/dp/B07QJWVY84/ref=sr_1_5?dchild=1&keywords=Adaptador+Bluetooth+Jack&qid=1615821131&sr=8-5

14.- Crear aplicación para coche a control remoto

https://www.youtube.com/watch?v=MiWLzbpfqr0&list=PLnwu2s7SIakT3OW0kOQHMuyJYm9aorwc4&index=2

15.- Using a joystick on Raspberry Pi (No inalámbrico)

https://tutorials-raspberrypi.com/raspberry-pi-joystick-with-mcp3008/

16.- DIY Wireless Joystick

https://www.instructables.com/DIY-Wireless-Joystick-Wireless-Gaming/

17.- Spatialize Monoaural Audio into 5.1 Channel using Matlab

https://es.mathworks.com/help/supportpkg/raspberrypi/ref/spatialize-monaural-audio-into-5-1-channel-surround-sound-using-matlab-function-block-in-simulink.html

18.- Online MP3 Cutter

https://mp3cut.net/es/

19.- Playing audio on Raspberry Pi

https://raspberry-projects.com/pi/programming-in-c/audio/playing-audio

https://www.makeuseof.com/tag/play-mp3-audio-raspberry-pi/

20.- Web audio spatialization basics

https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Web_audio_spatialization_basics

21.- Robot evasor de obstáculos Arduino (3 sensores). Posible buena opción incorporar 4 sensores para controlar la posición del coche (derecha, izquierda, delante y atrás).

https://www.taloselectronics.com/blogs/tutoriales/robot-evasor-de-obstaculos-para-arduino-codigo

22.- HC-SR04 Sensor on the Raspberry Pi

https://thepihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

24.- ELCOche (manuales)

https://github.com/ELCOche/ELCOche/tree/master/Manuals

25.- Vídeo Mando PS4 Raspberry 4 controlador con código y Bluetooth

https://youtu.be/pRZIiqL7ZnU

26.- How to Connect a PS3 Remote to the New Raspberry Pi Zero W

https://www.piborg.org/blog/pi-zero-11:39

wifi-ps3

27.- Cómo montar un robot controlado por radiofrecuencia (Joystick)

https://leantec.es/robot-4wd-controlado-por-radiofrecuencia/

28.- Mando PS4 conectado por Bluetooth (Raspberry Pi y Python)

https://www.youtube.com/watch?v=CeyGP3_kKZI

https://github.com/ArturSpirin/pyPS4Controller

29.- Bluetooth Controlled Raspberry Pi Rover

https://www.youtube.com/watch?v=Ro4wRQGtnBw

30.- Video explicativo de música en 8D

https://www.youtube.com/watch?v=e6Ekz7ZDV-w

31.- Enlace para la descarga del programa Unity

https://resonance-audio.github.io/resonance-audio/develop/unity/getting-started

32.- Sensores

https://www.amazon.es/dp/B06W9JD4X2/ref=sspa_dk_detail_0?psc=1&pd_rd_i=B06W9JD4X2&pd_rd_w=3IMCb&pf_rd_p=c4c9c78d-928d-4cf2-ba41-9873ab24035f&pd_rd_wg=IKdC9&pf_rd_r=5ZRTXW2M7R6ABHXMRASW&pd_rd_r=91152893-9d89-4b62-b805-8bfa47ff59cb&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzQk41QzAwODE1V1BJJmVuY3J5cHRlZElkPUExMDIwNTM5MTcwMVBMR1VGOVpZNCZlbmNyeXB0ZWRBZElkPUEwOTg0MDEzM0pXSlJSNkk2NDM5MyZ3aWRnZXROYW1lPXNwX2RldGFpbCZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=

33.- Protoboards

https://www.amazon.es/Electrely-Breadboard-Prototipo-Soldaduras-Arduino/dp/B07GKVTM88/ref=sr_1_14?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=24S9H0WPVRWRV&dchild=1&keywords=breadboard&qid=1616068331&sprefix=breadboard%2Caps%2C173&sr=8-14

34. Playing a wav file in C

http://www.science.smith.edu/dftwiki/index.php/CSC231_How_to_play_a_wav_file_in_C

35.- 3D Spatial Audio using Google Resonance SDK

https://www.youtube.com/watch?v=eWbaEr_mXRE

36.- PlaySound in C

https://stackoverflow.com/questions/29998655/how-to-use-playsound-in-c

37.- ROS For Beginners (ROS Noetic, Melodic, Kinetic)

https://www.udemy.com/course/ros-for-beginners/

38.- Install ROS Noetic on Raspberry Pi 4

https://varhowto.com/install-ros-noetic-raspberry-pi-4/

39.- sensor_msgs/Range Message

http://docs.ros.org/en/api/sensor_msgs/html/msg/Range.html

40.- An Introduction to ROS, the Robot Operating System: Controlling hardware

https://www.youtube.com/watch?v=dxcU-_PGZdw

41.- Playing audio files with Python

https://raspberrypi.stackexchange.com/questions/7088/playing-audio-files-with-python

8.- Anexos.

moving_node.cpp

```
#include <ros/ros.h>
#include <std_msgs/String.h>
#include <std_msgs/Int64.h>
#include <wiringPi.h>
#include <softPwm.h>

using namespace std;

//LED
#define LED_PIN   16

//Puente en H
#define IN1       17    // Cable amarillo
#define IN2       27    // Cable naranja
#define IN3       23    // Cable verde
#define IN4       24    // Cable azul

//Sensor delante
#define Trigger1  18          //Trigger del sensor 1
#define Echo1   25        //Echo del sensor 1

//Sensor detras
#define Trigger2  8         //Trigger del sensor 2
#define Echo2   7         //Echo del sensor 2

//Sensor delante derecha
#define Trigger3  20        //Trigger del sensor 3
#define Echo3   21        //Echo del sensor 3

//Sensor delante izquierda
#define Trigger4  6       //Trigger del sensor 4
#define Echo4   26        //Echo del sensor 4

//Sensor detras derecha
#define Trigger5  5   //Trigger del sensor 5
#define Echo5   9         //Echo del sensor 5

//Sensor detras izquierda
#define Trigger6  11    //Trigger del sensor 6
#define Echo6   10    //Echo del sensor 6
```

```
// Definicion de variables relacionadas con el sensor de distancia HC-
SR04
long t;             //tiempo que demora en llegar el echo
long d;             //distancia en centimetros

// Para controlar la velocidad de los motores
#define ENA       13    // Cable gris
#define ENB       19    // Cable marron

//Defaults
int fin = 0;
int distance_delante = 0;
int distance_atras = 0;
int distance_delante_derecha = 0;
int distance_delante_izquierda = 0;
int distance_detras_derecha = 0;
int distance_detras_izquierda = 0;

std::string distance_min;
std::string sentido;

std_msgs::String audio;
ros::Publisher pub;
std::string sound = "0";

//void compruebaAudio()
//{
//  if (sound != "0")
//  {
//  pub.publish(nh.getParam(audio));
//  }
//}

void callback_receive_joystick(const std_msgs::String& msg) {
 // ROS_INFO("Message received : %s", msg.data.c_str());

  std::string detras_izquierda = "detras_izquierda";
  std::string detras_derecha = "detras_derecha";
  std::string atras = "atras";
  std::string derecha = "derecha";
  std::string delante = "delante";
  std::string izquierda = "izquierda";
```

```
std::string delante_izquierda = "delante_izquierda";
std::string delante_derecha = "delante_derecha";
std::string parar = "parar";
std::string nulo = "0";

if (msg.data.c_str() == detras_izquierda)
{
 // ROS_INFO("DETRAS IZQUIERDA");
  digitalWrite (IN1, HIGH);
  digitalWrite (IN2, LOW);
  softPwmWrite (ENA, 100);  //Velocidad motor A
  digitalWrite (IN3, LOW);
  digitalWrite (IN4, HIGH);
  softPwmWrite (ENB, 1000); //Velocidad motor B
  sentido = "detras";
} else if (msg.data.c_str() == detras_derecha) {
 // ROS_INFO("DETRAS DERECHA");
  digitalWrite (IN1, HIGH);
  digitalWrite (IN2, LOW);
  softPwmWrite (ENA, 1000); //Velocidad motor A
  digitalWrite (IN3, LOW);
  digitalWrite (IN4, HIGH);
  softPwmWrite (ENB, 100);  //Velocidad motor B
  sentido = "detras";
} else if (msg.data.c_str() == derecha) {
 // ROS_INFO("DERECHA");
  digitalWrite (IN1, LOW);
  digitalWrite (IN2, HIGH);
  softPwmWrite (ENA, 1000); //Velocidad motor A
  digitalWrite (IN3, LOW);
  digitalWrite (IN4, HIGH);
  softPwmWrite (ENB, 1000); //Velocidad motor B
} else if (msg.data.c_str() == izquierda) {
 // ROS_INFO("IZQUIERDA");
  digitalWrite (IN1, HIGH);
  digitalWrite (IN2, LOW);
  softPwmWrite (ENA, 1000); //Velocidad motor A
  digitalWrite (IN3, HIGH);
  digitalWrite (IN4, LOW);
  softPwmWrite (ENB, 1000); //Velocidad motor B
} else if (msg.data.c_str() == delante) {
 // ROS_INFO("DELANTE");
  digitalWrite (IN1, LOW);
```

```
    digitalWrite (IN2, HIGH);
    softPwmWrite (ENA, 1000); //Velocidad motor A
    digitalWrite (IN3, HIGH);
    digitalWrite (IN4, LOW);
    softPwmWrite (ENB, 1000); //Velocidad motor B
    sentido = "delante";
  } else if (msg.data.c_str() == atras) {
   // ROS_INFO("ATRAS");
    digitalWrite (IN1, HIGH);
    digitalWrite (IN2, LOW);
    softPwmWrite (ENA, 1000); //Velocidad motor A
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, HIGH);
    softPwmWrite (ENB, 1000); //Velocidad motor B
    sentido = "detras";
  } else if (msg.data.c_str() == delante_derecha) {
   // ROS_INFO("DELANTE DERECHA");
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, HIGH);
    softPwmWrite (ENA, 1000); //Velocidad motor A
    digitalWrite (IN3, HIGH);
    digitalWrite (IN4, LOW);
    softPwmWrite (ENB, 100);  //Velocidad motor B
    sentido = "delante";
  } else if (msg.data.c_str() == delante_izquierda) {
   // ROS_INFO("DELANTE IZQUIERDA");
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, HIGH);
    softPwmWrite (ENA, 100);  //Velocidad motor A
    digitalWrite (IN3, HIGH);
    digitalWrite (IN4, LOW);
    softPwmWrite (ENB, 1000); //Velocidad motor B
    sentido = "delante";
  } else if (msg.data.c_str() == parar) {
   // ROS_INFO("PARAR");
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, LOW);
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, LOW);
  }
}

//ros::NodeHandle nh;
```

```
void callback_sensor1(const std_msgs::Int64& dist1)
{
  audio.data = "0";
  //audio = "0";
  pub.publish(audio);
  distance_min = "cero";
  //ros::NodeHandle nh;
  //ros::Publisher pub1 = nh.advertise<std_msgs::String>("/audio", 10);
  distance_delante = dist1.data;
  //if (distance_delante < 30 & distance_delante > 20) {
  //  softPwmWrite (ENA, 900);  //Velocidad motor A
  //  softPwmWrite (ENB, 900);  //Velocidad motor B
  //  distance_min = "cero";
  //}
  if (distance_delante <= 20 & sentido == "delante") {
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, LOW);
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, LOW);
    audio.data = "delante";
    pub.publish(audio);
    //sound = "delante";
    //compruebaAudio();
    audio.data = "0";
    //audio = "0";
    pub.publish(audio);
    distance_min = "dos";
    //sentido == "delante";
  }
}


void callback_sensor2(const std_msgs::Int64& dist2)
{
  audio.data = "0";
  //audio = "0";
  pub.publish(audio);
  distance_min = "cero";
  //ros::NodeHandle nh;
  //ros::Publisher pub2 = nh.advertise<std_msgs::String>("/audio", 10);
  distance_atras = dist2.data;
  //if (distance_atras < 30 & distance_atras > 20) {
```

```
  //  softPwmWrite (ENA, 900);  //Velocidad motor A
  //  softPwmWrite (ENB, 900);  //Velocidad motor B
  //  distance_min = "cero";
  //}
  if (distance_atras <= 20 & sentido == "detras") {
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, LOW);
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, LOW);
    audio.data = "atras";
    sound = "atras";
    pub.publish(audio);
    //compruebaAudio();
    audio.data = "0";
    //audio = "0";
    pub.publish(audio);
    distance_min = "dos";
    sentido == "detras";
  }
}

void callback_sensor3(const std_msgs::Int64& dist3)
{
  audio.data = "0";
  //audio = "0";
  pub.publish(audio);
  distance_min = "cero";
  //ros::NodeHandle nh;
  //ros::Publisher pub3 = nh.advertise<std_msgs::String>("/audio", 10);
  distance_delante_derecha = dist3.data;
  //if (distance_delante_derecha < 25 & distance_delante_derecha > 15)
{
  //  softPwmWrite (ENA, 900);  //Velocidad motor A
  //  softPwmWrite (ENB, 900);  //Velocidad motor B
  //  distance_min = "cero";
  //}
  if (distance_delante_derecha <= 15 & sentido == "delante") {
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, LOW);
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, LOW);
    audio.data = "delante_derecha";
    sound = "delante_derecha";
```

```
    pub.publish(audio);
    //compruebaAudio();
    audio.data = "0";
    //audio = "0";
    pub.publish(audio);
    distance_min = "dos";
    sentido == "delante";
  }
}

void callback_sensor4(const std_msgs::Int64& dist4)
{
  audio.data = "0";
  //audio = "0";
  pub.publish(audio);
  distance_min = "cero";
  //ros::NodeHandle nh;
  //ros::Publisher pub4 = nh.advertise<std_msgs::String>("/audio", 10);
  distance_delante_izquierda = dist4.data;
  //if (distance_delante_izquierda < 25 & distance_delante_izquierda >
15) {
  //  softPwmWrite (ENA, 900);  //Velocidad motor A
  //  softPwmWrite (ENB, 900);  //Velocidad motor B
  //  distance_min = "cero";
  //}
  if (distance_delante_izquierda <= 15 & sentido == "delante") {
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, LOW);
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, LOW);
    audio.data = "delante_izquierda";
    sound = "delante_izquierda";
    pub.publish(audio);
    //compruebaAudio();
    audio.data = "0";
    //audio = "0";
    pub.publish(audio);
    distance_min = "dos";
    sentido == "delante";
  }
}

void callback_sensor5(const std_msgs::Int64& dist5)
```

```
{
  audio.data = "0";
  //audio = "0";
  pub.publish(audio);
  distance_min = "cero";
  //ros::NodeHandle nh;
  //ros::Publisher pub5 = nh.advertise<std_msgs::String>("/audio", 10);
  distance_detras_derecha = dist5.data;
  //if (distance_detras_derecha < 25 & distance_detras_derecha > 15) {
  //  softPwmWrite (ENA, 900);  //Velocidad motor A
  //  softPwmWrite (ENB, 900);  //Velocidad motor B
  //  distance_min = "cero";
  //}
  if (distance_detras_derecha <= 15 & sentido == "detras") {
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, LOW);
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, LOW);
    audio.data = "detras_derecha";
    sound = "detras_derecha";
    pub.publish(audio);
    //compruebaAudio();
    audio.data = "0";
    //audio = "0";
    pub.publish(audio);
    distance_min = "dos";
    sentido == "detras";
  }
}

void callback_sensor6(const std_msgs::Int64& dist6)
{
  audio.data = "0";
  //audio = "0";
  pub.publish(audio);
  distance_min = "cero";
  //ros::NodeHandle nh;
  //ros::Publisher pub6 = nh.advertise<std_msgs::String>("/audio", 10);
  distance_detras_izquierda = dist6.data;
 // if (distance_detras_izquierda < 25 & distance_detras_izquierda >
15) {
 //   softPwmWrite (ENA, 900);  //Velocidad motor A
 //   softPwmWrite (ENB, 900);  //Velocidad motor B
```

```
 //   distance_min = "cero";
 // }
  if (distance_detras_izquierda <= 15 & sentido == "detras") {
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, LOW);
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, LOW);
    audio.data = "detras_izquierda";
    pub.publish(audio);
    //sound = "detras_izquierda";
    //compruebaAudio();
    audio.data = "0";
    //audio = "0";
    pub.publish(audio);
    distance_min = "dos";
    sentido == "detras";
  }
}

int main (int argc, char **argv)
{
  ros::init(argc, argv, "moving_node",
ros::init_options::AnonymousName);
  ros::NodeHandle nh;
  pub = nh.advertise<std_msgs::String>("/audio", 10);
  //ros::Publisher pub1 = nh.advertise<std_msgs::String>("/audio", 10);
  //ros::Publisher pub2 = nh.advertise<std_msgs::String>("/audio", 10);
  //ros::Publisher pub3 = nh.advertise<std_msgs::String>("/audio", 10);
  //ros::Publisher pub4 = nh.advertise<std_msgs::String>("/audio", 10);
  //ros::Publisher pub5 = nh.advertise<std_msgs::String>("/audio", 10);
  //ros::Publisher pub6 = nh.advertise<std_msgs::String>("/audio", 10);

  wiringPiSetupGpio();
  ROS_INFO("This node has been started");

  ros::Subscriber sub = nh.subscribe("/move", 1000,
callback_receive_joystick);
  ros::Subscriber subsens1 = nh.subscribe("/sensor1_node", 1000,
callback_sensor1);
  ros::Subscriber subsens2 = nh.subscribe("/sensor2_node", 1000,
callback_sensor2);
  ros::Subscriber subsens3 = nh.subscribe("/sensor3_node", 1000,
callback_sensor3);
```

```
  ros::Subscriber subsens4 = nh.subscribe("/sensor4_node", 1000,
callback_sensor4);
  ros::Subscriber subsens5 = nh.subscribe("/sensor5_node", 1000,
callback_sensor5);
  ros::Subscriber subsens6 = nh.subscribe("/sensor6_node", 1000,
callback_sensor6);

  pinMode(LED_PIN, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  softPwmCreate(ENA,1,1000);
  softPwmCreate(ENB,1,1000);

  int i = 0;
  while (fin == 0)
  {
    for(i=0;i<2;i++)
    {
      digitalWrite(LED_PIN, HIGH);
      ROS_INFO("Set GPIO HIGH");
      ros::Duration(1.0).sleep();
      digitalWrite(LED_PIN, LOW);
      ROS_INFO("Set GPIO LOW");
      ros::Duration(1.0).sleep();
      fin = 1;
    }
  }
  //compruebaAudio();
  ros::spin();
}
```

