Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo de un prototipo de vehículo con asistencia al estacionamiento

Autor

Javier Roche Agudo

Director

Daniel Mercado Barraqueta

EINA

2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Javier Roche Agudo,

con nº de DNI 76918257R en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Desarrollo de un prototipo de vehículo con asistencia al estacionamiento,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 24 de Noviembre de 2017

Fdo: Javier Roche Agudo

ÍNDICE DE CONTENIDO y ANEXOS

1. INTRODUCCIÓN

1.1. Objeto del proyecto

El objetivo principal del proyecto es que un vehículo radiocontrol pueda estacionar tanto en cordón como en batería en función de las ordenes que le sean enviadas a través de un dispositivo móvil (una aplicación de móvil). Se ha de tener en cuenta el entorno del lugar de estacionamiento a la hora de programar el proceso, por lo que todo lo obtenido será teórico y no experimental, ya que no es posible simular todos los escenarios. También destacar que para los dos tipos de estacionamiento el vehículo ha de buscar el sitio por sí mismo a partir de un punto con una referencia cercana. En este aspecto reside la mayor complejidad del proyecto que se irá analizando y resolviendo.

En este proyecto se indaga un poco más sobre este tema, mostrando los cálculos, valores y programación con los que trabajan los vehículos de estacionamiento automático. Se aplicarán los valores obtenidos en la teoría para observar las notables diferencias con la realidad y, de esta manera, intentar predecir los posibles sucesos y actuar en consecuencia. Para ello se ha utilizado un vehículo radiocontrol de proporciones dimensionales similares a las de un vehículo real, aunque su comportamiento no se asemejará en algunos aspectos.

1.2. Alcance del proyecto

Para poder cumplir con los requisitos previos se ha realizado un estudio del vehículo radiocontrol tanto geométrica como electrónicamente. También se han realizado algunas simplificaciones en su geometría para facilitar los cálculos y programación posteriores.

Se han analizado todas las maniobras de estacionamiento posibles que deberá realizar y se han tenido en cuenta los obstáculos que puedan aparecer en el entorno del vehículo a lo largo del proceso. Por otra parte, se han tenido en cuenta las dificultades y limitaciones que pueden hallarse al trabajar con los sensores utilizados.

Finalmente, todo el código realizado, junto con los cálculos, se han implementado en un controlador de la familia Arduino conectado al prototipo, de modo que se puedan ejecutar todas las maniobras, enviando y recibiendo señales de los sensores situados en el contorno del vehículo.

1.3. Motivación

Debido a la actual necesidad de ahorrar el máximo tiempo en las acciones cotidianas y el gran interés que suscita el estacionamiento autónomo en el mundo de la automoción, se ha considerado realizar un proyecto enfocado a este tema.

1.4. Antecedentes

La aparición de nuevas tecnologías y su implantación en el sector automovilístico ha hecho que cada vez sea menos complejo el manejo de un vehículo. La conducción autónoma de un vehículo está siendo objeto de discusión entre diversos grupos sociales, ya que un vehículo no puede tomar las decisiones de un ser humano. Al margen de esto, en el mercado existen vehículos con asistencia en la conducción y estacionamiento mediante sensores, siempre y cuando el conductor esté presente en el interior del vehículo (en unos casos, el vehículo realiza todo automáticamente, en otros, solo da indicaciones de la posición y maniobras que debe llevar a cabo el conductor).
Actualmente también existen prototipos de vehículos que aparcan automáticamente dentro de un recinto mediante una señal telefónica. Ampliar los límites del estacionamiento de dicho prototipo será el siguiente objetivo que se buscará de tal manera que, en cualquier aparcamiento, tanto exterior como interior sea posible realizar el estacionamiento autónomo .[*1]

1.5. Programación del estacionamiento

Existen dos tipos de estacionamiento en función de la disposición de los vehículos: en cordón y en batería. Las diferencias entre ambos se encuentran principalmente en las distancias de separación y avance, y en las maniobras que se deben realizar en la última fase.

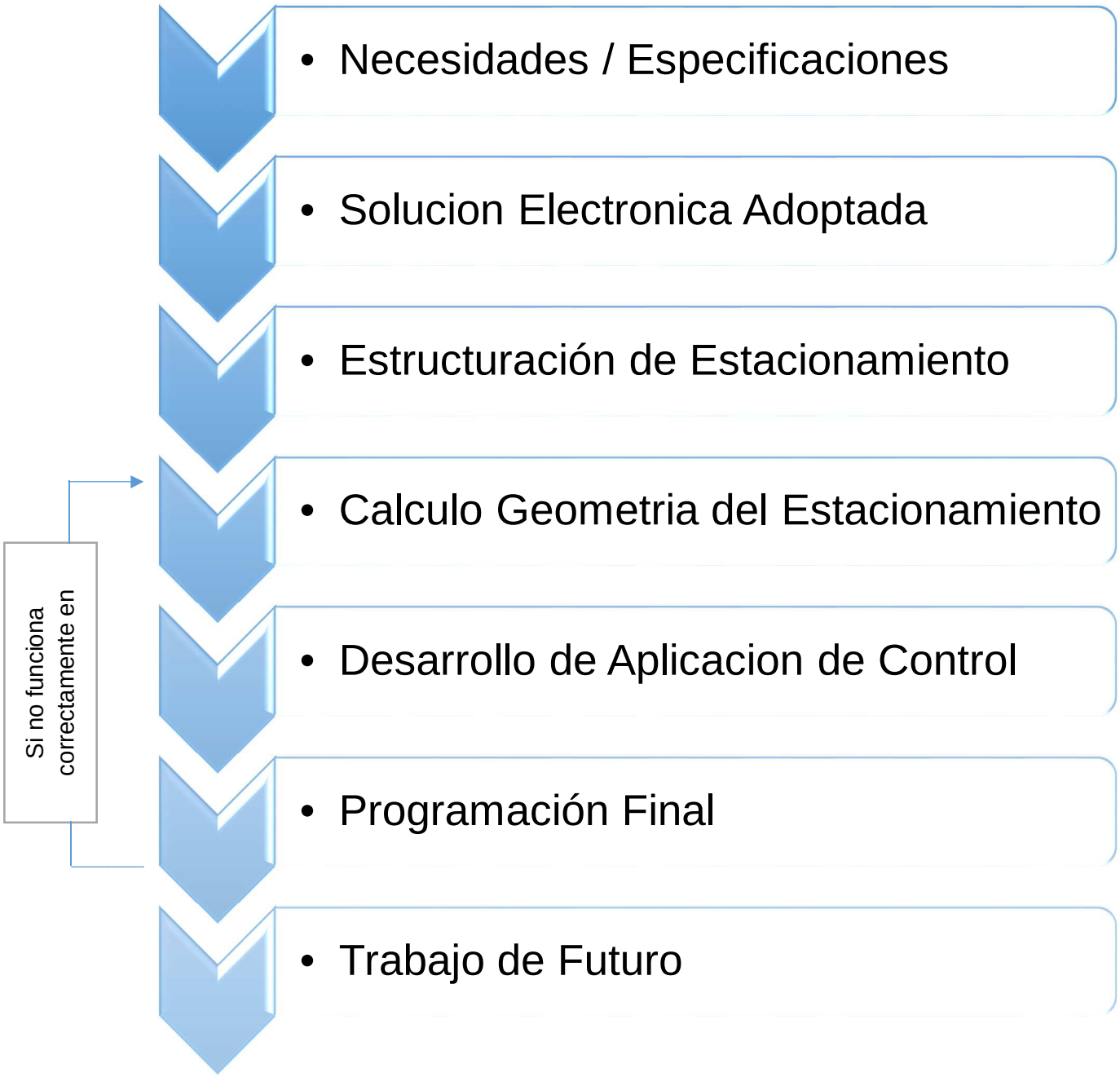
A continuación, se presenta la estructura general para ambos estacionamientos:

1- Posicionamiento inicial, el cual se divide en la rectificación de la guiñada y el desplazamiento lateral para situarse en el lugar deseado.
2- Búsqueda de hueco para estacionar. El vehículo avanza teniendo en cuenta la posible desviación de guiñada y la aparición de obstáculos repentinos.
3- Comprobación del tamaño del hueco. Se compara la longitud del sitio que está analizando con la longitud mínima impuesta para estacionar.
4- Posicionamiento para estacionar. El vehículo, al haberse desplazado lateralmente instantes previos, únicamente debe avanzar hasta la posición correcta para realizar la maniobra.
5- Maniobras de estacionamiento. Una vez posicionado, se inician las maniobras necesarias para realizar el estacionamiento.

Una vez definida la estructura a seguir, se pueden analizar los aspectos mecánicos y geométricos, y realizar el código en C++.

2. ESQUEMA GENERAL

La metodología del proyecto se puede representar mediante el siguiente diagrama de flujo:



En el diagrama aparece cada una de las fases estructuradas de manera que, sin obtener los datos de una fase, no es posible seguir avanzando. También se ha añadido una realimentación tras la fase de programación, ya que, en el caso de aparecer un error en pruebas, se ha de volver a medir y calcular los nuevos parámetros de entrada del programa.

En primer lugar, es necesario conocer las necesidades o especificaciones del proyecto, lo que se pide que haga el vehículo, ya que a partir de estos requisitos se planteará una solución.

Una vez obtenidos los criterios, se divide el proyecto en los respectivos campos de estudio (matemática, para la geometría; electrónica, interfase entre la programación y la mecánica; e informática, para la programación del código).

Tras esto, se inicia el estudio en el campo de la electrónica, con el fin de encontrar una solución que cumpla las especificaciones previamente definidas.

Después, se ha de realizar la estructuración del estacionamiento, ya que será la base del proyecto a partir de la cual se harán suposiciones y cálculos de las variables. Una vez realizada, se pueden realizar las simulaciones para el estacionamiento en cordón y en batería con sus cálculos correspondientes.