audio.py

```
#!/usr/bin/env python3

import rospy
from pygame import mixer
from std_msgs.msg import String
import pygame
import os
import time
# from playsound import playsound
# from pydub import AudioSegment
# from pydub.playback import play

def callback_receive_audio(msg):
    rospy.loginfo(msg)
    if (msg.data == "delante"):

mixer.music.load('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delante.wav')
        mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
       # os.system('mpg321 /home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delante.mp3 &')
       # os.system('omxplayer -o alsa /home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delante.mp3')
       # playsound('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delante.mp3')
       # song1 = AudioSegment.from_wav("/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delante.wav")
       # play(song1)
       # time.sleep(2)
        exit(0)
    elif (msg.data == "detras_izquierda"):

mixer.music.load('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras_izquierda.wav')
        mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
       # os.system('mpg321 /home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras_izquierda.mp3 &')
```

```
        # os.system('omxplayer -o alsa
/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras_izquierda.mp3')
        #
playsound('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras_izquie
rda.mp3')
        # song2 =
AudioSegment.from_wav("/home/pi/ros_catkin_ws/src/elco_oursin/scripts/D
etras_izquierda.wav")
        # play(song2)
        # time.sleep(2)
         exit(0)
    elif (msg.data == "detras_derecha"):

mixer.music.load('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras
_derecha.wav')
         mixer.music.play()
         while pygame.mixer.music.get_busy() == True:
             continue
        # os.system('mpg321
/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras_derecha.mp3 &')
        # os.system('omxplayer -o alsa
/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras_derecha.mp3')
        #
playsound('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras_derech
a.mp3')
        # song3 =
AudioSegment.from_wav("/home/pi/ros_catkin_ws/src/elco_oursin/scripts/D
etras_derecha.wav")
        # play(song3)
        # time.sleep(2)
         exit(0)
    elif (msg.data == "atras"):

mixer.music.load('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras
.wav')
         mixer.music.play()
         while pygame.mixer.music.get_busy() == True:
             continue
        # os.system('mpg321
/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras.mp3 &')
        # os.system('omxplayer -o alsa
/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras.mp3')
```

```
        #
playsound('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Detras.mp3')
        # song4 =
AudioSegment.from_wav("/home/pi/ros_catkin_ws/src/elco_oursin/scripts/D
etras.wav")
        # play(song4)
        # time.sleep(2)
         exit(0)
    elif (msg.data == "delante_derecha"):

mixer.music.load('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delant
e_derecha.wav')
         mixer.music.play()
         while pygame.mixer.music.get_busy() == True:
             continue
        # os.system('mpg321
/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delante_derecha.mp3 &')
        # os.system('omxplayer -o alsa
/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delante_derecha.mp3')
        #
playsound('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delante_derec
ha.mp3')
        # song5 =
AudioSegment.from_wav("/home/pi/ros_catkin_ws/src/elco_oursin/scripts/D
elante_derecha.wav")
        # play(song5)
        # time.sleep(2)
         exit(0)
    elif (msg.data == "delante_izquierda"):

mixer.music.load('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delant
e_izquierda.wav')
         mixer.music.play()
         while pygame.mixer.music.get_busy() == True:
             continue
        # os.system('mpg321
/home/pi/ros_catkin_ws/src/elco_oursin/scripts/izquierda.mp3 &')
        # os.system('omxplayer -o alsa
/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delante_izquierda.mp3')
        #
playsound('/home/pi/ros_catkin_ws/src/elco_oursin/scripts/Delante_izqui
erda.mp3')
```

```
        # song6 =
AudioSegment.from_wav("/home/pi/ros_catkin_ws/src/elco_oursin/scripts/D
elante_izquierda.wav")
        # play(song6)
        # time.sleep(2)
         exit(0)

if __name__ == '__main__':
    rospy.init_node('audio_node', anonymous=True)
    sub = rospy.Subscriber("/audio", String, callback_receive_audio)
    mixer.init()
    pygame.mixer.pre_init(44100, 16, 2, 4096) #frequency, size,
channels, buffersize
    pygame.init()
    rospy.spin()
```

sensor1.py

```
#!/usr/bin/env python3

#Sensor delante

import RPi.GPIO as GPIO
import time
import rospy
#import decimal
from std_msgs.msg import Int64

# Pin GPIO donde está conectado el activador (entrada) del
# sensor HC-SR04.
TRIG1 = 18

# Pin GPIO donde está conectado el eco (salida) del sensor
# HC-SR04.
ECHO1 = 25

def distance1():

    try:
        # Ciclo infinito.
        # Para terminar el programa se debe presionar Ctrl-C.
        while True:

            # Apagar el pin activador y permitir un par de
            # segundos para que se estabilice.
            GPIO.output(TRIG1, GPIO.LOW)
            print ("Esperando a que el sensor se estabilice")
            time.sleep(0.2)

            # Prender el pin activador por 10 microsegundos
            # y después volverlo a apagar.
            GPIO.output(TRIG1, GPIO.HIGH)
            time.sleep(0.00001)
            GPIO.output(TRIG1, GPIO.LOW)

            # En este momento el sensor envía 8 pulsos
            # ultrasónicos de 40kHz y coloca su salida ECHO
            # en HIGH. Se debe detectar este evento e iniciar
            # la medición del tiempo.
            print ("Iniciando eco")
```

```
            while True:
                pulso_inicio = time.time()
                if GPIO.input(ECHO1) == GPIO.HIGH:
                    break

            # La salida ECHO se mantendrá en HIGH hasta recibir
            # el eco reflejado por el obstáculo. En ese momento
            # el sensor pondrá ECHO en LOW y se debe terminar
            # la medición del tiempo.
            while True:
                pulso_fin = time.time()
                if GPIO.input(ECHO1) == GPIO.LOW:
                    break

            # La medición del tiempo es en segundos.
            duracion = pulso_fin - pulso_inicio

            # Calcular la distancia usando la velocidad del
            # sonido y considerando que la duración incluye
            # la ida y vuelta.
            distancia_delante = Int64()
            distancia_delante.data = int((34300 * duracion) / 2)

            # Imprimir resultado.
            print ("Distancia: %.2f cm" + str(distancia_delante))

            # Publicar el resultado
            pub.publish(distancia_delante)

    finally:
        # Reiniciar todos los canales de GPIO.
        GPIO.cleanup()

if __name__ == '__main__':
    rospy.init_node('sensor1', anonymous=True)
    rospy.loginfo("Sensor 1 has been started")
    pub = rospy.Publisher("/sensor1_node", Int64, queue_size=10)

    GPIO.setwarnings(False)
    # Indicar que se usa el esquema de numeración de pines
    # de BCM (Broadcom SOC channel), es decir los números de
    # pines GPIO (General-Purpose Input/Output).
    GPIO.setmode(GPIO.BCM)
```

```
# Establecer que TRIG es un canal de salida.
GPIO.setup(TRIG1, GPIO.OUT)

# Establecer que ECHO es un canal de entrada.
GPIO.setup(ECHO1, GPIO.IN)

distance1()
rospy.spin()
```

sensor2.py

```
#!/usr/bin/env python3

#Sensor delante

import RPi.GPIO as GPIO
import time
import rospy
#import decimal
from std_msgs.msg import Int64

# Pin GPIO donde está conectado el activador (entrada) del
# sensor HC-SR04.
TRIG2 = 8

# Pin GPIO donde está conectado el eco (salida) del sensor
# HC-SR04.
ECHO2 = 7

def distance2():

    try:
        # Ciclo infinito.
        # Para terminar el programa se debe presionar Ctrl-C.
        while True:

            # Apagar el pin activador y permitir un par de
            # segundos para que se estabilice.
            GPIO.output(TRIG2, GPIO.LOW)
            print ("Esperando a que el sensor se estabilice")
            time.sleep(0.2)

            # Prender el pin activador por 10 microsegundos
            # y después volverlo a apagar.
            GPIO.output(TRIG2, GPIO.HIGH)
            time.sleep(0.00001)
            GPIO.output(TRIG2, GPIO.LOW)

            # En este momento el sensor envía 8 pulsos
            # ultrasónicos de 40kHz y coloca su salida ECHO
            # en HIGH. Se debe detectar este evento e iniciar
            # la medición del tiempo.
```

```
            print ("Iniciando eco")
            while True:
                pulso_inicio = time.time()
                if GPIO.input(ECHO2) == GPIO.HIGH:
                    break

            # La salida ECHO se mantendrá en HIGH hasta recibir
            # el eco reflejado por el obstáculo. En ese momento
            # el sensor pondrá ECHO en LOW y se debe terminar
            # la medición del tiempo.
            while True:
                pulso_fin = time.time()
                if GPIO.input(ECHO2) == GPIO.LOW:
                    break

            # La medición del tiempo es en segundos.
            duracion = pulso_fin - pulso_inicio

            # Calcular la distancia usando la velocidad del
            # sonido y considerando que la duración incluye
            # la ida y vuelta.
            distancia_atras = Int64()
            distancia_atras.data = int((34300 * duracion) / 2)

            # Imprimir resultado.
            print ("Distancia: %.2f cm" + str(distancia_atras))

            # Publicar el resultado
            pub.publish(distancia_atras)

    finally:
        # Reiniciar todos los canales de GPIO.
        GPIO.cleanup()

if __name__ == '__main__':
    rospy.init_node('sensor2', anonymous=True)
    rospy.loginfo("Sensor 2 has been started")
    pub = rospy.Publisher("/sensor2_node", Int64, queue_size=10)

    GPIO.setwarnings(False)
    # Indicar que se usa el esquema de numeración de pines
    # de BCM (Broadcom SOC channel), es decir los números de
    # pines GPIO (General-Purpose Input/Output).
```

```
GPIO.setmode(GPIO.BCM)

# Establecer que TRIG es un canal de salida.
GPIO.setup(TRIG2, GPIO.OUT)

# Establecer que ECHO es un canal de entrada.
GPIO.setup(ECHO2, GPIO.IN)

distance2()
rospy.spin()
```

sensor3.py

```
#!/usr/bin/env python3

#Sensor delante derecha

import RPi.GPIO as GPIO
import time
import rospy
#import decimal
from std_msgs.msg import Int64

# Pin GPIO donde está conectado el activador (entrada) del
# sensor HC-SR04.
TRIG3 = 20

# Pin GPIO donde está conectado el eco (salida) del sensor
# HC-SR04.
ECHO3 = 21

def distance3():

    try:
        # Ciclo infinito.
        # Para terminar el programa se debe presionar Ctrl-C.
        while True:

            # Apagar el pin activador y permitir un par de
            # segundos para que se estabilice.
            GPIO.output(TRIG3, GPIO.LOW)
            print ("Esperando a que el sensor se estabilice")
            time.sleep(0.2)

            # Prender el pin activador por 10 microsegundos
            # y después volverlo a apagar.
            GPIO.output(TRIG3, GPIO.HIGH)
            time.sleep(0.00001)
            GPIO.output(TRIG3, GPIO.LOW)

            # En este momento el sensor envía 8 pulsos
            # ultrasónicos de 40kHz y coloca su salida ECHO
            # en HIGH. Se debe detectar este evento e iniciar
            # la medición del tiempo.
            print ("Iniciando eco")
```

```
            while True:
                pulso_inicio = time.time()
                if GPIO.input(ECHO3) == GPIO.HIGH:
                    break

            # La salida ECHO se mantendrá en HIGH hasta recibir
            # el eco reflejado por el obstáculo. En ese momento
            # el sensor pondrá ECHO en LOW y se debe terminar
            # la medición del tiempo.
            while True:
                pulso_fin = time.time()
                if GPIO.input(ECHO3) == GPIO.LOW:
                    break

            # La medición del tiempo es en segundos.
            duracion = pulso_fin - pulso_inicio

            # Calcular la distancia usando la velocidad del
            # sonido y considerando que la duración incluye
            # la ida y vuelta.
            distancia_delante_derecha = Int64()
            distancia_delante_derecha.data = int((34300 * duracion) /
2)

            # Imprimir resultado.
            print ("Distancia: %.2f cm" +
str(distancia_delante_derecha))

            # Publicar el resultado
            pub.publish(distancia_delante_derecha)

    finally:
        # Reiniciar todos los canales de GPIO.
        GPIO.cleanup()

if __name__ == '__main__':
    rospy.init_node('sensor3', anonymous=True)
    rospy.loginfo("Sensor 3 has been started")
    pub = rospy.Publisher("/sensor3_node", Int64, queue_size=10)

    GPIO.setwarnings(False)
    # Indicar que se usa el esquema de numeración de pines
    # de BCM (Broadcom SOC channel), es decir los números de
```

```
# pines GPIO (General-Purpose Input/Output).
GPIO.setmode(GPIO.BCM)

# Establecer que TRIG es un canal de salida.
GPIO.setup(TRIG3, GPIO.OUT)

# Establecer que ECHO es un canal de entrada.
GPIO.setup(ECHO3, GPIO.IN)

distance3()
rospy.spin()
```

sensor4.py

```
#!/usr/bin/env python3

#Sensor delante izquierda

import RPi.GPIO as GPIO
import time
import rospy
#import decimal
from std_msgs.msg import Int64

# Pin GPIO donde está conectado el activador (entrada) del
# sensor HC-SR04.
TRIG4 = 6

# Pin GPIO donde está conectado el eco (salida) del sensor
# HC-SR04.
ECHO4 = 26

def distance4():

    try:
        # Ciclo infinito.
        # Para terminar el programa se debe presionar Ctrl-C.
        while True:

            # Apagar el pin activador y permitir un par de
            # segundos para que se estabilice.
            GPIO.output(TRIG4, GPIO.LOW)
            print ("Esperando a que el sensor se estabilice")
            time.sleep(0.2)

            # Prender el pin activador por 10 microsegundos
            # y después volverlo a apagar.
            GPIO.output(TRIG4, GPIO.HIGH)
            time.sleep(0.00001)
            GPIO.output(TRIG4, GPIO.LOW)

            # En este momento el sensor envía 8 pulsos
            # ultrasónicos de 40kHz y coloca su salida ECHO
            # en HIGH. Se debe detectar este evento e iniciar
            # la medición del tiempo.
            print ("Iniciando eco")
```

```
            while True:
                pulso_inicio = time.time()
                if GPIO.input(ECHO4) == GPIO.HIGH:
                    break

            # La salida ECHO se mantendrá en HIGH hasta recibir
            # el eco reflejado por el obstáculo. En ese momento
            # el sensor pondrá ECHO en LOW y se debe terminar
            # la medición del tiempo.
            while True:
                pulso_fin = time.time()
                if GPIO.input(ECHO4) == GPIO.LOW:
                    break

            # La medición del tiempo es en segundos.
            duracion = pulso_fin - pulso_inicio

            # Calcular la distancia usando la velocidad del
            # sonido y considerando que la duración incluye
            # la ida y vuelta.
            distancia_delante_izquierda = Int64()
            distancia_delante_izquierda.data = int((34300 * duracion) /
2)

            # Imprimir resultado.
            print ("Distancia: %.2f cm" +
str(distancia_delante_izquierda))

            # Publicar el resultado
            pub.publish(distancia_delante_izquierda)

    finally:
        # Reiniciar todos los canales de GPIO.
        GPIO.cleanup()

if __name__ == '__main__':
    rospy.init_node('sensor4', anonymous=True)
    rospy.loginfo("Sensor 4 has been started")
    pub = rospy.Publisher("/sensor4_node", Int64, queue_size=10)

    GPIO.setwarnings(False)
    # Indicar que se usa el esquema de numeración de pines
    # de BCM (Broadcom SOC channel), es decir los números de
```

```
# pines GPIO (General-Purpose Input/Output).
GPIO.setmode(GPIO.BCM)

# Establecer que TRIG es un canal de salida.
GPIO.setup(TRIG4, GPIO.OUT)

# Establecer que ECHO es un canal de entrada.
GPIO.setup(ECHO4, GPIO.IN)

distance4()
rospy.spin()
```

sensor5.py

```
#!/usr/bin/env python3

#Sensor atras derecha

import RPi.GPIO as GPIO
import time
import rospy
#import decimal
from std_msgs.msg import Int64

# Pin GPIO donde está conectado el activador (entrada) del
# sensor HC-SR04.
TRIG5 = 5

# Pin GPIO donde está conectado el eco (salida) del sensor
# HC-SR04.
ECHO5 = 9

def distance5():

    try:
        # Ciclo infinito.
        # Para terminar el programa se debe presionar Ctrl-C.
        while True:

            # Apagar el pin activador y permitir un par de
            # segundos para que se estabilice.
            GPIO.output(TRIG5, GPIO.LOW)
            print ("Esperando a que el sensor se estabilice")
            time.sleep(0.2)

            # Prender el pin activador por 10 microsegundos
            # y después volverlo a apagar.
            GPIO.output(TRIG5, GPIO.HIGH)
            time.sleep(0.00001)
            GPIO.output(TRIG5, GPIO.LOW)

            # En este momento el sensor envía 8 pulsos
            # ultrasónicos de 40kHz y coloca su salida ECHO
            # en HIGH. Se debe detectar este evento e iniciar
            # la medición del tiempo.
            print ("Iniciando eco")
```

```
            while True:
                pulso_inicio = time.time()
                if GPIO.input(ECHO5) == GPIO.HIGH:
                    break

            # La salida ECHO se mantendrá en HIGH hasta recibir
            # el eco reflejado por el obstáculo. En ese momento
            # el sensor pondrá ECHO en LOW y se debe terminar
            # la medición del tiempo.
            while True:
                pulso_fin = time.time()
                if GPIO.input(ECHO5) == GPIO.LOW:
                    break

            # La medición del tiempo es en segundos.
            duracion = pulso_fin - pulso_inicio

            # Calcular la distancia usando la velocidad del
            # sonido y considerando que la duración incluye
            # la ida y vuelta.
            distancia_atras_derecha = Int64()
            distancia_atras_derecha.data = int((34300 * duracion) / 2)

            # Imprimir resultado.
            print ("Distancia: %.2f cm" + str(distancia_atras_derecha))

            # Publicar el resultado
            pub.publish(distancia_atras_derecha)

    finally:
        # Reiniciar todos los canales de GPIO.
        GPIO.cleanup()

if __name__ == '__main__':
    rospy.init_node('sensor5', anonymous=True)
    rospy.loginfo("Sensor 5 has been started")
    pub = rospy.Publisher("/sensor5_node", Int64, queue_size=10)

    GPIO.setwarnings(False)
    # Indicar que se usa el esquema de numeración de pines
    # de BCM (Broadcom SOC channel), es decir los números de
    # pines GPIO (General-Purpose Input/Output).
    GPIO.setmode(GPIO.BCM)
```

```
# Establecer que TRIG es un canal de salida.
GPIO.setup(TRIG5, GPIO.OUT)

# Establecer que ECHO es un canal de entrada.
GPIO.setup(ECHO5, GPIO.IN)

distance5()
rospy.spin()
```

sensor6.py

```
#!/usr/bin/env python3

#Sensor atras izquierda

import RPi.GPIO as GPIO
import time
import rospy
#import decimal
from std_msgs.msg import Int64

# Pin GPIO donde está conectado el activador (entrada) del
# sensor HC-SR04.
TRIG6 = 11

# Pin GPIO donde está conectado el eco (salida) del sensor
# HC-SR04.
ECHO6 = 10

def distance6():

    try:
        # Ciclo infinito.
        # Para terminar el programa se debe presionar Ctrl-C.
        while True:

            # Apagar el pin activador y permitir un par de
            # segundos para que se estabilice.
            GPIO.output(TRIG6, GPIO.LOW)
            print ("Esperando a que el sensor se estabilice")
            time.sleep(0.2)

            # Prender el pin activador por 10 microsegundos
            # y después volverlo a apagar.
            GPIO.output(TRIG6, GPIO.HIGH)
            time.sleep(0.00001)
            GPIO.output(TRIG6, GPIO.LOW)

            # En este momento el sensor envía 8 pulsos
            # ultrasónicos de 40kHz y coloca su salida ECHO
            # en HIGH. Se debe detectar este evento e iniciar
            # la medición del tiempo.
            print ("Iniciando eco")
```

```
            while True:
                pulso_inicio = time.time()
                if GPIO.input(ECHO6) == GPIO.HIGH:
                    break

            # La salida ECHO se mantendrá en HIGH hasta recibir
            # el eco reflejado por el obstáculo. En ese momento
            # el sensor pondrá ECHO en LOW y se debe terminar
            # la medición del tiempo.
            while True:
                pulso_fin = time.time()
                if GPIO.input(ECHO6) == GPIO.LOW:
                    break

            # La medición del tiempo es en segundos.
            duracion = pulso_fin - pulso_inicio

            # Calcular la distancia usando la velocidad del
            # sonido y considerando que la duración incluye
            # la ida y vuelta.
            distancia_atras_izquierda = Int64()
            distancia_atras_izquierda.data = int((34300 * duracion) /
2)

            # Imprimir resultado.
            print ("Distancia: %.2f cm" +
str(distancia_atras_izquierda))

            # Publicar el resultado
            pub.publish(distancia_atras_izquierda)

    finally:
        # Reiniciar todos los canales de GPIO.
        GPIO.cleanup()

if __name__ == '__main__':
    rospy.init_node('sensor6', anonymous=True)
    rospy.loginfo("Sensor 6 has been started")
    pub = rospy.Publisher("/sensor6_node", Int64, queue_size=10)

    GPIO.setwarnings(False)
    # Indicar que se usa el esquema de numeración de pines
    # de BCM (Broadcom SOC channel), es decir los números de
```

```
# pines GPIO (General-Purpose Input/Output).
GPIO.setmode(GPIO.BCM)

# Establecer que TRIG es un canal de salida.
GPIO.setup(TRIG6, GPIO.OUT)

# Establecer que ECHO es un canal de entrada.
GPIO.setup(ECHO6, GPIO.IN)

distance6()
rospy.spin()
```