Dado que el programa no funcionara sin una señal de inicio, se ha de crear una aplicación que mediante señales inalámbricas (Bluetooth en este caso) envíe la información necesaria para la activación del prototipo.

Finalmente, se realizan las pruebas del programa en el prototipo y se analizan los resultados obtenidos. Si no son los esperados o no se encuentran dentro de los rangos de error definidos, se recalculan las variables necesarias o se cambian tolerancias hasta que funcione de la forma esperada.

3. NECESIDADES Y ESPECIFICACIONES DE DISEÑO

El objetivo principal, como se ha especificado anteriormente, es crear un prototipo que pueda:

- ➔ Realizar todas las fases de estacionamiento sin importar donde se encuentre el lugar donde debe aparcar.
- ➔ Detenerse al detectar un obstáculo.

Como imposición se pondrá que el lugar de estacionamiento será recto y de ancho suficiente como para poder realizar las maniobras oportunas.

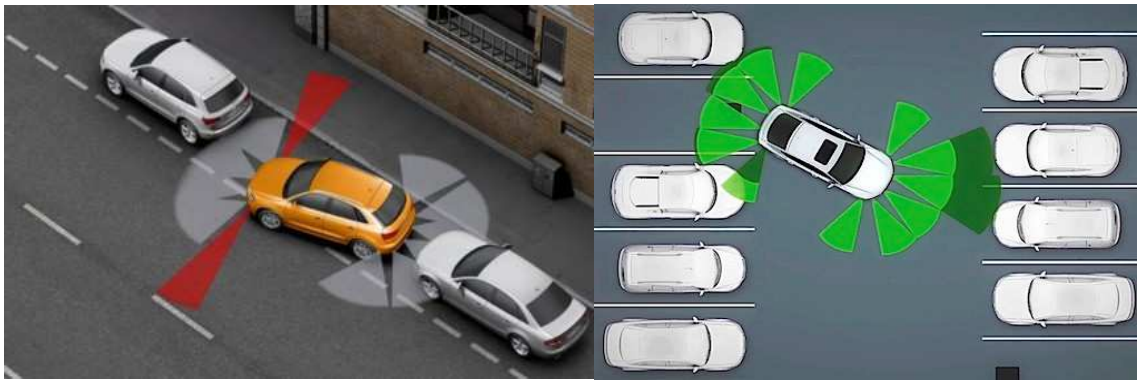


Fig. 1

En el *estacionamiento en cordón*, el vehículo deberá avanzar hasta encontrar un hueco donde estacionar, siempre alerta a cualquier obstáculo.

Una vez encontrado, la maniobra que realizará dependiendo del tamaño será:

- No aparcar (si no es lo suficientemente grande).
- Realizar la cantidad de maniobras necesarias hasta colocar el vehículo alineado con los situados delante y detrás suyo.
- Ejecutar el estacionamiento mediante una sola maniobra programada.

En el *estacionamiento en batería,* el recorrido principal será el mismo que en cordón salvo la fase de maniobras. En este caso serán:

- No aparcar (si no es lo suficientemente grande).
- Ejecutar el estacionamiento mediante una sola maniobra programada. Es decir, en este caso no hay un tamaño intermedio, sino que el vehículo puede aparcar o no puede.

A partir de estas especificaciones, se han de elegir las variables oportunas y realizar las suposiciones necesarias para que el prototipo estacione como se verá más adelante.

4. SOLUCION ELECTRONICA ADOPTADA.

A nivel de electrónica el problema consiste en mover, sensorizar y controlar un prototipo de vehículo eléctrico, constituido por la mecánica de un coche radiocontrol.

El movimiento del sistema se produce por medio del motor eléctrico original del vehículo y para su activación se decide utilizar la electrónica de potencia propio del sistema. Del mismo modo, la actuación del sistema de dirección se realiza con los elementos originales.

Se considera que la sensorización mínima necesaria está constituida por: seis sensores de distancia por ultrasonidos, distribuidos por el chasis del vehículo (en las cuatro esquinas y en centro de cada paragolpes), de manera que permitan detectar obstáculos en las proximidades, y una brújula que permita determinar la orientación del vehículo.

Para el sistema lógico de control se opta por un microcontrolador de la familia Arduino, en particular el Arduino Nano.

- *Electrónica del vehículo*

La electrónica inicial del vehículo empleado como base del prototipo se compone de: una fuente de alimentación de 9 V, dos puentes H (uno para el motor de tracción y otro para el de dirección), un receptor de señales inalámbricas (ondas radio) y un microprocesador (analiza las señales recibidas y envía las correspondientes a los puentes H para el correcto funcionamiento del conjunto).

Para que todos los elementos funcionen debidamente, es necesario adaptar toda la electrónica implantada a la inicial eliminando los componentes innecesarios como el receptor de señales, que será sustituido por un receptor Bluetooth, y el microcontrolador, ya que se va a trabajar con un Arduino en su lugar. A continuación, se muestra el esquema de la electrónica inicial del vehículo donde se pueden apreciar todos los bloques mencionados anteriormente:

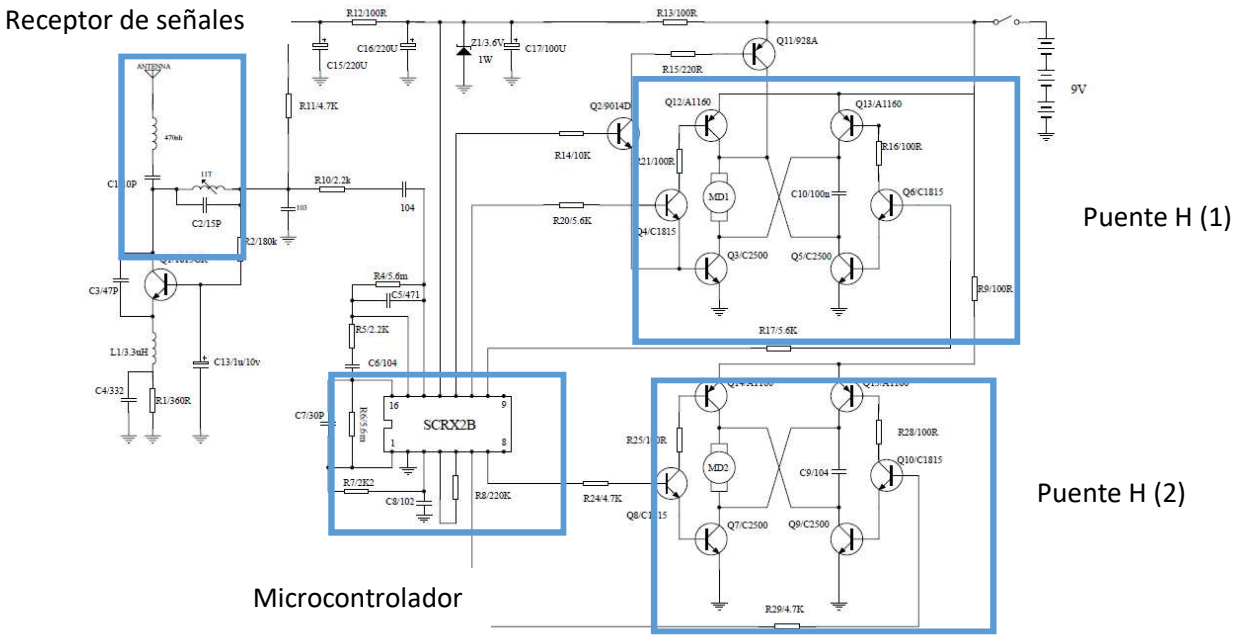


Fig. 2

- ***Componentes del conjunto electrónico***

Para poder implementar la programación del estacionamiento a la realidad, es necesario de un conjunto formado por diversos componentes electrónicos que funcionen mediante señales analógicas o digitales. Por ello, la base de todos los elementos será una placa Arduino Nano, en este caso, a la que se unirán mediante cableado el resto de componentes electrónicos que se explican a continuación:

- **Arduino Nano**

El dispositivo principal utilizado, para el control de todos los elementos del conjunto, es un Arduino Nano, a través del cual, mediante la conexión con los diferentes pines de E/S, se envía y recibe información de otros dispositivos, ya sea a través de señales digitales o analógicas.

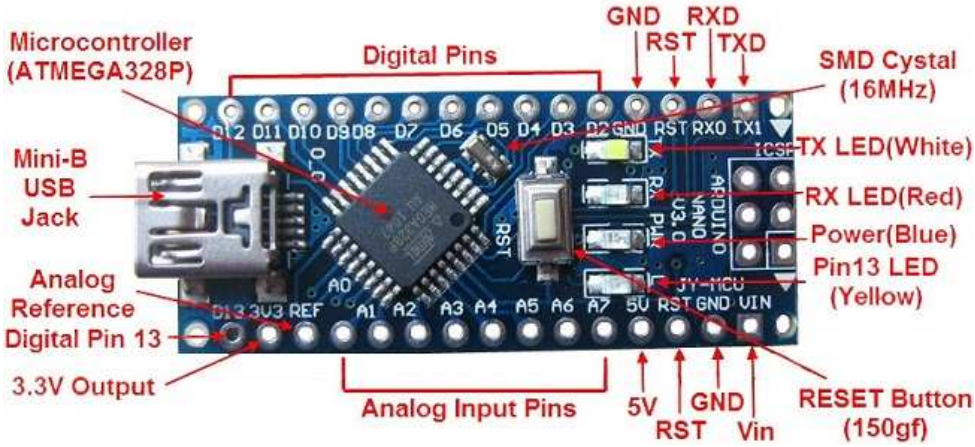


**Fig. 3: Imagen obtenida de https://wiki.eprolabs.com – Arduino Nano*

- **Multiplexor**

Este elemento se hace necesario para compatibilizar el uso del Arduino Nano con el resto de elementos, en particular con los seis detectores de ultrasonidos, reduce el número de pines de conexión necesarios. Este dispositivo se puede definir como un selector de canales que utiliza un código binario para realizar la selección del pin correspondiente.

Se compone de dieciséis pines de entrada/salida, donde se conectan los sensores de ultrasonidos; cuatro pines de entrada, donde se introducirá el código binario para la selección del canal de trabajo; dos pines de alimentación (Vcc) y neutro (GND); un pin de entrada/salida, donde circula la información correspondiente la señal de trabajo; y un pin de inhabilitación, para bloquear el dispositivo si fuera necesario.

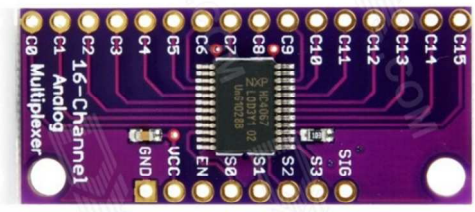


Fig. 4

- **Brújula *digital* GY − 273.*[2]**

La mayoría de maniobras se realizan mediante el cálculo de la posición angular instantánea, por lo que, para la obtención de esta, se ha implantado una brújula digital HMC5883. Este dispositivo mide el valor del campo magnético en tres ejes, de forma que se obtiene su posición respecto al campo magnético de la tierra.

La comunicación de este módulo con el resto del circuito es una comunicación I2C.

La conexión del elemento viene dada por 4 pines: alimentación Vcc, neutro GND y 2 pines de salida SDA y SCL, que son por donde se envía la información.

**Los cálculos para el análisis de datos de la brújula se presentan en el ANEXO 4*

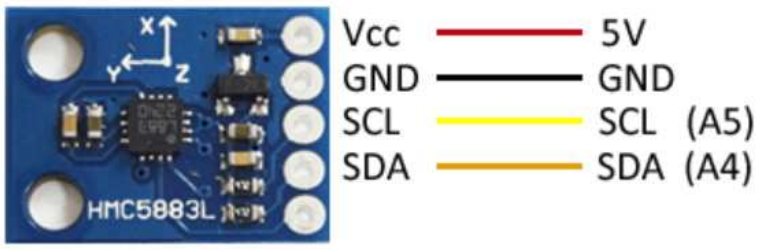


Fig. 5

- **Sensores de ultrasonidos HC-SR04.**

El vehículo necesita detectar los objetos de su entorno y su posición, por lo que se han añadido seis sensores de ultrasonidos para obtener dicha información. Este tipo de sensor detecta las distancias mediante la medición del tiempo que transcurre entre la señal de salida y la de entrada. El rango de funcionabilidad del sensor en distancias se sitúa entre dos centímetros y cuatro metros, de modo que según la fórmula utilizada para analizar las distancias:

$$Distancia = Tiempo_{Echo} * vel.sonido\ (340\frac{m}{s})/2$$

el rango de tiempos de funcionabilidad será 120 µs - 23530 µs.

Fig. 6

Este dispositivo se compone de cuatro pines: alimentación (Vcc de 5 V), neutro (GND), lanzador de pulsos (TRIGGER) y receptor de pulsos (ECHO). Los dos últimos están conectados al multiplexor de forma que a través de un pin (SIGNAL del multiplexor) se pueda acceder al TRIGGER o ECHO de cualquier sensor de ultrasonidos.

- *Correcciones para posibles fallos de medida*.

El problema de trabajar con varios sensores que emiten señales ultrasónicas, es principalmente la interferencia entre ellos. Las ondas emitidas por un sensor de ultrasonidos pueden ser captadas por otro distinto, realizando, de esta manera, una medida errónea. Para solventar este hecho se han implantado dos medidas:

- Filtrar los tiempos medidos en la función a través de un bucle condicionado, de tal manera que no puedan tener valor nulo o mayor al máximo rango de medida (23530 µs).

- Esperar un tiempo mínimo de veinte milisegundos entre dos llamadas consecutivas a la función *Lee_ultrasonidos*. De esta forma, se asegura que las ondas emitidas por el primero no afectaran al segundo sensor a pesar de ralentizar el proceso en curso.

- **Puentes H**

El sentido de la alimentación de los motores define el sentido de giro de estos y los puentes en H son los subconjuntos que, controlados por una señal PWM, realizan la conexión de alimentación de los motores, determinando la dirección de giro y la velocidad. Para el avance y retroceso del vehículo, los pines utilizados son D9 y D10, mientras que para el giro a izquierda y derecha los pines empleados son D7 y D8. Cada puente en H posee dos entradas dependiendo del sentido que se desea activar.

- **Modulo bluetooth.**

Se ha optado por un módulo HC04 que permite ser configurado tanto en modo maestro como esclavo. En esta aplicación funcionara en modo esclavo dejando al dispositivo móvil como maestro.

Este módulo es un conversor de comunicación serie a bluetooth de manera que todo el funcionamiento del bluetooth es transparente al programa. Para su conexión necesita de cinco pines correspondientes a dos de alimentación, dos de transmisión y uno de inhabilitación.

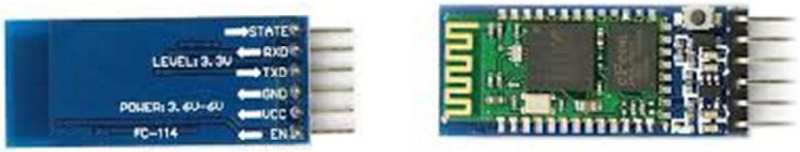


Fig. 7

- ***Justificación de la introducción del multiplexor.***

Las dimensiones del vehículo no permiten instalar una electrónica de gran tamaño, por lo que el conjunto instalado ha de comprimirse lo máximo posible. Los elementos que crean mayores problemas de espacio son los sensores de ultrasonidos, tanto en la protoboard como en el microcontrolador.

El objetivo es poder utilizar una placa microcontrolador de pequeño tamaño (Arduino Nano) y conectar, a su vez, seis sensores de ultrasonidos. Para ello se ha implantado un elemento intermediario entre ellos (multiplexor) que utiliza solamente cinco pines de E/S de Arduino, uno para la gestión de las señales de Trigger y Echo de los sensores y cuatro pines para la selección del sensor. De esta manera, mediante el envío de una señal simultánea a través de dichos pines, se accede a las conexiones de cualquiera de los seis sensores instalados y, una vez se tiene acceso a este, se mandan o reciben señales en función si se está conectando con el Trigger o Echo.

- ***Señales empleadas***

  - *Sensores de Ultrasonidos*

    El número de señales digitales enviadas y recibidas de los sensores de ultrasonidos desde el punto de vista de Arduino son dos (Trigger y Echo). La señal de salida (Trigger) desde Arduino es básicamente un tren de pulsos, activada a través del Arduino mediante una señal de 10 microsegundos, y la señal de entrada (Echo) que un pulso de duración igual al tiempo transcurrido entre la emisión de un tren de ultrasonidos y el eco de este reflejado en un obstáculo, y que consigue llegar al sensor mientras éste sigue activo en modo recepción. Ambas señales se administran desde el mismo pin de Arduino (D2).

  - *Brújula*

    El envío de información entre la brújula y Arduino se realiza a través de los pines A4 y A6 mediante comunicación I2C.

  - *Multiplexor*

    Las señales de entrada y salida de tipo digital correspondientes al multiplexor son:

    - Las señales de selección de sensor, una señal emitida a través de cuatros pines (D4 = S0, D5 = S1, D6 = S2, D7 = S3) de forma simultánea. Esto se realiza para evitar seleccionar otros sensores indeseados ya que, si se mandan las señales individualmente, existe una transición de tiempo entre los instantes en que se envía una y la siguiente.
      La selección del sensor viene definida por el numero binario formado por dichos pines, siendo: los numero pares y cero el Trigger de los sensores, y los impares el Echo. El sensor seleccionado viene dado por uno más la mitad del número de puerta asignada. Por ejemplo, si la entrada del multiplexor es 8, se habrá seleccionado el Trigger del sensor 5 = 1+ 8/2. Si en lugar de 8, se asignara un 9, la conexión seria con el Echo del sensor 5.

- La señal que se envía o recibe del propio sensor mediante el pin D2 (=SIG). En función de la salida asignada, la señal a través de este pin será de entrada (Echo) o de salida (Trigger).

o Bluetooth

Las señales existentes entre Arduino y el dispositivo Bluetooth son las correspondientes a una comunicación serie a través de los pines A3 y A12.

o PWM (motores)

Para regular de forma controlada la potencia transferida a los motores y, por tanto, la velocidad, es necesario enviar señales tipo PWM a la entrada correspondiente del puente del motor requerido.

Estas señales las forma el propio dispositivo variando el duty de la señal, de amplitud 5 Voltios, en función del valor introducido (de 0 a 255).

- ***Variables Electrónicas***

El correcto funcionamiento del vehículo depende de que las señales enviadas y recibidas por el Arduino sean adecuadas, tanto en intensidad como duración. Por tanto, a continuación, se mostrarán las variables que influyen en la electrónica del conjunto:

- Señales PWM: son necesarias para controlar las diferentes acciones del vehículo (giro izquierdo, giro derecha, marcha hacia delante y marcha hacia atrás). No se programa la señal directamente, sino que señala el duty deseado, entre 0 y 255, y el programa de Arduino se encarga de la gestión de los tiempos en alto y bajo de la señal emitida.
- Entradas digitales: se corresponden con las señales de Echo de cada uno de los sensores de ultrasonidos.
- Salidas digitales: además de las correspondientes a las señales PWM, solamente existen la de los Trigger de los sensores e ultrasonidos, y las utilizadas para la selección de un sensor en concreto a través de un multiplexor.
- Éstas se realizarán mediante un dispositivo Bluetooth que transmite señales desde una aplicación móvil hasta el Arduino.
- Brújula electrónica: La variable que impera en la realización de las maniobras es la posición angular del vehículo, por lo que el análisis de las variables y el correcto funcionamiento del dispositivo es fundamental.