joystick.py

```
#!/usr/bin/env python3

import rospy
import pygame
import time
from std_msgs.msg import Int64
from std_msgs.msg import String

def joy():

    pygame.init()
    j = pygame.joystick.Joystick(0)
    j.init()

    try:
        while True:
            events = pygame.event.get()
            for event in events:
                if event.type == pygame.JOYBUTTONDOWN:
                    print("Button Pressed")
                    if j.get_button(6):
                        print("L2 pulsado / Izquierda detras 360")
                        rospy.set_param("/FLAG_DETRAS_IZQUIERDA_360",
"detras_izquierda")
                        rospy.set_param("/flag",
rospy.get_param("/FLAG_DETRAS_IZQUIERDA_360"))
                        rospy.set_param("/FLAG_DETRAS_IZQUIERDA_360",
"0")
                    elif j.get_button(7):
                        print("R2 pulsado / Derecha detras 360")
                        rospy.set_param("/FLAG_DETRAS_DERECHA_360",
"detras_derecha")
                        rospy.set_param("/flag",
rospy.get_param("/FLAG_DETRAS_DERECHA_360"))
                        rospy.set_param("/FLAG_DETRAS_DERECHA_360",
"0")
                    elif j.get_button(0):
                        print("X pulsada / Atras")
                        rospy.set_param("/FLAG_ATRAS", "atras")
                        rospy.set_param("/flag",
rospy.get_param("/FLAG_ATRAS"))
                        rospy.set_param("/FLAG_ATRAS", "0")
```

```
                    elif j.get_button(1):
                        print("Circulo pulsado / Derecha")
                        rospy.set_param("/FLAG_DERECHA", "derecha")
                        rospy.set_param("/flag",
rospy.get_param("/FLAG_DERECHA"))
                        rospy.set_param("/FLAG_DERECHA", "0")
                    elif j.get_button(2):
                        print("Triangulo pulsado / Delante")
                        rospy.set_param("/FLAG_DELANTE", "delante")
                        rospy.set_param("/flag",
rospy.get_param("/FLAG_DELANTE"))
                        rospy.set_param("/FLAG_DELANTE", "0")
                    elif j.get_button(3):
                        print("Cuadrado pulsado / Izquierda")
                        rospy.set_param("/FLAG_IZQUIERDA", "izquierda")
                        rospy.set_param("/flag",
rospy.get_param("/FLAG_IZQUIERDA"))
                        rospy.set_param("/FLAG_IZQUIERDA", "0")
                    elif j.get_button(4):
                        print("L1 pulsado / Izquierda delante 360")
                        rospy.set_param("/FLAG_DELANTE_IZQUIERDA_360",
"delante_izquierda")
                        rospy.set_param("/flag",
rospy.get_param("/FLAG_DELANTE_IZQUIERDA_360"))
                        rospy.set_param("/FLAG_DELANTE_IZQUIERDA_360",
"0")
                    elif j.get_button(5):
                        print("R1 pulsado / Derecha delante 360")
                        rospy.set_param("/FLAG_DELANTE_DERECHA_360",
"delante_derecha")
                        rospy.set_param("/flag",
rospy.get_param("/FLAG_DELANTE_DERECHA_360"))
                        rospy.set_param("/FLAG_DELANTE_DERECHA_360",
"0")
                    elif j.get_button(8):
                        print("Share pulsado / Salirse")
                        rospy.set_param("/FLAG_PARAR", "parar")
                        rospy.set_param("/flag",
rospy.get_param("/FLAG_PARAR"))
                        rospy.set_param("/FLAG_PARAR", "0")
                        exit(0);
                    elif j.get_button(9):
                        print("Options pulsado")
```

```
                    elif j.get_button(10):
                        print("PS pulsado / Parar")
                        rospy.set_param("/FLAG_PARAR", "parar")
                        rospy.set_param("/flag",
rospy.get_param("/FLAG_PARAR"))
                        rospy.set_param("/FLAG_PARAR", "0")
                    elif j.get_button(11):
                        print("L3 pulsado")
                    elif j.get_button(12):
                        print("R3 pulsado")
                elif event.type == pygame.JOYBUTTONUP:
                    print("Button Released")

                msg = String()
                dato = str(rospy.get_param("/flag"))
                msg.data = dato
                pub.publish(msg)
                #time.sleep(0.05)
            rospy.set_param("/flag", "0")

    except KeyboardInterrupt:
        print("EXITING NOW")
        j.quit()

if __name__ == '__main__':
    rospy.init_node('joystick', anonymous=True)
    rospy.loginfo("This node has been started")
    rospy.set_param("/flag", "parar")
    pub = rospy.Publisher("/move", String, queue_size=10)
   # rospy.set_param("/flag", "0")
    joy()
    rospy.spin()
```

elco_oursin_app.launch

```
<launch>

    <param name="/FLAG_DERECHA" type="string" value="0" />
        <param name="/FLAG_IZQUIERDA" type="string" value="0" />
    <param name="/FLAG_DELANTE" type="string" value="0" />
    <param name="/FLAG_ATRAS" type="string" value="0" />
    <param name="/FLAG_DELANTE_DERECHA_360" type="string" value="0" />
    <param name="/FLAG_DETRAS_DERECHA_360" type="string" value="0" />
    <param name="/FLAG_DELANTE_IZQUIERDA_360" type="string" value="0"
/>
    <param name="/FLAG_DETRAS_IZQUIERDA_360" type="string" value="0" />
    <param name="/FLAG_PARAR" type="string" value="0" />
    <param name="/FLAG_DELANTE_360" type="string" value="0" />
    <param name="/FLAG_SENSOR1" type="string" value="0" />
    <param name="/FLAG_SENSOR2" type="string" value="0" />
    <param name="/FLAG_SENSOR3" type="string" value="0" />
    <param name="/FLAG_SENSOR4" type="string" value="0" />
    <param name="/FLAG_SENSOR5" type="string" value="0" />
    <param name="/FLAG_SENSOR6" type="string" value="0" />
        <param name="/FLAG_MSG" type="string" value="0" />
        <param name="/flag" type="string" value="0" />
        <param name="/flagaudio" type="string" value="0" />
        <param name="/FLAG_MSG" type="string" value="0" />

    <node name="joystick" pkg="elco_oursin" type="joystick.py" />
        <node name="moving_node" pkg="elco_oursin" type="moving_node"
/>
    <node name="sensor1_node" pkg="elco_oursin" type="sensor1.py" />
    <node name="sensor2_node" pkg="elco_oursin" type="sensor2.py" />
    <node name="sensor3_node" pkg="elco_oursin" type="sensor3.py" />
    <node name="sensor4_node" pkg="elco_oursin" type="sensor4.py" />
    <node name="sensor5_node" pkg="elco_oursin" type="sensor5.py" />
    <node name="sensor6_node" pkg="elco_oursin" type="sensor6.py" />
    <node name="audio_node" pkg="elco_oursin" type="audio.py" />

</launch>
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0.2)
project(elco_oursin)

## Compile as C++11, supported in ROS Kinetic and newer
# add_compile_options(-std=c++11)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
)

## System dependencies are found with CMake's conventions
# find_package(Boost REQUIRED COMPONENTS system)


## Uncomment this if the package has a setup.py. This macro ensures
## modules and global scripts declared therein get installed
## See http://ros.org/doc/api/catkin/html/user_guide/setup_dot_py.html
# catkin_python_setup()

################################################
## Declare ROS messages, services and actions ##
################################################

## To declare and build messages, services or actions from within this
## package, follow these steps:
## * Let MSG_DEP_SET be the set of packages whose message types you use
in
##   your messages/services/actions (e.g. std_msgs, actionlib_msgs,
...).
## * In the file package.xml:
##   * add a build_depend tag for "message_generation"
##   * add a build_depend and a exec_depend tag for each package in
MSG_DEP_SET
##   * If MSG_DEP_SET isn't empty the following dependency has been
pulled in
##     but can be declared for certainty nonetheless:
##     * add a exec_depend tag for "message_runtime"
```

```
## * In this file (CMakeLists.txt):
##   * add "message_generation" and every package in MSG_DEP_SET to
##     find_package(catkin REQUIRED COMPONENTS ...)
##   * add "message_runtime" and every package in MSG_DEP_SET to
##     catkin_package(CATKIN_DEPENDS ...)
##   * uncomment the add_*_files sections below as needed
##     and list every .msg/.srv/.action file to be processed
##   * uncomment the generate_messages entry below
##   * add every package in MSG_DEP_SET to
generate_messages(DEPENDENCIES ...)

## Generate messages in the 'msg' folder
# add_message_files(
#   FILES
#   Message1.msg
#   Message2.msg
# )

## Generate services in the 'srv' folder
# add_service_files(
#   FILES
#   Service1.srv
#   Service2.srv
# )

## Generate actions in the 'action' folder
# add_action_files(
#   FILES
#   Action1.action
#   Action2.action
# )

## Generate added messages and services with any dependencies listed
here
# generate_messages(
#   DEPENDENCIES
#   std_msgs
# )

################################################
## Declare ROS dynamic reconfigure parameters ##
################################################
```

```
## To declare and build dynamic reconfigure parameters within this
## package, follow these steps:
## * In the file package.xml:
##   * add a build_depend and a exec_depend tag for
"dynamic_reconfigure"
## * In this file (CMakeLists.txt):
##   * add "dynamic_reconfigure" to
##     find_package(catkin REQUIRED COMPONENTS ...)
##   * uncomment the "generate_dynamic_reconfigure_options" section
below
##     and list every .cfg file to be processed

## Generate dynamic reconfigure parameters in the 'cfg' folder
# generate_dynamic_reconfigure_options(
#   cfg/DynReconf1.cfg
#   cfg/DynReconf2.cfg
# )

###################################
## catkin specific configuration ##
###################################
## The catkin_package macro generates cmake config files for your
package
## Declare things to be passed to dependent projects
## INCLUDE_DIRS: uncomment this if your package contains header files
## LIBRARIES: libraries you create in this project that dependent
projects also need
## CATKIN_DEPENDS: catkin_packages dependent projects also need
## DEPENDS: system dependencies of this project that dependent projects
also need
catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES elco_oursin
#  CATKIN_DEPENDS roscpp rospy std_msgs
#  DEPENDS system_lib
)

###########
## Build ##
###########

## Specify additional locations of header files
## Your package locations should be listed before other locations
```

```
include_directories(
# include
  ${catkin_INCLUDE_DIRS}
)

## Declare a C++ library
# add_library(${PROJECT_NAME}
#   src/${PROJECT_NAME}/elco_oursin.cpp
# )

## Add cmake target dependencies of the library
## as an example, code may need to be generated before libraries
## either from message generation or dynamic reconfigure
# add_dependencies(${PROJECT_NAME} ${${PROJECT_NAME}_EXPORTED_TARGETS}
${catkin_EXPORTED_TARGETS})

## Declare a C++ executable
## With catkin_make all packages are built within a single CMake
context
## The recommended prefix ensures that target names across packages
don't collide
# add_executable(${PROJECT_NAME}_node src/elco_oursin_node.cpp)

add_executable(moving_node src/moving_node.cpp)
target_link_libraries(moving_node ${catkin_LIBRARIES})
find_library(wiringPi_LIB wiringPi)
target_link_libraries(moving_node ${wiringPi_LIB})

#add_executable(audio_node src/audio_node.cpp)
#target_link_libraries(audio_node ${catkin_LIBRARIES})

## Rename C++ executable without prefix
## The above recommended prefix causes long target names, the following
renames the
## target back to the shorter version for ease of user use
## e.g. "rosrun someones_pkg node" instead of "rosrun someones_pkg
someones_pkg_node"
# set_target_properties(${PROJECT_NAME}_node PROPERTIES OUTPUT_NAME
node PREFIX "")

## Add cmake target dependencies of the executable
## same as for the library above
```

```
# add_dependencies(${PROJECT_NAME}_node
${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

#############
## Install ##
#############

# all install targets should use catkin DESTINATION variables
# See http://ros.org/doc/api/catkin/html/adv_user_guide/variables.html

## Mark executable scripts (Python etc.) for installation
## in contrast to setup.py, you can choose the destination
# catkin_install_python(PROGRAMS
#   scripts/my_python_script
#   DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
# )

## Mark executables for installation
## See
http://docs.ros.org/melodic/api/catkin/html/howto/format1/building_exec
utables.html
# install(TARGETS ${PROJECT_NAME}_node
#   RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
# )

## Mark libraries for installation
## See
http://docs.ros.org/melodic/api/catkin/html/howto/format1/building_libr
aries.html
# install(TARGETS ${PROJECT_NAME}
#   ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
#   LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
#   RUNTIME DESTINATION ${CATKIN_GLOBAL_BIN_DESTINATION}
# )

## Mark cpp header files for installation
# install(DIRECTORY include/${PROJECT_NAME}/
#   DESTINATION ${CATKIN_PACKAGE_INCLUDE_DESTINATION}
```

```
#   FILES_MATCHING PATTERN "*.h"
#   PATTERN ".svn" EXCLUDE
# )

## Mark other files for installation (e.g. launch and bag files, etc.)
# install(FILES
#   # myfile1
#   # myfile2
#   DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
# )

#############
## Testing ##
#############

## Add gtest based cpp test target and link libraries
# catkin_add_gtest(${PROJECT_NAME}-test test/test_elco_oursin.cpp)
# if(TARGET ${PROJECT_NAME}-test)
#   target_link_libraries(${PROJECT_NAME}-test ${PROJECT_NAME})
# endif()

## Add folders to be run by python nosetests
# catkin_add_nosetests(test)
```