**Las tablas de definición de variables están contenidas en el Anexo 2.*

- Resumen

Teniendo en cuenta todo lo anterior, el funcionamiento del vehículo desde el punto de vista electrónico se puede resumir en los siguientes puntos:

- La velocidad, variable en función de la maniobra en la que se encuentre el prototipo, se regula mediante una señal PWM procedente de Arduino. También cambia si se detectan obstáculos en la fase de búsqueda de sitio.

- El dispositivo Arduino se conectará a los sensores de ultrasonidos mediante un multiplexor, de 16 canales (se utilizan 2 por sensor [Trigger, Echo]), reduciendo el número de pines necesarios en el microcontrolador. En el funcionamiento se van activando y desactivando los distintos canales en función de las necesidades de acceder a uno u otro sensor de forma que con 4 pines (entrada/salida y activación/bloqueo) se pueden manejar los 6 sensores.

- La brújula se conecta directamente a dos pines asociados a la comunicación I2C de Arduino.

- El sentido de avance y la dirección dependerá de cual sea la señal enviada a los puentes en H.

A continuación, se muestra el esquema de la protoboard creado con Fritzing:

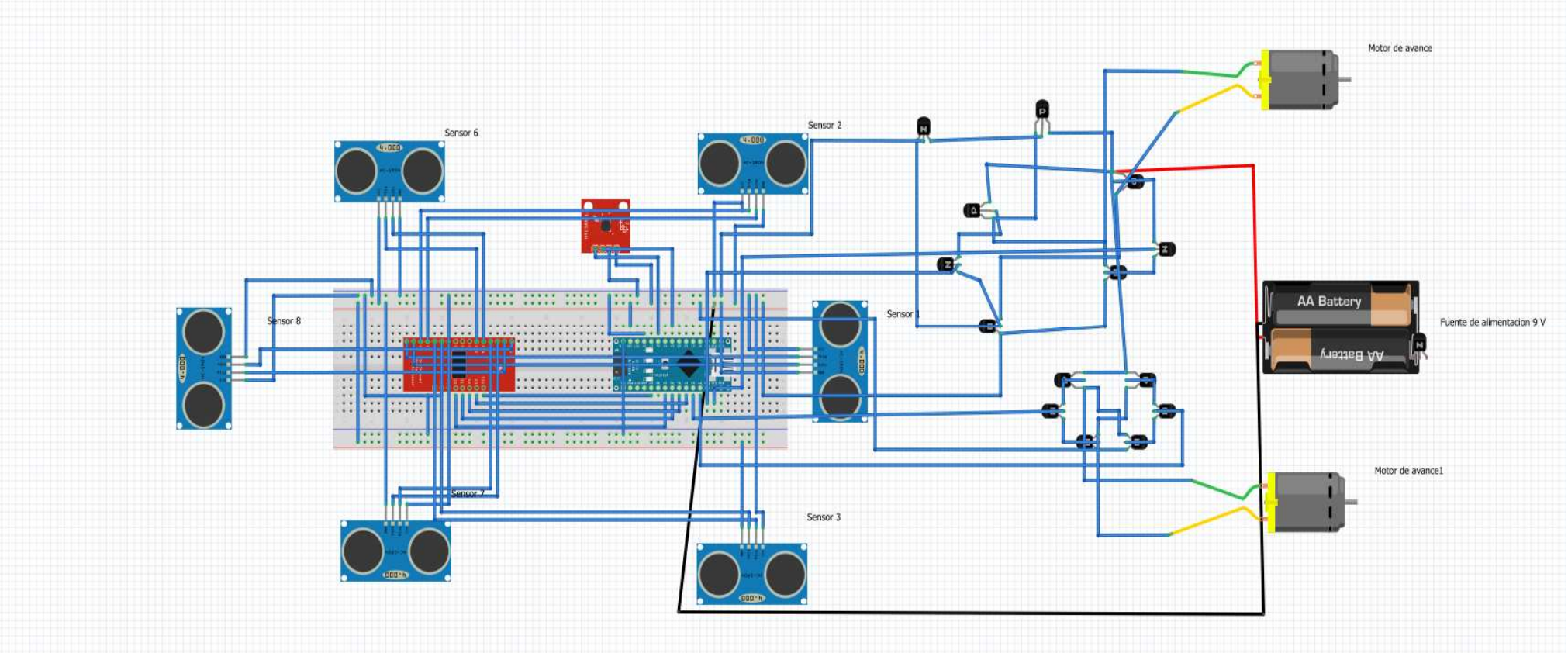


*Fig. 8: *Vista de protoboard creada con Fritzing*

5. ESTRUCTURACION DEL PROCESO DE ESTACIONAMIENTO

Como se ha mencionado anteriormente no existe una única manera de realizar el estacionamiento para todo vehículo, aunque siempre se siguen las mismas pautas o fases generales.

Las fases del proceso de estacionamiento se resumen en:

I. Posicionamiento inicial.
II. Búsqueda de sitio.
III. Comprobación de tamaño de hueco.
IV. Posicionamiento para maniobra.
V. Ejecución de maniobra.

4.1. Posicionamiento inicial

Inicialmente, el vehículo se encuentra en un estado aleatorio en el carril correspondiente, por lo que lo primero que se hace es posicionarlo respecto a un objeto de referencia (una pared, otro vehículo…). Se supone que existe una referencia cercana plana sobre la cual poder realizar dicho proceso. En la realidad, este apartado se traduce en mantener el vehículo paralelo a la vía al entrar en una calle o parking. Para ambos tipos de estacionamiento se realiza de la misma manera.

Desarrollo del proceso:

1.1. En primer lugar, los sensores obtienen las distancias del lado en el que se desea estacionar, de tal manera que se pueda hallar la guiñada deseada del vehículo.

1.2. Posteriormente, el vehículo posiciona las ruedas hacia el lado correspondiente y se desplaza hasta que ambas distancias sean iguales.

La importancia del uso de la brújula reside en la continua verificación de que el vehículo se encuentra en la posición correcta, ya que almacena el ángulo final del posicionamiento. Además, se utilizará para el resto de procesos cuando se trabaje con el movimiento angular del vehículo.

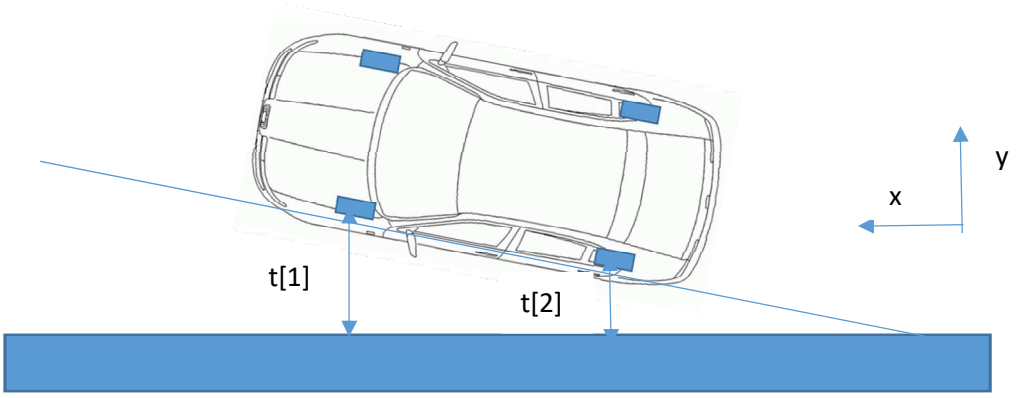


Fig. 9

1.3. Finalmente, el vehículo se desplaza en dirección Y hasta la posición definida para cordón/batería. Para ello se compara el valor de la constante del tipo de estacionamiento correspondiente (d_despl_f, d_despl_b) con las distancias obtenidas de los sensores (t[i]). De la anterior comparación se obtiene el ángulo de giro con el que realizara la maniobra de desplazamiento que se resume en un doble giro, el primero con las ruedas giradas hacia lado al que se desea desplazar, y el segundo hacia el lado contrario.

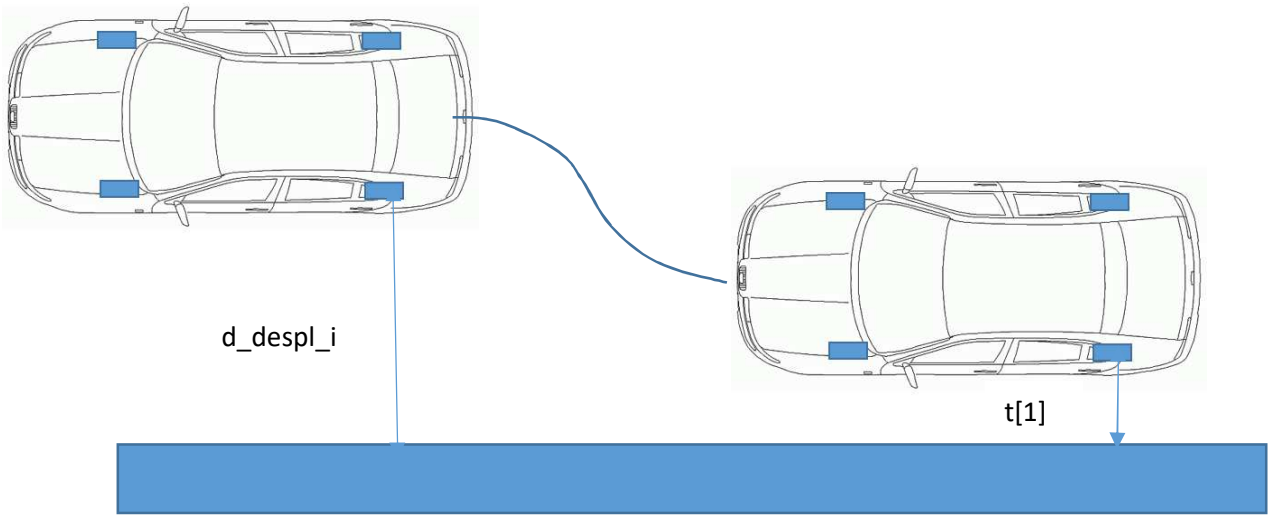
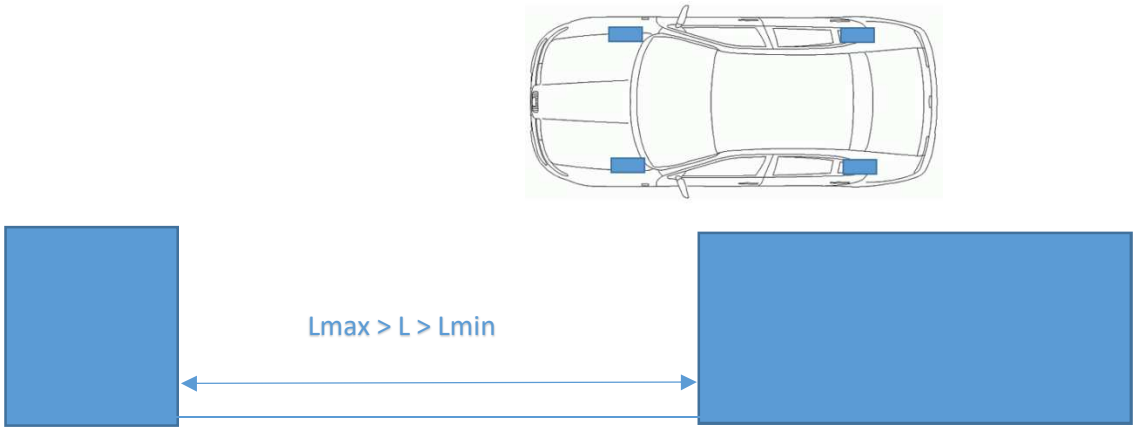


Fig. 10

4.2. Búsqueda de sitio.

Básicamente, ningún vehículo puede estacionar si el hueco disponible no es lo suficientemente grande para realizar la maniobra sin golpear a los vehículos contiguos. Por ello, el espacio necesario es la incógnita principal en todos los cálculos de esta sección. Cada tipo de estacionamiento tendrá una serie de variables y constantes asignadas de manera que se pueda optimizar el código en cuanto a espacio y comprensión. El proceso que se lleva a cabo es el siguiente:

- Una vez posicionado el vehículo, pasa a la fase de búsqueda, en la que fija una velocidad adecuada para la correcta detección de los sensores. Se utilizan los 2 sensores laterales del lado correspondiente al estacionamiento para verificar que existan las distancias necesarias a lo largo del proceso. A su vez, mediante el uso de la brújula, se mantendrá la dirección del vehículo constante corrigiendo cualquier posible desviación.
- El primer sensor en detectar hueco debe ser el delantero cuando detecte un hueco en su zona. Una vez se activa, la velocidad se reduce considerablemente para mejorar las medidas de los sensores y evitar errores.



* *Fig. 11*: *Buscando sitio en cordón*

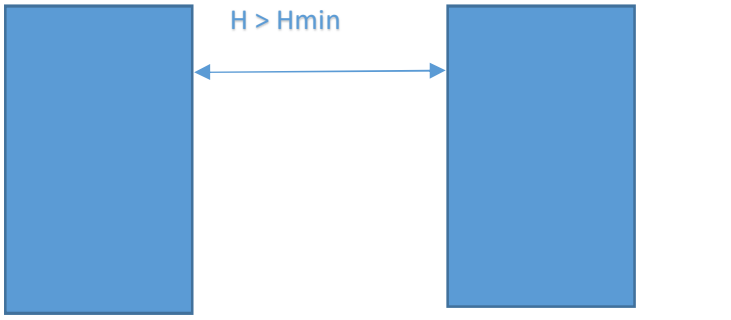
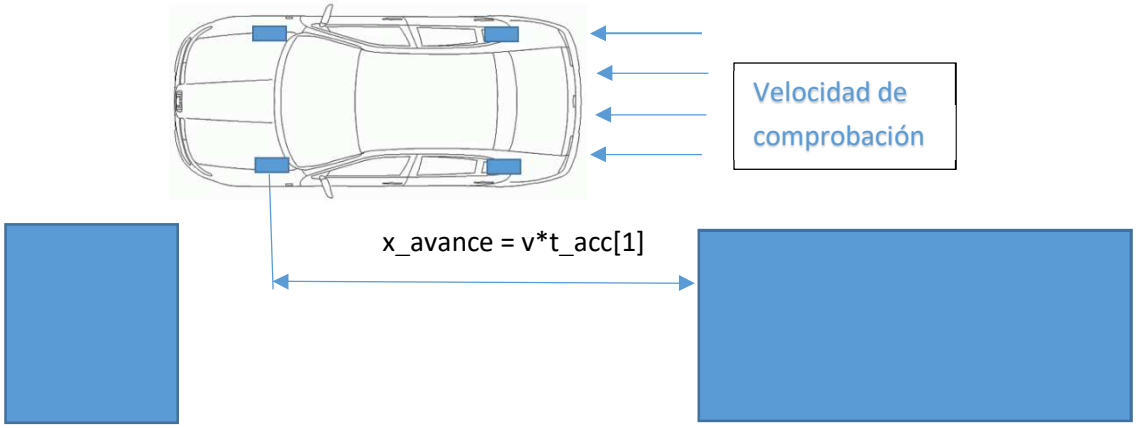


*Fig. 12: Buscando sitio en batería

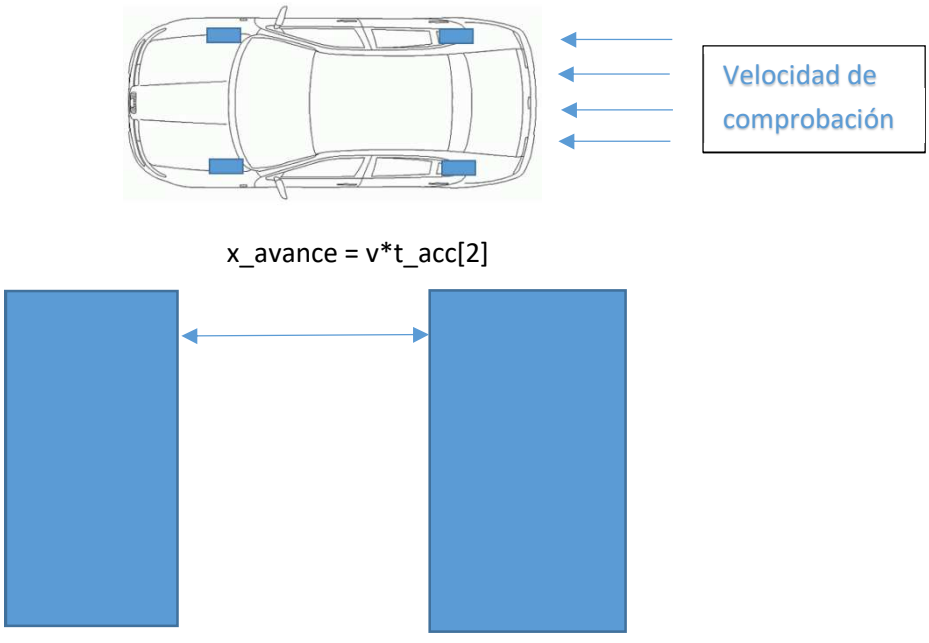
4.3. Comprobación de tamaño de hueco.

Esta fase se basa en la verificación de que el sitio que se ha encontrado es lo suficientemente grande para aparcar. Para ello, se compara el valor mínimo del hueco necesario para estacionar en cordón con el obtenido por los sensores.

- El vehículo avanza hasta haber alcanzado una distancia determinada o haber detectado un vehículo con el sensor delantero. Dependiendo de la situación que aparezca pueden existir 3 posibilidades:
    - En el caso de que detecte un vehículo antes de avanzar la mínima distancia de estacionamiento, el proceso vuelve al estado de búsqueda inicial.
    - En el caso de que detecte un vehículo tras haber avanzado la distancia, pero sin llegar a la máxima, se pasa a la fase de posicionamiento para estacionar en las maniobras necesarias.
    - En el caso de que alcance la distancia máxima, el vehículo comienza la fase de posicionamiento para estacionar en una sola maniobra.