Programa en C (para Eclipse):

moving.c

```
#include "fsm.h"
#include "main.h"
#include "moving.h"
#include "time.h"
#include "wiringPi.h"
#include <stdio.h>
#include <stdlib.h>
#include <softPwm.h>
#include <pthread.h>

//#include <windows.h>
//#include <mmsystem.h>
//#pragma comment(lib, "Winmm.lib")

//#include <vlc/vlc.h>
//#include <vlc/libvlc.h>
//#include <vlc/libvlc_media.h>
//#include <vlc/libvlc_media_player.h>
//#include "pthread.h"
/*
#include "libvlc_media_player.h"
#include "libvlc_media.h"
#include "libvlc.h"
*/

#define IN1         17      // Cable amarillo
#define IN2         27      // Cable naranja
#define IN3         23      // Cable verde
#define IN4         24      // Cable azul

//Sensor delante
#define Trigger1    18      //Trigger del sensor 1
#define Echo1       25      //Echo del sensor 1

//Sensor detr□s
#define Trigger2    8       //Trigger del sensor 2
#define Echo2       7       //Echo del sensor 2

//Sensor delante derecha
#define Trigger3    20      //Trigger del sensor 3
#define Echo3       21      //Echo del sensor 3

//Sensor delante izquierda
#define Trigger4    6       //Trigger del sensor 4
#define Echo4       26      //Echo del sensor 4
```

```
//Sensor detr�s derecha
#define Trigger5    5       //Trigger del sensor 5
#define Echo5       9      //Echo del sensor 5

//Sensor detr�s izquierda
#define Trigger6    11      //Trigger del sensor 6
#define Echo6       10      //Echo del sensor 6

// Definici�n de variables relacionadas con el sensor de distancia HC-
SR04
long t;             //tiempo que demora en llegar el echo
long d;             //distancia en centimetros

tipo_timers timer1;

//#define LED1      21      // Cable azul

// Para controlar la velocidad de los motores
#define ENA         13      // Cable morado
#define ENB         19      // Cable marron

//-------------------------------------------------------
// FUNCIONES DE TRANSICION DE LA MAQUINA DE ESTADOS
//-------------------------------------------------------

int CompruebaHaciaDelante (fsm_t* this) {
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_DELANTE);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int CompruebaHaciaAtras(fsm_t* this) {
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_ATRAS);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int CompruebaHaciaDerecha(fsm_t* this) {
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_DERECHA);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}
```

```
int CompruebaHaciaIzquierda(fsm_t* this) {
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_IZQUIERDA);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int CompruebaParar(fsm_t* this) {
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_PARAR);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int CompruebaTimeoutSensores(fsm_t* this) { // Todos los sensores
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_SENSORES);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int CompruebaTimeoutSensor1(fsm_t* this) { // Delante
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_SENSOR1);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int CompruebaTimeoutSensor2(fsm_t* this) { // Detr◆s
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_SENSOR2);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int CompruebaTimeoutSensor3(fsm_t* this) { // Delante derecha
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_SENSOR3);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}
```

```
int CompruebaTimeoutSensor4(fsm_t* this) { // Delante izquierda
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_SENSOR4);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int CompruebaTimeoutSensor5(fsm_t* this) { // Detr?s derecha
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_SENSOR5);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int CompruebaTimeoutSensor6(fsm_t* this) { // Detr?s izquierda
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_SENSOR6);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int Comprueba360DchaDelante(fsm_t* this) {
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_DCHA_DELANTE_360);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int Comprueba360IzqdaDelante(fsm_t* this) {
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_IZQDA_DELANTE_360);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}

int Comprueba360DchaAtras(fsm_t* this) {
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_DCHA_ATRAS_360);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}
```

```
int Comprueba360IzqdaAtras(fsm_t* this) {
    int result = 0;
    piLock(SYSTEM_FLAGS_KEY);
    result = (FLAG_IZQDA_ATRAS_360);
    piUnlock(SYSTEM_FLAGS_KEY);
    return result;
}
//-------------------------------------------------------
// FUNCIONES DE ACCION DE LA MAQUINA DE ESTADOS
//-------------------------------------------------------

void haciaDelante() {
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_DELANTE = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);

    //Hacia delante

    digitalWrite (IN1, LOW);
    digitalWrite (IN2, HIGH);
    softPwmWrite (ENA, 1000); //Velocidad motor A

    //Direcci□n motor B
    digitalWrite (IN3, HIGH);
    digitalWrite (IN4, LOW);
    softPwmWrite (ENB, 1000); //Velocidad motor B

    printf("\nHacia delante\n");
    fflush(stdout);
}

void haciaAtras(){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_ATRAS = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);

    // Hacia atras

    //Sentido motor A : inversi□n de los niveles HIGH y LOW que ten□am
os en la marche adelante
    //para que el motor cambie de sentido de rotaci□n
    digitalWrite (IN1, HIGH);
    digitalWrite (IN2, LOW);
    softPwmWrite (ENA, 1000); //Velocidad motor A

    //Sentido motor B : inversi□n de los niveles HIGH y LOW que ten□am
os en la marche adelante
```

```
    //para que el motor cambie de sentido de rotaci�n
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, HIGH);
    softPwmWrite (ENB, 1000); //Velocidad motor B

    printf("\nHacia atras\n");
    fflush(stdout);
}

void haciaDerecha(){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_DERECHA = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);

    //Hacia la derecha

    //Sentido motor A
    //Al mantener un pin LOW y el otro HIGH el motor gira en sentido co
ntrario al anterior
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, HIGH);
    softPwmWrite (ENA, 1000); //Velocidad motor A

    //Direccion motor B
    // Al mantener un pin LOW y el otro HIGH el motor gira en sentido c
ontrario al anterior
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, HIGH);
    softPwmWrite (ENB, 1000); //Velocidad motor B

    printf("\nHacia la derecha\n");
    fflush(stdout);
}

void haciaIzquierda(){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_IZQUIERDA = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);

    //Hacia la izquierda

    //Sentido motor A
    // Al mantener un pin HIGH y el otro LOW el motor gira en un sentid
o
    digitalWrite (IN1, HIGH);
    digitalWrite (IN2, LOW);
    softPwmWrite (ENA, 1000); //Velocidad motor A
```

```
    //Direccion motor B
    //Al mantener un pin HIGH y el otro LOW el motor gira en un sentido
    digitalWrite (IN3, HIGH);
    digitalWrite (IN4, LOW);
    softPwmWrite (ENB, 1000); //Velocidad motor B

    printf("\nHacia la izquierda\n");
    fflush(stdout);


}

void parar(){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_PARAR = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);

    //Parar

    //Direcci□n motor A
    digitalWrite (IN1, LOW);
    digitalWrite (IN2, LOW);
    //digitalWrite (ENA, 150); //Velocidad motor A

    //Direcci□n motor B
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, LOW);
    //digitalWrite (ENB, 150); //Velocidad motor B

    printf("\nParar\n");
    fflush(stdout);
}

void gira360DchaDelante(){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_DCHA_DELANTE_360 = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);

    digitalWrite (IN1, LOW);
    digitalWrite (IN2, HIGH);
    softPwmWrite (ENA, 1000); //Velocidad motor A

    //Direcci□n motor B
    digitalWrite (IN3, HIGH);
    digitalWrite (IN4, LOW);
    softPwmWrite (ENB, 100); //Velocidad motor B

    printf("\n360 Derecha Delante\n");
    fflush(stdout);
```

```
}

void gira360IzqdaDelante(){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_IZQDA_DELANTE_360 = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);

    //Hacia la izquierda

    //Sentido motor A

    digitalWrite (IN1, LOW);
    digitalWrite (IN2, HIGH);
    softPwmWrite (ENA, 100); //Velocidad motor A

    //Direcci□n motor B
    digitalWrite (IN3, HIGH);
    digitalWrite (IN4, LOW);
    softPwmWrite (ENB, 1000); //Velocidad motor B

    printf("\n360 Izquierda Delante\n");
    fflush(stdout);
}

void gira360DchaAtras(){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_DCHA_ATRAS_360 = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);

    digitalWrite (IN1, HIGH);
    digitalWrite (IN2, LOW);
    softPwmWrite (ENA, 1000); //Velocidad motor A

    //Direcci□n motor B
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, HIGH);
    softPwmWrite (ENB, 100); //Velocidad motor B

    printf("\n360 Derecha Atras\n");
    fflush(stdout);
}

void gira360IzqdaAtras(){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_IZQDA_ATRAS_360 = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);

    //Hacia la izquierda
```

```
    //Sentido motor A

    digitalWrite (IN1, HIGH);
    digitalWrite (IN2, LOW);
    softPwmWrite (ENA, 100); //Velocidad motor A

    //Direcci□n motor B
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, HIGH);
    softPwmWrite (ENB, 1000); //Velocidad motor B

    printf("\n360 Izquierda Atras\n");
    fflush(stdout);
}



void actualizaDistancia1(fsm_t* this){  //Sensor delante
    tipo_timers *t_timer;
    t_timer = (tipo_timers*)(this->user_data);
    //int cancion = 2;
    //ReproductorMusica *R = ReproductorMusicaNuevo(cancion);

    digitalWrite(Trigger1, HIGH);
    delayMicroseconds(10);          //Enviamos un pulso de 10us
    digitalWrite(Trigger1, LOW);

    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;

    //piLock(SYSTEM_FLAGS_KEY);
/*  FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;*/
    //piUnlock(SYSTEM_FLAGS_KEY);
```

```
    int echo1, previousEcho1, lowHigh1, highLow1;
    long startTime1, stopTime1, difference1;
    float rangeCm1;
    lowHigh1 = highLow1 = echo1 = previousEcho1 = 0;
    while(0 == lowHigh1 || highLow1 == 0) {
      previousEcho1 = echo1;
      echo1 = digitalRead(Echo1);
      if(0 == lowHigh1 && 0 == previousEcho1 && 1 == echo1) {
        lowHigh1 = 1;
        startTime1 = getMicrotime();
      }
      if(1 == lowHigh1 && 1 == previousEcho1 && 0 == echo1) {
        highLow1 = 1;
        stopTime1 = getMicrotime();
      }
    }
    difference1 = stopTime1 - startTime1;
    rangeCm1 = difference1 / 58;
    //printf("Start: %ld, stop: %ld, difference: %ld, range: %.2f cm\n"
, startTime, stopTime, difference, rangeCm);
    printf("Distancia Sensor 1: %.2f cm\n", rangeCm1);

/*    if(rangeCm1 <= MAX_RANGE_1 && rangeCm1 > MIN_RANGE_1) {
        setVelocidadENA(300);
        setVelocidadENB(300);
    } else*/ if(rangeCm1 <= MIN_RANGE_1) {
        FLAG_CHOQUE_SENSOR1 = 1;
        FLAG_PARAR = 0;
        //Parar

        //Direcci□n motor A
        digitalWrite (IN1, LOW);
        digitalWrite (IN2, LOW);
        //digitalWrite (ENA, 150); //Velocidad motor A

        //Direcci□n motor B
        digitalWrite (IN3, LOW);
        digitalWrite (IN4, LOW);
        //digitalWrite (ENB, 150); //Velocidad motor B

        printf("\nParar\n");
        fflush(stdout);

        //system("cvlc Delante.mp3");
        system("omxplayer -o alsa Delante.mp3");

        //playSound("C:/Users/Alfonso/Desktop/Audios_ELCO_Alarma_wav/De
recha.wav");
```

```
        //playSound(("Derecha.wav"));

        //PlaySound("C:\Users\Alfonso\Desktop\Audios_ELCO_Alarma_wav\De
lante.wav", NULL, SND_SYNC | SND_LOOP | SND_FILENAME);

        //sndPlaySound("C:\\Users\\Alfonso\\Desktop\\Audios_ELCO_Alarma
_wav\\Delante.wav", 0);

        //PlaySound("C:\Users\Alfonso\Desktop\Audios_ELCO_Alarma_wav\De
lante.wav", NULL, SND_LOOP);

        //system("C:\\Users\\Alfonso\\Desktop\\Audios_ELCO_Alarma_wav\\
Delante.wav");
        //reproduceMusica(&R);

    /*  //Lanzamos la hebra para el audio
        pthread_t thread1;
        pthread_create(&thread1, NULL, &reproduceMusica, R);
        pthread_join(thread1, NULL);*/

        //parar();
    } else {
        setVelocidadENA(1000);
        setVelocidadENB(1000);
    }

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor1), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor2), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor3), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor4), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor5), TIMEOUT_SENSOR1);
```

```
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor6), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);
}

void actualizaDistancia2(fsm_t* this){  //Sensor atr□s
    tipo_timers *t_timer;
    t_timer = (tipo_timers*)(this->user_data);
    //int cancion = 6;
    //ReproductorMusica *R = ReproductorMusicaNuevo(cancion);

    digitalWrite(Trigger2, HIGH);
    delayMicroseconds(10);          //Enviamos un pulso de 10us
    digitalWrite(Trigger2, LOW);

    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;

    //piLock(SYSTEM_FLAGS_KEY);
/*  FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;*/
    //piUnlock(SYSTEM_FLAGS_KEY);

    int echo2, previousEcho2, lowHigh2, highLow2;
    long startTime2, stopTime2, difference2;
    float rangeCm2;
    lowHigh2 = highLow2 = echo2 = previousEcho2 = 0;
    while(0 == lowHigh2 || highLow2 == 0) {
      previousEcho2 = echo2;
      echo2 = digitalRead(Echo2);
      if(0 == lowHigh2 && 0 == previousEcho2 && 1 == echo2) {
```

```
        lowHigh2 = 1;
        startTime2 = getMicrotime();
      }
      if(1 == lowHigh2 && 1 == previousEcho2 && 0 == echo2) {
        highLow2 = 1;
        stopTime2 = getMicrotime();
      }
    }
    difference2 = stopTime2 - startTime2;
    rangeCm2 = difference2 / 58;
    //printf("Start: %ld, stop: %ld, difference: %ld, range: %.2f cm\n"
, startTime, stopTime, difference, rangeCm);
    printf("Distancia Sensor 2: %.2f cm\n", rangeCm2);
/*    if(rangeCm2 <= MAX_RANGE_2 && rangeCm2 > MIN_RANGE_2) {
        setVelocidadENA(300);
        setVelocidadENB(300);
    } else */ if(rangeCm2 <= MIN_RANGE_2) {
        FLAG_CHOQUE_SENSOR2 = 1;
        FLAG_PARAR = 0;
        //Parar

        //Direcci□n motor A
        digitalWrite (IN1, LOW);
        digitalWrite (IN2, LOW);
        //digitalWrite (ENA, 150); //Velocidad motor A

        //Direcci□n motor B
        digitalWrite (IN3, LOW);
        digitalWrite (IN4, LOW);
        //digitalWrite (ENB, 150); //Velocidad motor B

        printf("\nParar\n");
        fflush(stdout);

        //reproduceMusica(&R);
        system("omxplayer -o alsa Detras.mp3");
/*
        //Lanzamos la hebra para el audio
        pthread_t thread1;
        pthread_create(&thread1, NULL, &reproduceMusica, R);
        pthread_join(thread1, NULL);
*/

        //parar();
    } else {
        setVelocidadENA(1000);
        setVelocidadENB(1000);
    }
```

```
    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor1), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor2), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor3), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor4), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor5), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor6), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);
}

void actualizaDistancia3(fsm_t* this){  //Sensor delante derecha
    tipo_timers *t_timer;
    t_timer = (tipo_timers*)(this->user_data);
    //int cancion = 0;
    //ReproductorMusica *R = ReproductorMusicaNuevo(cancion);

    digitalWrite(Trigger3, HIGH);
    delayMicroseconds(10);          //Enviamos un pulso de 10us
    digitalWrite(Trigger3, LOW);

    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
```

```
    //piLock(SYSTEM_FLAGS_KEY);
/*  FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;*/
    //piUnlock(SYSTEM_FLAGS_KEY);

    int echo3, previousEcho3, lowHigh3, highLow3;
    long startTime3, stopTime3, difference3;
    float rangeCm3;
    lowHigh3 = highLow3 = echo3 = previousEcho3 = 0;
    while(0 == lowHigh3 || highLow3 == 0) {
      previousEcho3 = echo3;
      echo3 = digitalRead(Echo3);
      if(0 == lowHigh3 && 0 == previousEcho3 && 1 == echo3) {
        lowHigh3 = 1;
        startTime3 = getMicrotime();
      }
      if(1 == lowHigh3 && 1 == previousEcho3 && 0 == echo3) {
        highLow3 = 1;
        stopTime3 = getMicrotime();
      }
    }
    difference3 = stopTime3 - startTime3;
    rangeCm3 = difference3 / 58;
    //printf("Start: %ld, stop: %ld, difference: %ld, range: %.2f cm\n"
, startTime, stopTime, difference, rangeCm);
    printf("Distancia Sensor 3: %.2f cm\n", rangeCm3);
/*    if(rangeCm3 <= MAX_RANGE_1 && rangeCm3 > MIN_RANGE_1) {
        setVelocidadENA(300);
        setVelocidadENB(300);
    } else*/ if(rangeCm3 <= MIN_RANGE_1) {
        FLAG_CHOQUE_SENSOR3 = 1;
        FLAG_PARAR = 0;
        //Parar

        //Direcci□n motor A
        digitalWrite (IN1, LOW);
        digitalWrite (IN2, LOW);
        //digitalWrite (ENA, 150); //Velocidad motor A
```

```
    //Direcci?n motor B
    digitalWrite (IN3, LOW);
    digitalWrite (IN4, LOW);
    //digitalWrite (ENB, 150); //Velocidad motor B

    printf("\nParar\n");
    fflush(stdout);

    //reproduceMusica(&R);
    system("omxplayer -o alsa Delante_derecha.mp3");

    /*//Lanzamos la hebra para el audio
    pthread_t thread1;
    pthread_create(&thread1, NULL, &reproduceMusica, R);
    pthread_join(thread1, NULL);*/

    //parar();
} else {
    setVelocidadENA(1000);
    setVelocidadENB(1000);
}

//piLock(SYSTEM_FLAGS_KEY);
tmr_startms((tmr_t*)(t_timer->tmr_sensor1), TIMEOUT_SENSOR1);
delay(5);
//piUnlock(SYSTEM_FLAGS_KEY);

//piLock(SYSTEM_FLAGS_KEY);
tmr_startms((tmr_t*)(t_timer->tmr_sensor2), TIMEOUT_SENSOR1);
delay(5);
//piUnlock(SYSTEM_FLAGS_KEY);

//piLock(SYSTEM_FLAGS_KEY);
tmr_startms((tmr_t*)(t_timer->tmr_sensor3), TIMEOUT_SENSOR1);
delay(5);
//piUnlock(SYSTEM_FLAGS_KEY);

//piLock(SYSTEM_FLAGS_KEY);
tmr_startms((tmr_t*)(t_timer->tmr_sensor4), TIMEOUT_SENSOR1);
delay(5);
//piUnlock(SYSTEM_FLAGS_KEY);

//piLock(SYSTEM_FLAGS_KEY);
tmr_startms((tmr_t*)(t_timer->tmr_sensor5), TIMEOUT_SENSOR1);
delay(5);
//piUnlock(SYSTEM_FLAGS_KEY);
```

```
    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor6), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);
}

void actualizaDistancia4(fsm_t* this){  //Sensor delante izquierda
    tipo_timers *t_timer;
    t_timer = (tipo_timers*)(this->user_data);
    //int cancion = 1;
    //ReproductorMusica *R = ReproductorMusicaNuevo(cancion);

    digitalWrite(Trigger4, HIGH);
    delayMicroseconds(10);          //Enviamos un pulso de 10us
    digitalWrite(Trigger4, LOW);

    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;

    //piLock(SYSTEM_FLAGS_KEY);
/*  FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;*/
    //piUnlock(SYSTEM_FLAGS_KEY);

    int echo4, previousEcho4, lowHigh4, highLow4;
    long startTime4, stopTime4, difference4;
    float rangeCm4;
    lowHigh4 = highLow4 = echo4 = previousEcho4 = 0;
    while(0 == lowHigh4 || highLow4 == 0) {
      previousEcho4 = echo4;
      echo4 = digitalRead(Echo4);
      if(0 == lowHigh4 && 0 == previousEcho4 && 1 == echo4) {
        lowHigh4 = 1;
        startTime4 = getMicrotime();
      }
```

```
      if(1 == lowHigh4 && 1 == previousEcho4 && 0 == echo4) {
        highLow4 = 1;
        stopTime4 = getMicrotime();
      }
    }
    difference4 = stopTime4 - startTime4;
    rangeCm4 = difference4 / 58;
    //printf("Start: %ld, stop: %ld, difference: %ld, range: %.2f cm\n"
, startTime, stopTime, difference, rangeCm);
    printf("Distancia Sensor 4: %.2f cm\n", rangeCm4);
/*    if(rangeCm4 <= MAX_RANGE_4 && rangeCm4 > MIN_RANGE_4) {
        setVelocidadENA(300);
        setVelocidadENB(300);
    } else*/ if(rangeCm4 <= MIN_RANGE_4) {
        FLAG_CHOQUE_SENSOR4 = 1;
        FLAG_PARAR = 0;
        //Parar

        //Direcci□n motor A
        digitalWrite (IN1, LOW);
        digitalWrite (IN2, LOW);
        //digitalWrite (ENA, 150); //Velocidad motor A

        //Direcci□n motor B
        digitalWrite (IN3, LOW);
        digitalWrite (IN4, LOW);
        //digitalWrite (ENB, 150); //Velocidad motor B

        printf("\nParar\n");
        fflush(stdout);

        //reproduceMusica(&R);
        system("omxplayer -o alsa Delante_izquierda.mp3");

/*      //Lanzamos la hebra para el audio
        pthread_t thread1;
        pthread_create(&thread1, NULL, &reproduceMusica, R);
        pthread_join(thread1, NULL);*/
        //parar();
    } else {
        setVelocidadENA(1000);
        setVelocidadENB(1000);
    }

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor1), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);
```

```
    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor2), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor3), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor4), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor5), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor6), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);
}

void actualizaDistancia5(fsm_t* this){  //Sensor detr□s derecha
    tipo_timers *t_timer;
    t_timer = (tipo_timers*)(this->user_data);
    //int cancion = 1;
    //ReproductorMusica *R = ReproductorMusicaNuevo(cancion);

    digitalWrite(Trigger5, HIGH);
    delayMicroseconds(10);          //Enviamos un pulso de 10us
    digitalWrite(Trigger5, LOW);

    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;

    //piLock(SYSTEM_FLAGS_KEY);
/*  FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
```

```
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;*/
    //piUnlock(SYSTEM_FLAGS_KEY);

    int echo5, previousEcho5, lowHigh5, highLow5;
    long startTime5, stopTime5, difference5;
    float rangeCm5;
    lowHigh5 = highLow5 = echo5 = previousEcho5 = 0;
    while(0 == lowHigh5 || highLow5 == 0) {
      previousEcho5 = echo5;
      echo5 = digitalRead(Echo5);
      if(0 == lowHigh5 && 0 == previousEcho5 && 1 == echo5) {
        lowHigh5 = 1;
        startTime5 = getMicrotime();
      }
      if(1 == lowHigh5 && 1 == previousEcho5 && 0 == echo5) {
        highLow5 = 1;
        stopTime5 = getMicrotime();
      }
    }
    difference5 = stopTime5 - startTime5;
    rangeCm5 = difference5 / 58;
    //printf("Start: %ld, stop: %ld, difference: %ld, range: %.2f cm\n"
, startTime, stopTime, difference, rangeCm);
    printf("Distancia Sensor 5: %.2f cm\n", rangeCm5);
/*    if(rangeCm5 <= MAX_RANGE_5 && rangeCm5 > MIN_RANGE_5) {
        setVelocidadENA(300);
        setVelocidadENB(300);
    } else*/ if(rangeCm5 <= MIN_RANGE_5) {
        FLAG_CHOQUE_SENSOR5 = 1;
        FLAG_PARAR = 0;
        //Parar

        //Direcci□n motor A
        digitalWrite (IN1, LOW);
        digitalWrite (IN2, LOW);
        //digitalWrite (ENA, 150); //Velocidad motor A

        //Direcci□n motor B
        digitalWrite (IN3, LOW);
        digitalWrite (IN4, LOW);
        //digitalWrite (ENB, 150); //Velocidad motor B
```

```
        printf("\nParar\n");
        fflush(stdout);

        //reproduceMusica(&R);
        system("omxplayer -o alsa Detras_derecha.mp3");

/*      //Lanzamos la hebra para el audio
        pthread_t thread1;
        pthread_create(&thread1, NULL, &reproduceMusica, R);
        pthread_join(thread1, NULL);*/
        //parar();
    } else {
        setVelocidadENA(1000);
        setVelocidadENB(1000);
    }

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor1), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor2), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor3), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor4), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor5), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor6), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);
}
```

```
void actualizaDistancia6(fsm_t* this){  //Sensor detr�s izquierda
    tipo_timers *t_timer;
    t_timer = (tipo_timers*)(this->user_data);
    //int cancion = 1;
    //ReproductorMusica *R = ReproductorMusicaNuevo(cancion);

    digitalWrite(Trigger6, HIGH);
    delayMicroseconds(10);          //Enviamos un pulso de 10us
    digitalWrite(Trigger6, LOW);

    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;

    //piLock(SYSTEM_FLAGS_KEY);
/*  FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;*/
    //piUnlock(SYSTEM_FLAGS_KEY);

    int echo6, previousEcho6, lowHigh6, highLow6;
    long startTime6, stopTime6, difference6;
    float rangeCm6;
    lowHigh6 = highLow6 = echo6 = previousEcho6 = 0;
    while(0 == lowHigh6 || highLow6 == 0) {
      previousEcho6 = echo6;
      echo6 = digitalRead(Echo6);
      if(0 == lowHigh6 && 0 == previousEcho6 && 1 == echo6) {
        lowHigh6 = 1;
        startTime6 = getMicrotime();
      }
      if(1 == lowHigh6 && 1 == previousEcho6 && 0 == echo6) {
        highLow6 = 1;
        stopTime6 = getMicrotime();
      }
    }
    difference6 = stopTime6 - startTime6;
```

```
    rangeCm6 = difference6 / 58;
    //printf("Start: %ld, stop: %ld, difference: %ld, range: %.2f cm\n"
, startTime, stopTime, difference, rangeCm);
    printf("Distancia Sensor 6: %.2f cm\n", rangeCm6);
/*    if(rangeCm6 <= MAX_RANGE_6 && rangeCm6 > MIN_RANGE_6) {
        setVelocidadENA(300);
        setVelocidadENB(300);
    } else*/ if(rangeCm6 <= MIN_RANGE_6) {
        FLAG_CHOQUE_SENSOR6 = 1;
        FLAG_PARAR = 0;
        //Parar

        //Direcci□n motor A
        digitalWrite (IN1, LOW);
        digitalWrite (IN2, LOW);
        //digitalWrite (ENA, 150); //Velocidad motor A

        //Direcci□n motor B
        digitalWrite (IN3, LOW);
        digitalWrite (IN4, LOW);
        //digitalWrite (ENB, 150); //Velocidad motor B

        printf("\nParar\n");
        fflush(stdout);

        //reproduceMusica(&R);
        system("omxplayer -o alsa Detras_izquierda.mp3");

/*      //Lanzamos la hebra para el audio
        pthread_t thread1;
        pthread_create(&thread1, NULL, &reproduceMusica, R);
        pthread_join(thread1, NULL);*/
        //parar();
    } else {
        setVelocidadENA(1000);
        setVelocidadENB(1000);
    }

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor1), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor2), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);
```

```
    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor3), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor4), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor5), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);

    //piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor6), TIMEOUT_SENSOR1);
    delay(5);
    //piUnlock(SYSTEM_FLAGS_KEY);
}

void timer_sensores (union sigval value){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_SENSORES = 1;
    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);
}

void timer_sensor1 (union sigval value){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_SENSOR1 = 1;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
```

```
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);
}

void timer_sensor2 (union sigval value){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 1;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);
}

void timer_sensor3 (union sigval value){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 1;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);
}

void timer_sensor4 (union sigval value){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 1;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
```

```
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);
}

void timer_sensor5 (union sigval value){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 1;
    FLAG_SENSOR6 = 0;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);
}

void timer_sensor6 (union sigval value){
    //piLock(SYSTEM_FLAGS_KEY);
    FLAG_SENSOR1 = 0;
    FLAG_SENSOR2 = 0;
    FLAG_SENSOR3 = 0;
    FLAG_SENSOR4 = 0;
    FLAG_SENSOR5 = 0;
    FLAG_SENSOR6 = 1;
    FLAG_DELANTE = 0;
    FLAG_IZQUIERDA = 0;
    FLAG_DERECHA = 0;
    FLAG_ATRAS = 0;
    FLAG_PARAR = 0;
    //piUnlock(SYSTEM_FLAGS_KEY);
}

long getMicrotime(){
    struct timeval currentTime;
    gettimeofday(&currentTime, NULL);

  return currentTime.tv_sec * (int)1e6 + currentTime.tv_usec;
}

void setVelocidadENA(int valor) {
    softPwmWrite (ENA, valor); //Velocidad motor B
}
```

```
void setVelocidadENB(int valor) {
    softPwmWrite (ENB, valor); //Velocidad motor B
}
```

moving.h

```
#ifndef MOVING_H_
#define MOVING_H_

#include "fsm.h"
#include "main.h"

int CompruebaHaciaDelante (fsm_t* this);

int CompruebaHaciaAtras(fsm_t* this);

int CompruebaHaciaDerecha(fsm_t* this);

int CompruebaHaciaIzquierda(fsm_t* this);

int CompruebaParar(fsm_t* this) ;

int CompruebaTimeoutSensores(fsm_t* this);

int CompruebaTimeoutSensor1(fsm_t* this);

int CompruebaTimeoutSensor2(fsm_t* this);

int CompruebaTimeoutSensor3(fsm_t* this);

int CompruebaTimeoutSensor4(fsm_t* this);

int CompruebaTimeoutSensor5(fsm_t* this);

int CompruebaTimeoutSensor6(fsm_t* this);

int Comprueba360DchaDelante(fsm_t* this);

int Comprueba360IzqdaDelante(fsm_t* this);

int Comprueba360DchaAtras(fsm_t* this);

int Comprueba360IzqdaAtras(fsm_t* this);

void haciaDelante ();

void haciaAtras ();

void haciaDerecha ();

void haciaIzquierda ();

void parar ();
```

```
void actualizaDistancia(fsm_t* this);

void actualizaDistancia1(fsm_t* this);

void actualizaDistancia2(fsm_t* this);

void actualizaDistancia3(fsm_t* this);

void actualizaDistancia4(fsm_t* this);

void actualizaDistancia5(fsm_t* this);

void actualizaDistancia6(fsm_t* this);

void gira360DchaDelante ();

void gira360IzqdaDelante ();

void gira360DchaAtras ();

void gira360IzqdaAtras ();

void timer_sensores (union sigval value);

void timer_sensor1 (union sigval value);

void timer_sensor2 (union sigval value);

void timer_sensor3 (union sigval value);

void timer_sensor4 (union sigval value);

void timer_sensor5 (union sigval value);

void timer_sensor6 (union sigval value);

long getMicrotime();

void setVelocidadENA (int valor);

void setVelocidadENB (int valor);

#endif /* MOVING_H_ */
```