* *Fig. 13:* Comprobación hueco cordón



- *Fig. 14:* Comprobación hueco batería

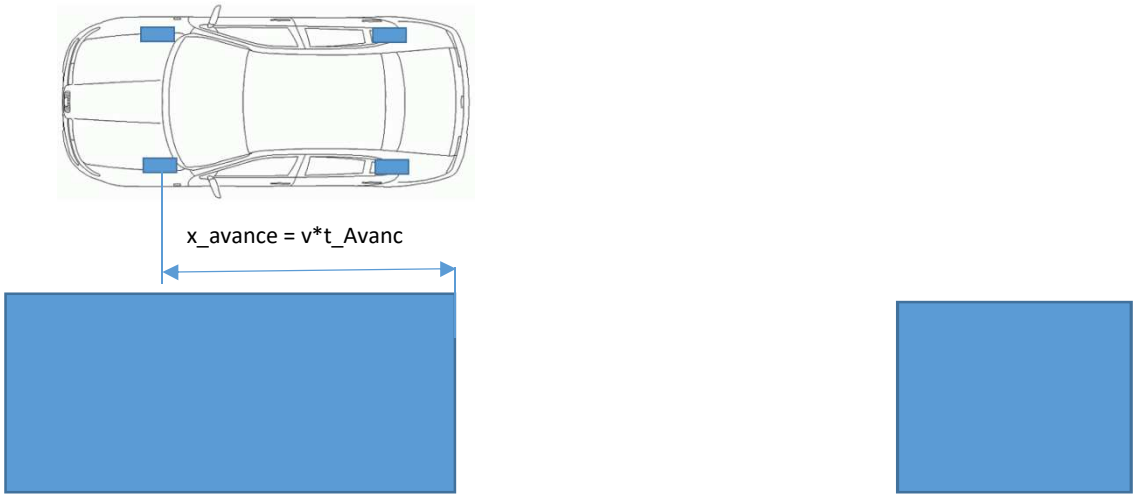
4.4. Posicionamiento para maniobra.

El posicionamiento para la maniobra solo se realiza en el eje X (longitudinal a la vía), ya que, en el posicionamiento inicial, se tuvo en cuenta la posición en el eje Y al realizar el desplazamiento lateral. El error que se comete es menor de esta manera, a la vez que se reduce el número de maniobras que ejecuta para situarse.

Al tratarse de la posición pre maniobra se debe tener en cuenta cualquier posible desviación, ya que, si hay algún error de posición, los cálculos previos realizados para evitar la colisión con el resto de elementos se verían afectados negativamente.

Para realizar dicho posicionamiento, el vehículo, una vez ha detectado que el tamaño del hueco es mayor que el valor mínimo impuesto, avanza una distancia determinada de tal manera que:

- El vehículo debe finalizar paralelo a la vía.
- La distancia en el eje Y debe mantenerse igual a la inicial.
- La distancia en el eje X debe permitir al vehículo realizar la maniobra e impedir la colisión con los elementos tomados como referencia de maniobra.



***** *Fig. 15*: *Posicionamiento maniobra*

4.5. Ejecución de la maniobra.

Una vez llegado a este punto, se ha de tener en cuenta la acumulación de errores que puedan existir en los pasos previos, por tanto, las maniobras deben realizarse con un margen de seguridad, lo que se traduce en la introducción de distancias de seguridad para cada maniobra. Estas distancias son proporcionales al tamaño del vehículo por lo que se calculan en función a parámetros o constantes del propio vehículo.

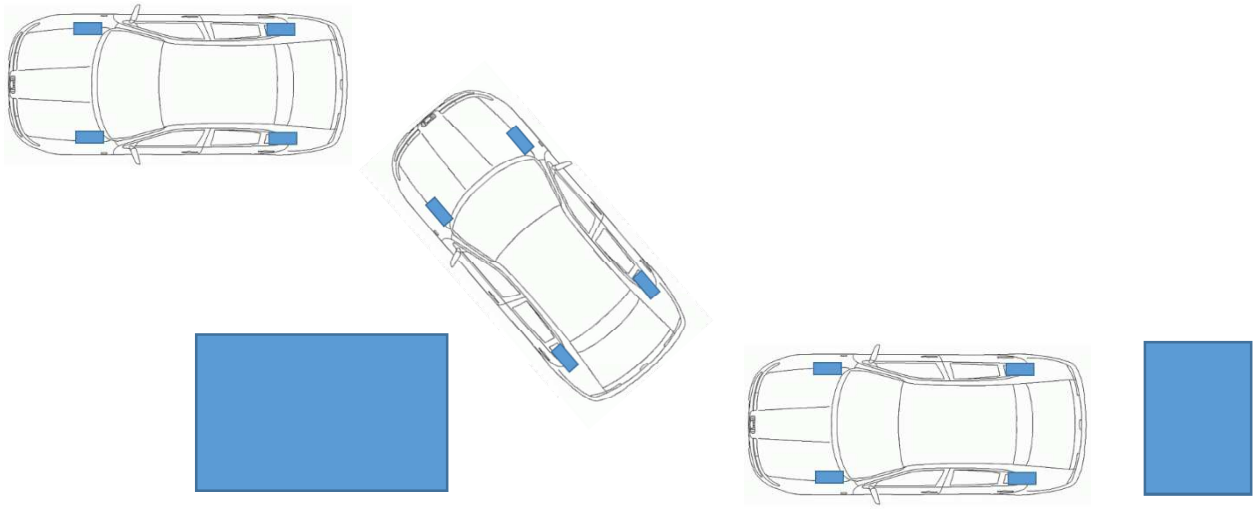
La ejecución de la maniobra se realiza, tanto para cordón como para batería, a la velocidad mínima, ya que la exactitud de los ultrasonidos juega un papel fundamental. El proceso para ambos se estructura como se explica a continuación:

➔ Proceso de maniobra en cordón:

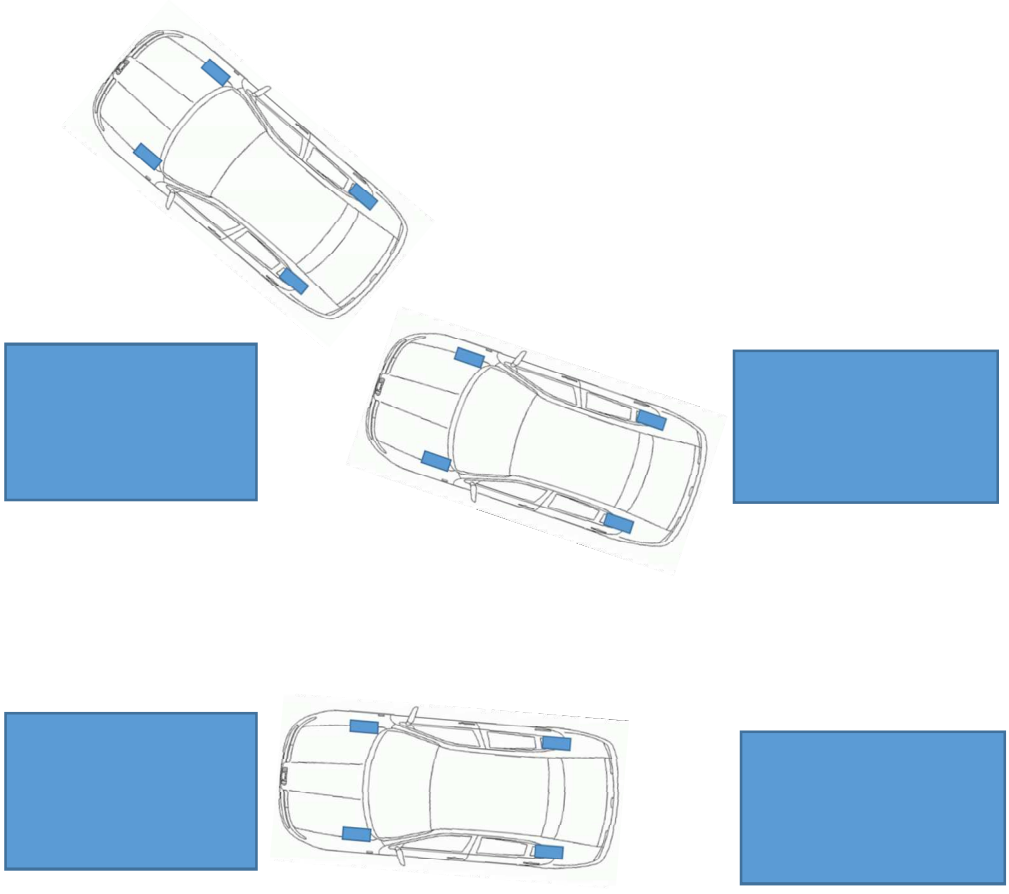
(1) Primer paso: Tras la previa colocación del vehículo, se giran las ruedas hacia el lado en el que se va a aparcar. Después, se acciona la marcha atrás hasta que el vehículo ha girado 45 grados. Al finalizar la acción se detiene
(2) Segundo paso: Una vez la brújula ha detectado que el vehículo se encuentra a 45º respecto a la referencia inicial, se giran las ruedas en sentido contrario y se vuelve a accionar la marcha atrás. En esta fase el sensor trasero del vehículo informara si hay riesgo de colisión con el objeto trasero.
(3) Tras la primera maniobra de estacionamiento, si el vehículo no ha conseguido completarlo, se iniciará una segunda que consiste en el giro de las ruedas y avance en sentido contrario. Este proceso se realizará hasta que la brújula indique que la posición angular del vehículo es igual a la referencia.

➔ Maniobra de estacionamiento en batería:

(1) Primer paso: Tras la previa colocación del vehículo, se giran las ruedas hacia el lado en el que se va a aparcar. Después, se acciona la marcha atrás hasta que el vehículo ha girado 90 grados y se detiene.
(2) Segundo paso: se deshace el giro de las ruedas y se inicia la marcha atrás. A la vez que se va introduciendo el vehículo en el hueco los sensores delanteros detectaran si existe algún obstáculo cercano. En el momento en el que se active uno significara que ya se ha estacionado correctamente sin llegar a colisionar con la pared (si la hubiera) y se detiene.



** Fig. 16*: *1 Maniobra en cordón*



** Fig. 17*: *Más de 1 Maniobra en cordón*

6. CALCULO DE LA GEOMETRIA DEL ESTACIONAMIENTO

La complejidad y longitud del código hace necesaria la definición de variables para una mejor comprensión de éste. Antes de comenzar con el análisis referente a la geometría, se van a definir todas las variables que se emplearan a lo largo del apartado.