main.c

```
/*
 * main.c
 *
 *  Created on: 17 de mar. de 2021
 *      Author: Alfonso
 */
#include "stdio.h"
#include "pthread.h"
#include "moving.h"
#include "kbhit.h"
#include <time.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <sys/time.h>
#include "wiringPi.h"
#include "main.h"
#include <softPwm.h>

/*
#include <SDL2/SDL_mixer.h>
#include <SDL2/SDL.h>
*/
#include<stdbool.h>// Booleanos
//#pragma comment(lib, "Winmm.lib")

//#include <windows.h>
//#include <mmsystem.h>

//#include WMPLib.dll;
/*
#include <conio/conio.h>
#include <fmod/fmod.h>
*/

/*
#include <vlc/vlc.h>
#include <vlc/libvlc.h>
#include <vlc/libvlc_media.h>
#include <vlc/libvlc_media_player.h>
*/

/*
#include "libvlc_media_player.h"
```

```
#include "libvlc_media.h"
#include "libvlc.h"
*/

#define LED1        16      // Cable del led
#define IN1         17      // Cable amarillo
#define IN2         27      // Cable naranja
#define IN3         23      // Cable verde
#define IN4         24      // Cable azul

//Sensor delante
#define Trigger1    18      //Trigger del sensor 1
#define Echo1       25      //Echo del sensor 1

//Sensor detras
#define Trigger2    8       //Trigger del sensor 2
#define Echo2       7       //Echo del sensor 2

//Sensor delante derecha
#define Trigger3    20      //Trigger del sensor 3
#define Echo3       21      //Echo del sensor 3

//Sensor delante izquierda
#define Trigger4    6       //Trigger del sensor 4
#define Echo4       26      //Echo del sensor 4

//Sensor detras derecha
#define Trigger5    5       //Trigger del sensor 5
#define Echo5       9      //Echo del sensor 5

//Sensor detras izquierda
#define Trigger6    11      //Trigger del sensor 6
#define Echo6       10      //Echo del sensor 6

// Definicion de variables relacionadas con el sensor de distancia HC-
SR04
long t;             //tiempo que demora en llegar el echo
long d;             //distancia en centimetros

// Para controlar la velocidad de los motores
#define ENA         13      // Cable gris
#define ENB         19      // Cable marron

tipo_timers timer1;

int Inicializar (tipo_timers *t_timer) {
    int result = 0;
```

```
    //r = system("ls -l");

    t_timer->tmr_sensor1 = tmr_new(timer_sensor1);
    t_timer->tmr_sensor2 = tmr_new(timer_sensor2);
    t_timer->tmr_sensor3 = tmr_new(timer_sensor3);
    t_timer->tmr_sensor4 = tmr_new(timer_sensor4);
    t_timer->tmr_sensor5 = tmr_new(timer_sensor5);
    t_timer->tmr_sensor6 = tmr_new(timer_sensor6);

    //t_timer->tmr_sensores = tmr_new(timer_sensores);

    // Lanzamos thread para exploracion del teclado convencional del PC
, comentar para no usar el teclado
    result = piThreadCreate(pi_Thread);

/*  tmr_startms((tmr_t*)(t_timer->tmr_sensor1), TIMEOUT_SENSOR1);
    //delay(10);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor2), TIMEOUT_SENSOR1);
    //delay(10);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor3), TIMEOUT_SENSOR1);
    //delay(10);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor4), TIMEOUT_SENSOR1);*/

/*    piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensores), TIMEOUT_SENSOR1);
    delay(10);
    piUnlock(SYSTEM_FLAGS_KEY);*/

    piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor1), TIMEOUT_SENSOR1);
    delay(10);
    piUnlock(SYSTEM_FLAGS_KEY);

    piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor2), TIMEOUT_SENSOR1);
    delay(10);
    piUnlock(SYSTEM_FLAGS_KEY);

    piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor3), TIMEOUT_SENSOR1);
    delay(10);
    piUnlock(SYSTEM_FLAGS_KEY);

    piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor4), TIMEOUT_SENSOR1);
    delay(10);
    piUnlock(SYSTEM_FLAGS_KEY);
```

```
    piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor5), TIMEOUT_SENSOR1);
    delay(10);
    piUnlock(SYSTEM_FLAGS_KEY);

    piLock(SYSTEM_FLAGS_KEY);
    tmr_startms((tmr_t*)(t_timer->tmr_sensor6), TIMEOUT_SENSOR1);
    delay(10);
    piUnlock(SYSTEM_FLAGS_KEY);

    return result;
}

void *pi_Thread (void *arg) {
    char teclaPulsada;
    //actualizaDistancia();
    while(1) {
        piLock (STD_IO_BUFFER_KEY);
        if(kbhit()) {
            teclaPulsada = kbread();
            switch(teclaPulsada) {
            case 'w':
                //piLock(SYSTEM_FLAGS_KEY);
/*              FLAG_DELANTE = 1;
                FLAG_IZQUIERDA = 0;
                FLAG_DERECHA = 0;
                FLAG_ATRAS = 0;
                FLAG_PARAR = 0;
                FLAG_SENSORES = 0;
                FLAG_SENSOR1 = 0;
                FLAG_SENSOR2 = 0;
                FLAG_SENSOR3 = 0;
                FLAG_SENSOR4 = 0;
                FLAG_SENSOR5 = 0;
                FLAG_SENSOR6 = 0;
                FLAG_IZQDA_DELANTE_360 = 0;
                FLAG_DCHA_DELANTE_360 = 0;
                FLAG_IZQDA_ATRAS_360 = 0;
                FLAG_DCHA_ATRAS_360 = 0;*/
                FLAG_DELANTE = 1;
                //haciaDelante();
                printf("\n");
                printf("\nDELANTE\n");
                fflush(stdout);
                //actualizaDistancia();
                //piUnlock(SYSTEM_FLAGS_KEY);
                break;
            case 'a':
```

```
                //piLock(SYSTEM_FLAGS_KEY);
/*              FLAG_DELANTE = 0;
                FLAG_IZQUIERDA = 1;
                FLAG_DERECHA = 0;
                FLAG_ATRAS = 0;
                FLAG_PARAR = 0;
                FLAG_SENSORES = 0;
                FLAG_SENSOR1 = 0;
                FLAG_SENSOR2 = 0;
                FLAG_SENSOR3 = 0;
                FLAG_SENSOR4 = 0;
                FLAG_SENSOR5 = 0;
                FLAG_SENSOR6 = 0;
                FLAG_IZQDA_DELANTE_360 = 0;
                FLAG_DCHA_DELANTE_360 = 0;
                FLAG_IZQDA_ATRAS_360 = 0;
                FLAG_DCHA_ATRAS_360 = 0;*/
                FLAG_IZQUIERDA = 1;
                //haciaIzquierda();
                printf("\nIZQUIERDA\n");
                fflush(stdout);
                //haciaIzquierda();
                //piUnlock(SYSTEM_FLAGS_KEY);
                break;
            case 'd':
                //piLock(SYSTEM_FLAGS_KEY);
/*              FLAG_DELANTE = 0;
                FLAG_IZQUIERDA = 0;
                FLAG_DERECHA = 1;
                FLAG_ATRAS = 0;
                FLAG_PARAR = 0;
                FLAG_SENSORES = 0;
                FLAG_SENSOR1 = 0;
                FLAG_SENSOR2 = 0;
                FLAG_SENSOR3 = 0;
                FLAG_SENSOR4 = 0;
                FLAG_SENSOR5 = 0;
                FLAG_SENSOR6 = 0;
                FLAG_IZQDA_DELANTE_360 = 0;
                FLAG_DCHA_DELANTE_360 = 0;
                FLAG_IZQDA_ATRAS_360 = 0;
                FLAG_DCHA_ATRAS_360 = 0;*/
                //haciaDerecha();
                FLAG_DERECHA = 1;
                printf("\nDERECHA\n");
                fflush(stdout);
                //haciaDerecha();
                //piUnlock(SYSTEM_FLAGS_KEY);
```

```
                break;
            case 's':
                //piLock(SYSTEM_FLAGS_KEY);
/*              FLAG_DELANTE = 0;
                FLAG_IZQUIERDA = 0;
                FLAG_DERECHA = 0;
                FLAG_ATRAS = 1;
                FLAG_PARAR = 0;
                FLAG_SENSORES = 0;
                FLAG_SENSOR1 = 0;
                FLAG_SENSOR2 = 0;
                FLAG_SENSOR3 = 0;
                FLAG_SENSOR4 = 0;
                FLAG_SENSOR5 = 0;
                FLAG_SENSOR6 = 0;
                FLAG_IZQDA_DELANTE_360 = 0;
                FLAG_DCHA_DELANTE_360 = 0;
                FLAG_IZQDA_ATRAS_360 = 0;
                FLAG_DCHA_ATRAS_360 = 0;*/
                FLAG_ATRAS = 1;
                //haciaAtras();
                printf("\nATRAS\n");
                fflush(stdout);
                //haciaAtras();
                //piUnlock(SYSTEM_FLAGS_KEY);
                break;
            case 'p':
                //piLock(SYSTEM_FLAGS_KEY);
/*              FLAG_DELANTE = 0;
                FLAG_IZQUIERDA = 0;
                FLAG_DERECHA = 0;
                FLAG_ATRAS = 0;
                FLAG_PARAR = 1;
                FLAG_SENSORES = 0;
                FLAG_SENSOR1 = 0;
                FLAG_SENSOR2 = 0;
                FLAG_SENSOR3 = 0;
                FLAG_SENSOR4 = 0;
                FLAG_SENSOR5 = 0;
                FLAG_SENSOR6 = 0;
                FLAG_IZQDA_DELANTE_360 = 0;
                FLAG_DCHA_DELANTE_360 = 0;
                FLAG_IZQDA_ATRAS_360 = 0;
                FLAG_DCHA_ATRAS_360 = 0;*/
                FLAG_PARAR = 1;
                //parar();
                printf("\nPARAR\n");
                fflush(stdout);
```

```
                //actualizaDistancia();
                //piUnlock(SYSTEM_FLAGS_KEY);
                break;
            case 'l':
                //piLock(SYSTEM_FLAGS_KEY);
                /*FLAG_DELANTE = 0;
                FLAG_IZQUIERDA = 0;
                FLAG_DERECHA = 0;
                FLAG_ATRAS = 0;
                FLAG_PARAR = 0;
                FLAG_SENSORES = 0;
                FLAG_SENSOR1 = 0;
                FLAG_SENSOR2 = 0;
                FLAG_SENSOR3 = 0;
                FLAG_SENSOR4 = 0;
                FLAG_SENSOR5 = 0;
                FLAG_SENSOR6 = 0;
                FLAG_IZQDA_DELANTE_360 = 1;
                FLAG_DCHA_DELANTE_360 = 0;
                FLAG_IZQDA_ATRAS_360 = 0;
                FLAG_DCHA_ATRAS_360 = 0;*/
                FLAG_IZQDA_DELANTE_360 = 1;
                //gira360IzqdaDelante();
                printf("\nIZQUIERDA DELANTE 360\n");
                fflush(stdout);
                //actualizaDistancia();
                //piUnlock(SYSTEM_FLAGS_KEY);
                break;
            case 'r':
                //piLock(SYSTEM_FLAGS_KEY);
/*              FLAG_DELANTE = 0;
                FLAG_IZQUIERDA = 0;
                FLAG_DERECHA = 0;
                FLAG_ATRAS = 0;
                FLAG_PARAR = 0;
                FLAG_SENSORES = 0;
                FLAG_SENSOR1 = 0;
                FLAG_SENSOR2 = 0;
                FLAG_SENSOR3 = 0;
                FLAG_SENSOR4 = 0;
                FLAG_SENSOR5 = 0;
                FLAG_SENSOR6 = 0;
                FLAG_IZQDA_DELANTE_360 = 0;
                FLAG_DCHA_DELANTE_360 = 1;
                FLAG_IZQDA_ATRAS_360 = 0;
                FLAG_DCHA_ATRAS_360 = 0;*/
                FLAG_DCHA_DELANTE_360 = 1;
                //parar();
```

```
                printf("\nDERECHA DELANTE 360\n");
                fflush(stdout);
                //gira360DchaDelante();
                //piUnlock(SYSTEM_FLAGS_KEY);
                break;
            case 'm':
                //piLock(SYSTEM_FLAGS_KEY);
/*              FLAG_DELANTE = 0;
                FLAG_IZQUIERDA = 0;
                FLAG_DERECHA = 0;
                FLAG_ATRAS = 0;
                FLAG_PARAR = 0;
                FLAG_SENSORES = 0;
                FLAG_SENSOR1 = 0;
                FLAG_SENSOR2 = 0;
                FLAG_SENSOR3 = 0;
                FLAG_SENSOR4 = 0;
                FLAG_SENSOR5 = 0;
                FLAG_SENSOR6 = 0;
                FLAG_IZQDA_DELANTE_360 = 0;
                FLAG_DCHA_DELANTE_360 = 0;
                FLAG_IZQDA_ATRAS_360 = 0;
                FLAG_DCHA_ATRAS_360 = 1;*/
                FLAG_DCHA_ATRAS_360 = 1;
                //parar();
                printf("\DERECHA ATRAS 360 360\n");
                fflush(stdout);
                //gira360DchaAtras();
                //piUnlock(SYSTEM_FLAGS_KEY);
                break;
            case 'n':
                //piLock(SYSTEM_FLAGS_KEY);
/*              FLAG_DELANTE = 0;
                FLAG_IZQUIERDA = 0;
                FLAG_DERECHA = 0;
                FLAG_ATRAS = 0;
                FLAG_PARAR = 0;
                FLAG_SENSORES = 0;
                FLAG_SENSOR1 = 0;
                FLAG_SENSOR2 = 0;
                FLAG_SENSOR3 = 0;
                FLAG_SENSOR4 = 0;
                FLAG_SENSOR5 = 0;
                FLAG_SENSOR6 = 0;
                FLAG_IZQDA_DELANTE_360 = 0;
                FLAG_DCHA_DELANTE_360 = 0;
                FLAG_IZQDA_ATRAS_360 = 1;
                FLAG_DCHA_ATRAS_360 = 0;*/
```

```
                FLAG_IZQDA_ATRAS_360 = 1;
                //parar();
                printf("\IZQUIERDA ATRAS 360\n");
                fflush(stdout);
                //gira360IzqdaAtras();
                //piUnlock(SYSTEM_FLAGS_KEY);
                break;
            case 'q':
                parar();
                exit(0);
                break;
            default:
                printf("\nTecla incorrecta\n");
                break;
            }
        }
        piUnlock (STD_IO_BUFFER_KEY);
    }
    pthread_exit(NULL);
}

void delay_until (unsigned int next) {
    unsigned int now = millis();
    if (next > now) {
        delay (next - now);
    }
}

/*ReproductorMusica *ReproductorMusicaNuevo(int cancion) {
    const char *music[8] = {
            "file:///C:/Users/Alfonso/Desktop/Audios_ELCO_Alarma/Delant
e derecha.mp3",
            "file:///C:/Users/Alfonso/Desktop/Audios_ELCO_Alarma/Delant
e izquierda.mp3",
            "file:///C:/Users/Alfonso/Desktop/Audios_ELCO_Alarma/Delant
e.mp3",
            "file:///C:/Users/Alfonso/Desktop/Audios_ELCO_Alarma/Derech
a.mp3",
            "file:///C:/Users/Alfonso/Desktop/Audios_ELCO_Alarma/Detr?s
 derecha.mp3",
            "file:///C:/Users/Alfonso/Desktop/Audios_ELCO_Alarma/Detr?s
 izquierda.mp3",
            "file:///C:/Users/Alfonso/Desktop/Audios_ELCO_Alarma/Detr?s
.mp3",
            "file:///C:/Users/Alfonso/Desktop/Audios_ELCO_Alarma/Izquie
rda.mp3"
    };
```

```
    ReproductorMusica *reproductor = (ReproductorMusica *) malloc(sizeo
f(ReproductorMusica));
    reproductor->instance = libvlc_new(0, NULL);
    reproductor->mediaPlayer = livlc_media_player_new(reproductor-
>instance);
    reproductor->media = libvlc_media_new_location(reproductor-
>instance,music[cancion]);

    return reproductor;
}

void *reproduceMusica (void *data) {
    ReproductorMusica *R = (ReproductorMusica *) data;
    libvlc_media_player_set_media(R->mediaPlayer, R->media);
    libvlc_media_parse(R->media);
    libvlc_media_player_play(R->mediaPlayer);
    return NULL;
}

void ReproductorMusicaLibera(ReproductorMusica *R) {
    libvlc_media_release(R->media);
    libvlc_media_player_release(R->mediaPlayer);
    libvlc_release(R->instance);
    free(R);
}*/

int main (void) {
    tipo_timers timer;
    unsigned int next;

     wiringPiSetup();

     if (wiringPiSetupGpio () <0){
            printf("\nUnable to setup wiringPi\n");
            fflush(stdout);
            //piUnlock(STD_IO_BUFFER_KEY);
            return -1;
        }

        pinMode(LED1, OUTPUT);
        pinMode(IN1, OUTPUT);
        pinMode(IN2, OUTPUT);
        pinMode(IN3, OUTPUT);
        pinMode(IN4, OUTPUT);
        pinMode(ENA, OUTPUT);
        pinMode(ENB, OUTPUT);
        softPwmCreate(ENA,1,1000);
        softPwmCreate(ENB,1,1000);
```

```
        pinMode(Trigger1, OUTPUT);
        pinMode(Trigger2, OUTPUT);
        pinMode(Trigger3, OUTPUT);
        pinMode(Trigger4, OUTPUT);
        pinMode(Trigger5, OUTPUT);
        pinMode(Trigger6, OUTPUT);
        pinMode(Echo1, INPUT);
        pinMode(Echo2, INPUT);
        pinMode(Echo3, INPUT);
        pinMode(Echo4, INPUT);
        pinMode(Echo5, INPUT);
        pinMode(Echo6, INPUT);
        digitalWrite(Trigger1, LOW);     //Inicializamos el trigger del
 sensor 1 a 0
        digitalWrite(Trigger2, LOW);     //Inicializamos el trigger del
 sensor 2 a 0
        digitalWrite(Trigger3, LOW);     //Inicializamos el trigger del
 sensor 3 a 0
        digitalWrite(Trigger4, LOW);     //Inicializamos el trigger del
 sensor 4 a 0
        digitalWrite(Trigger5, LOW);     //Inicializamos el trigger del
 sensor 5 a 0
        digitalWrite(Trigger6, LOW);     //Inicializamos el trigger del
 sensor 6 a 0

        //LED parpadeante
        printf("\nLED de inicializacion\n");
        fflush(stdout);

        int i = 0;
         for (i = 0;i<3;i++){
            digitalWrite (LED1, HIGH);
         delay(500);
            digitalWrite (LED1, LOW);
         delay(500);
         }

        parar();

/*
        digitalWrite(Trigger1, HIGH);
        delayMicroseconds(10);          //Enviamos un pulso de 10us
        digitalWrite(Trigger1, LOW);
        delayMicroseconds(10);
        digitalWrite(Trigger2, HIGH);
        delayMicroseconds(10);          //Enviamos un pulso de 10us
        digitalWrite(Trigger2, LOW);
        delayMicroseconds(10);
```

```
        digitalWrite(Trigger3, HIGH);
        delayMicroseconds(10);          //Enviamos un pulso de 10us
        digitalWrite(Trigger3, LOW);
        delayMicroseconds(10);
        digitalWrite(Trigger4, HIGH);
        delayMicroseconds(10);          //Enviamos un pulso de 10us
        digitalWrite(Trigger4, LOW);
        delayMicroseconds(10);
        digitalWrite(Trigger5, HIGH);
        delayMicroseconds(10);          //Enviamos un pulso de 10us
        digitalWrite(Trigger5, LOW);
        delayMicroseconds(10);
        digitalWrite(Trigger6, HIGH);
        delayMicroseconds(10);          //Enviamos un pulso de 10us
        digitalWrite(Trigger6, LOW);
        delayMicroseconds(10);
*/

/*      while (digitalRead(Echo1) == HIGH) {
            tmr_startms((tmr_t*)(timer.tmr_sensor1), TIMEOUT_SENSOR1);
            while (digitalRead(Echo1) == LOW) {

            }
        }*/
        //delay(100);                       //Hacemos una pausa de 100m
s

        // Definimos las transiciones

        static fsm_trans_t sensor[] = {
                //{ESTADOSENSOR, CompruebaTimeoutSensores, ESTADOSENSOR
, actualizaDistancia },
                {ESTADOSENSOR, CompruebaTimeoutSensor1, ESTADOSENSOR, a
ctualizaDistancia1 },
                {ESTADOSENSOR, CompruebaTimeoutSensor2, ESTADOSENSOR, a
ctualizaDistancia2 },
                {ESTADOSENSOR, CompruebaTimeoutSensor3, ESTADOSENSOR, a
ctualizaDistancia3 },
                {ESTADOSENSOR, CompruebaTimeoutSensor4, ESTADOSENSOR, a
ctualizaDistancia4 },
                {ESTADOSENSOR, CompruebaTimeoutSensor5, ESTADOSENSOR, a
ctualizaDistancia5 },
                {ESTADOSENSOR, CompruebaTimeoutSensor6, ESTADOSENSOR, a
ctualizaDistancia6 },
                {-1, NULL, -1, NULL },
        };


        static fsm_trans_t movimiento[] = {
```

```
            { ESTADO1, CompruebaHaciaDelante, ESTADO1, haciaDelante },
            { ESTADO1, CompruebaHaciaAtras, ESTADO1, haciaAtras },
            { ESTADO1, CompruebaHaciaDerecha, ESTADO1, haciaDerecha },
            { ESTADO1, CompruebaHaciaIzquierda, ESTADO1, haciaIzquierda
 },
            { ESTADO1, CompruebaParar, ESTADO1, parar },
            { ESTADO1, Comprueba360DchaDelante, ESTADO1, gira360DchaDel
ante },
            { ESTADO1, Comprueba360IzqdaDelante, ESTADO1, gira360IzqdaD
elante },
            { ESTADO1, Comprueba360DchaAtras, ESTADO1, gira360DchaAtras
 },
            { ESTADO1, Comprueba360IzqdaAtras, ESTADO1, gira360IzqdaAtr
as },
            {-1, NULL, -1, NULL },
        };

        Inicializar(&timer);

        fsm_t* movimiento_fsm = fsm_new (ESTADO1, movimiento, &timer);

        fsm_t* sensor_fsm = fsm_new (ESTADOSENSOR, sensor, &timer);

        next = millis();

        while (1) {
            fsm_fire (movimiento_fsm);
            fsm_fire (sensor_fsm);
            next += CLK_MS;
            delay_until (next);
        }
        fsm_destroy (movimiento_fsm); //Creemos que no hay que ponerlo
        fsm_destroy (sensor_fsm); //Creemos que no hay que ponerlo

        return 0;
}
```

main.h

```
/*
 * main.h
 *
 *  Created on: 17 de mar. de 2021
 *      Author: Alfonso
 */

#ifndef MAIN_H_
#define MAIN_H_

#include "fsm.h"
#include "tmr.h"
#include <time.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <sys/time.h>
//#include <vlc/vlc.h>
//#include <vlc/libvlc.h>
//#include <vlc/libvlc_media.h>
//#include <vlc/libvlc_media_player.h>
//#pragma comment(lib, "winmm.lib")
/*
#include <windows/windows.h>
#include <_mingw/_mingw.h>
#include <mmsystem.h>
#pragma comment(lib, "Winmm.lib")
*/

#define CLK_MS                  1

#define TIMEOUT_SENSOR1         150
#define TIMEOUT_SENSOR2         200

#define MIN_RANGE_1             20
#define MIN_RANGE_2             20
#define MIN_RANGE_3             20
#define MIN_RANGE_4             20
#define MIN_RANGE_5             20
#define MIN_RANGE_6             20

#define MAX_RANGE_1             50
#define MAX_RANGE_2             50
```

```
#define MAX_RANGE_3             50
#define MAX_RANGE_4             50
#define MAX_RANGE_5             50
#define MAX_RANGE_6             50

int FLAG_DELANTE;
int FLAG_ATRAS;
int FLAG_DERECHA;
int FLAG_IZQUIERDA;
int FLAG_DCHA_DELANTE_360;
int FLAG_IZQDA_DELANTE_360;
int FLAG_DCHA_ATRAS_360;
int FLAG_IZQDA_ATRAS_360;
int FLAG_PARAR;
int FLAG_SENSORES;
int FLAG_SENSOR1;
int FLAG_SENSOR2;
int FLAG_SENSOR3;
int FLAG_SENSOR4;
int FLAG_SENSOR5;
int FLAG_SENSOR6;
int FLAG_CHOQUE_SENSOR1;
int FLAG_CHOQUE_SENSOR2;
int FLAG_CHOQUE_SENSOR3;
int FLAG_CHOQUE_SENSOR4;
int FLAG_CHOQUE_SENSOR5;
int FLAG_CHOQUE_SENSOR6;

#define KEYBOARD_KEY        0
#define SYSTEM_FLAGS_KEY    1
#define STD_IO_BUFFER_KEY   2

typedef struct tipo_timers {
    tmr_t* tmr_sensor1;
    tmr_t* tmr_sensor2;
    tmr_t* tmr_sensor3;
    tmr_t* tmr_sensor4;
    tmr_t* tmr_sensor5;
    tmr_t* tmr_sensor6;
    tmr_t* tmr_sensores;
} tipo_timers;

/*
typedef struct ReproductorMusica ReproductorMusica;

struct ReproductorMusica {
    libvlc_instance_t* instance; //Instancia
    libvlc_media_player_t* mediaPlayer; //Reproductor de audios
```

```
    libvlc_media_t* media; //Archivo de audios
};
*/

enum fsm_state {
    ESTADO1,
/*  ESTADO2,
    ESTADO3,
    ESTADO4,
    ESTADO5,*/
    };

enum fsm_sensor {
    ESTADOSENSOR,
    };

int Inicializar (tipo_timers *t_timer);

void *pi_Thread (void *arg);

/*ReproductorMusica *ReproductorMusicaNuevo(int cancion);

void *reproduceMusica (void *data);

void ReproductorMusicaLibera(ReproductorMusica *R);

int playSound( char *filename );*/

int main ();

#endif /* MAIN_H_ */
```

Además, hemos utilizado: fsm.c, fsm.h, dprintf.h, kbhit.c, kbhit.h, tmr.c, tmr.h y pthread.h DISPONIBLES EN INTERNET