En primer lugar, se analizarán las variables correspondientes a la geometría del propio vehículo. El resto de variables dependerán directamente de estas de tal forma que, aunque varíe el tamaño del vehículo, seguirá realizando las maniobras correctamente.

Una vez se han obtenido los datos referentes a la geometría, se analizan las variables relacionadas con: la posición relativa entre el vehículo y el entorno, las maniobras que realiza el vehículo a lo largo del proceso y los valores calculados con los que se compararán los obtenidos de los sensores. Puesto que en la programación se utilizarán valores de tiempos, se deberán dividir todas las distancias por la velocidad del sonido (340 m/s) o la velocidad del vehículo, dependiendo del tipo de variable. Para ambas maniobras se habrá de tener en cuenta el ángulo de giro en el que se encuentra el vehículo, además de los radios de giro de las dos esquinas del lado exterior (R' y R''). Dependiendo del tipo de estacionamiento habrá que definir variables adicionales y constantes.

Para el estudio del estacionamiento en cordón es necesario definir dos constantes XCI y YCI que harán referencia a las distancias iniciales del vehículo, respecto una referencia, y una tercera que sirva de variable para medir el avance del vehículo en el eje X (x'). A su vez, se tomará las distancias máximas de avance en X e Y para realizar futuras comprobaciones. Las variables correspondientes a la geometria del vehículo comprenden todas aquellas que se utilizaran como referencia para el resto de cálculos (base de cálculos). A continuación, se muestra más detalladamente a que esta referenciada cada una de ellas:

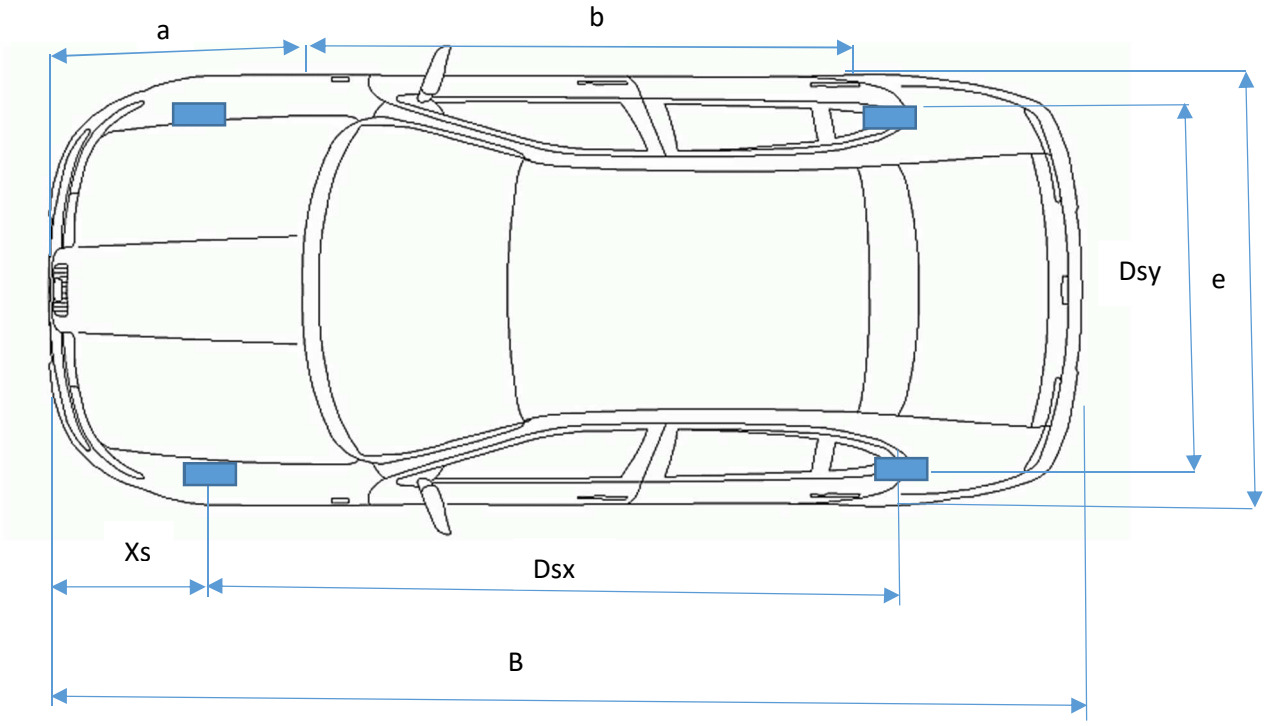


Fig. 18

**Las tablas de definición de variables están contenidas en el Anexo 2.*

Posicionamiento inicial.

Para el cálculo del ángulo a corregir en el posicionamiento inicial se tendrán en cuenta las medidas de los dos sensores del lado correspondiente (x1, x2) y la distancia entre los sensores en el eje X, de tal manera que la ecuación resultante sería la siguiente:

$$\beta[\text{rad}] = \text{atan}(\frac{|x2 - x1|}{\text{Dsx}})$$

Tras la corrección del ángulo anterior, el vehículo ha de desplazarse lateralmente hasta la distancia del estacionamiento correspondiente (cordón, batería). Para realizar la maniobra, el vehículo se desplazara girando hacia el frente hasta alcanzar un ángulo β y realizara la misma maniobra en la dirección opuesta. De esta manera la distancia desplazada puede calcularse como:

$$d[cm] = 2 * (R[cm] + \frac{e}{2}) * (1 - \cos(\beta[\text{rad}])$$

Búsqueda.

En el proceso de búsqueda lo primordial es el control de la velocidad del vehículo ya que el cálculo del hueco existente se realizará mediante tiempos de avance del vehículo.

Estacionamiento.

Puesto que la posición Y ya se ha fijado en el paso previo, solo resta fijar la posición en el eje X a la hora de posicionar el vehículo para estacionar. Los cálculos varían en función del tipo de estacionamiento como se muestra a continuación:

- *Cordón*
  - El posicionamiento se realizará respecto al vehículo situado en frente, por lo que el primer paso es hallar los centros de giro sobre los cuales se trabajará (C1 y C2). Como referencia se utiliza la esquina del parachoques trasero del vehículo delantero.

    C1x = X' (se calculará posteriormente y será la distancia que recorrerá el vehículo tras detectar que el espacio es lo suficientemente grande).

    C1y = Yci - R. (Yci→ Se calcula a partir de las limitaciones para realizar la maniobra como se explica a continuación)

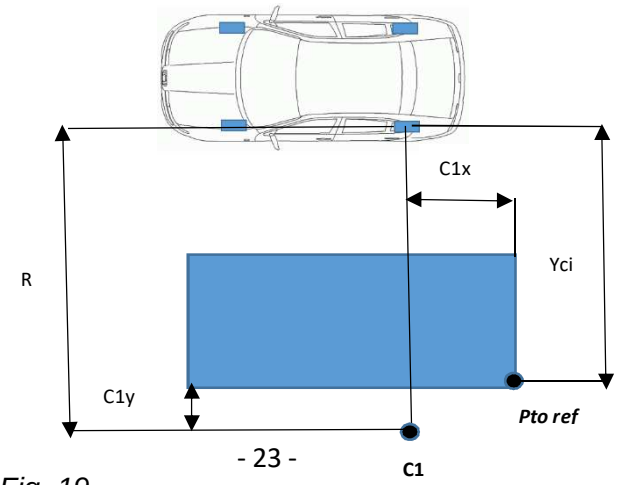


Fig. 19

- La maniobra que se realiza es de un doble giro de 45º en cada dirección, de forma que la distancia que se desplazará cualquier punto del vehículo en el eje Y será:

d =2 $*$ $(R + \frac{e}{2}) * (1 - \cos\beta)$ [para β = 45 y R cte] → d = 36,14 cm

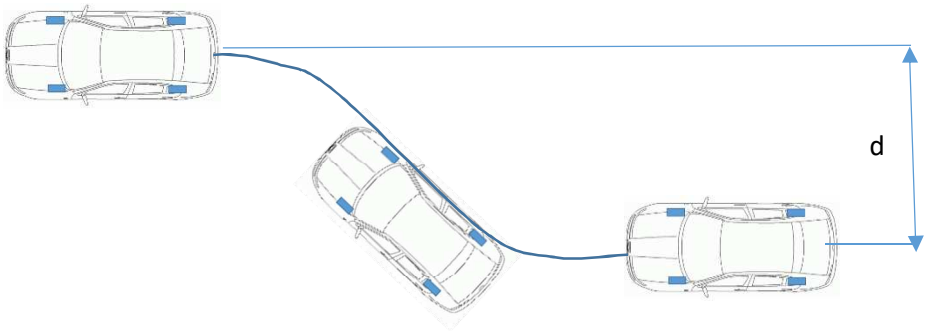


Fig. 20

Si luego le añadimos la parte trasera del vehículo, ya que el punto de análisis es la rueda trasera izquierda, que impactaría en la pared al realizar el segundo giro.

$$d' = d + (R' - e - R) = 37{,}44 \text{ cm}$$

siendo R' = $\sqrt{(R+e)^2 + a^2}$

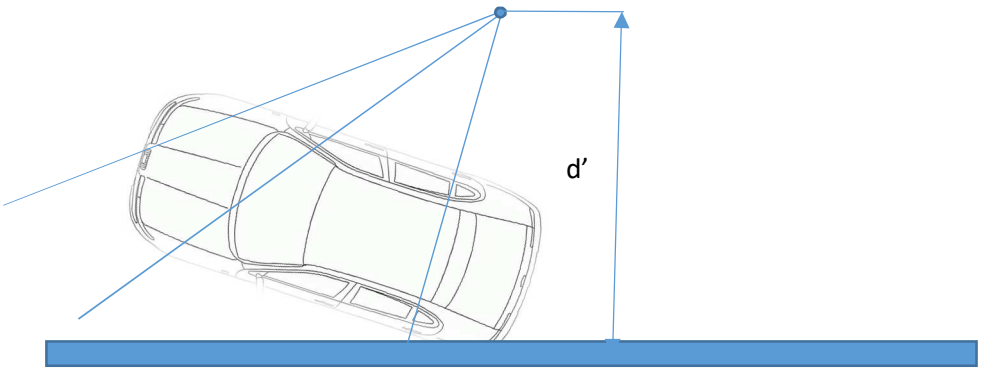


Fig. 21

De esta forma se asegura que no golpeara la pared (si la hay) al realizar la maniobra.

$$Xci = d'$$

- La segunda comprobación verifica que, al realizar el primer giro, el lateral del vehículo no roce la esquina del delantero.

  X' = $\sqrt{(R+MS)^2-(R+e-d'+MS)^2}$ por pitagoras → X' = -33,137 cm tomando como margen de seguridad 5 cm.

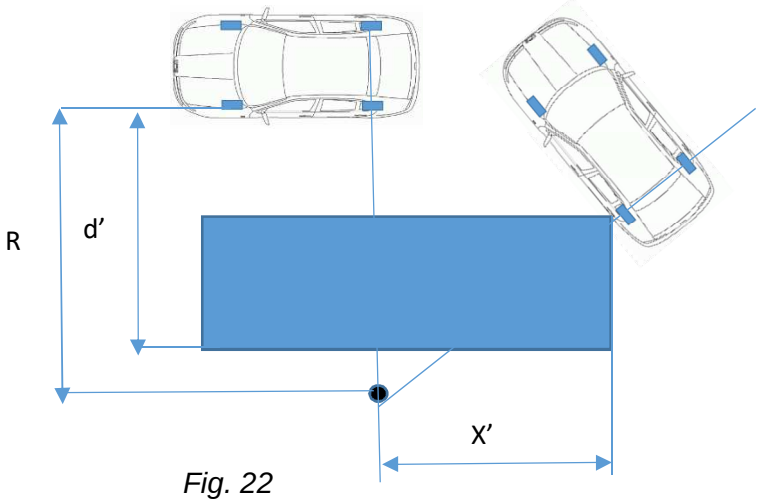


Fig. 22

- En tercer lugar, la esquina exterior del parachoques delantero no debe impactar con el vehículo delantero, por lo que se calcula la trayectoria r'' de dicho punto y se recalcula la distancia x'. De este apartado se obtendrá el hueco mínimo para realiza el estacionamiento mediante una sola maniobra.

  X' = $Avance\max-\sqrt{(R''+MS)^2-(r'-e+MS)^2}$ = 15,97 cm
  Siendo: R'' $=\sqrt{(R+e)^2+(a+b)^2}$ = 82,07 cm
  $Avance\max=2*(R+\frac{e}{2})*SIN(\beta)$ = 87,257 cm

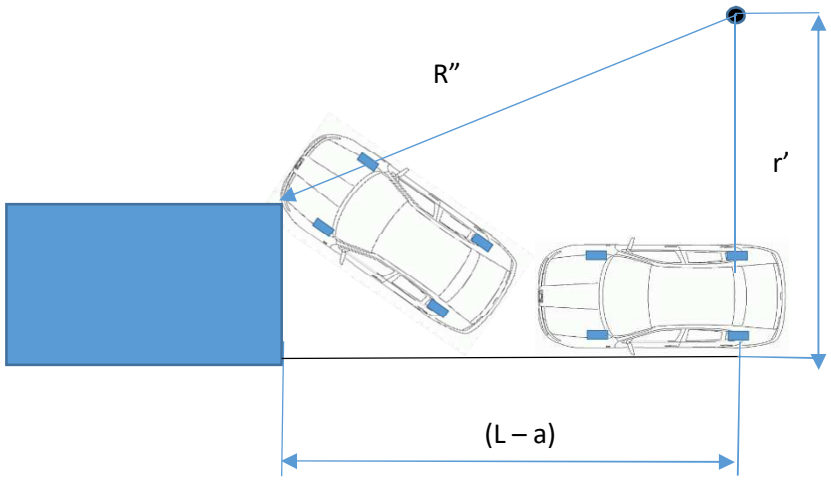


Fig. 23

- Al obtener dos valores distintos para una variable se optará por la más restrictiva, la menor de las dos, lo que se traduce en la búsqueda de un hueco mayor. De esta manera, ya se ha obtenido la posición exacta del vehículo con una referencia fija [X(parachoques coche delantero), Y(pared)].
  Xci = -15,97 cm → posicionamiento tras detectar hueco.
  Yci = 37,44 cm → posicionamiento inicial.

- Una vez se ha situado el vehículo se procede a realizar la simulación del estacionamiento.
  - En primer lugar, se tiene que fijar un punto del vehículo que se utilizará como referencia para los cálculos posteriores. En este caso, se utilizará la rueda del lado correspondiente al tipo de estacionamiento (derecha, izquierda) del eje trasero.
  - Tras fijar la referencia se realizan los cálculos de la posición X (Avance) del vehículo en cada fase de la maniobra, ya que se utilizarán los valores obtenidos para los siguientes cálculos.
  - La maniobra finaliza cuando el ángulo de inclinación del vehículo es igual al de referencia, por lo que también habrá que calcular los valores del ángulo en cada fase de la maniobra.

- *Batería*

- El posicionamiento para la maniobra, en este caso, se realiza tomando como referencia la esquina del lateral del vehículo consiguiente y de la misma forma que en el estacionamiento en cordón. La única diferencia es la distancia que se desplaza el vehículo (X') tras detectar el suficiente hueco. La distancia anterior se calcula a partir del radio de giro de dos puntos del vehículo:
  - Rueda interior trasera, debido a que es la parte con menor radio de giro, se ha de tener en cuenta en los cálculos.
  - Esquina exterior trasera. Dado que la maniobra se realiza marcha atrás, el elemento que corre riesgo de colisión es dicho punto.

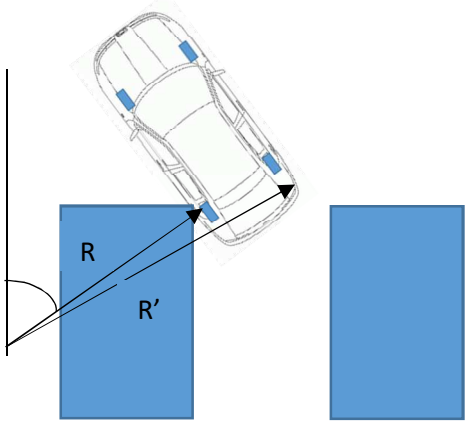


Fig. 24

- Por último, se acciona la marcha atrás durante el tiempo suficiente para que el vehículo recorra una distancia determinada. Puesto que si se utilizaran únicamente los sensores para asegurar en que momento llega al final, podrían existir escenarios de incertidumbre (que no haya vehículos alrededor). En este caso, dado que la maniobra es arbitraria, no se calculan ni el avance ni el ángulo del vehículo en cada instante. El punto de estudio será el mismo que en el análisis previo, la rueda del lado correspondiente al tipo de estacionamiento (derecha, izquierda) del eje trasero.
  La condición de fin de estacionamiento viene dada por la primera de dos situaciones: detección de un obstáculo por el sensor trasero o que la distancia definida haya sido alcanzada.

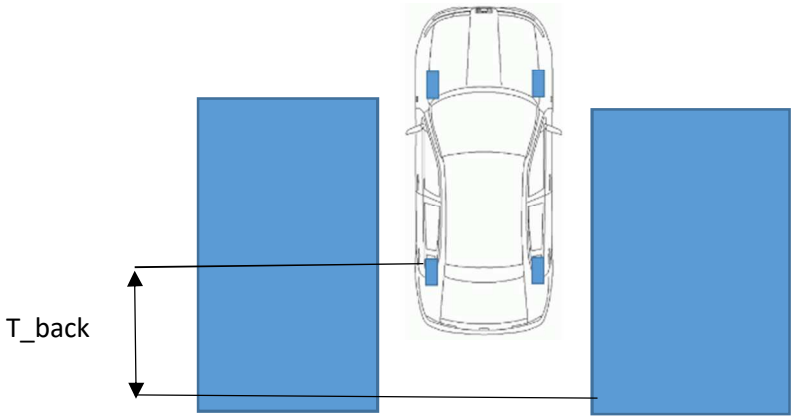


Fig. 25

- La simulación para este tipo de estacionamiento, de la misma forma que la de cordón, se basa en la variación de la amplitud del hueco y obtención de la posición inicial en la que debe situarse el vehículo para realizar la maniobra correctamente con las distancias de seguridad impuestas.

**Los cálculos para la realización del estacionamiento se presentan en el ANEXO 3*

La primera acción que se debe llevar a cabo es el posicionamiento inicial, tanto por la desviación angular como por la distancia a la que debe colocarse respecto a un objeto contiguo. Para ello se utilizará la ecuación (2) de la cual se obtendrá el ángulo de maniobra con el que, sumándole el aporte de la brújula, se realizará la maniobra.

Para ejecutar el estacionamiento se ha de hacer un estudio de los valores de las distancias que habrá que introducir:

- Estacionamiento en fila/cordón. Los datos que se introducirá serán, dependiendo del tipo de análisis que se elija; el hueco existente(L) o la distancia con el vehículo contiguo(Y). La distancia de posicionamiento previo al estacionamiento(X) y los parámetros de maniobra se calcularán a partir de los datos anteriores.
- Estacionamiento en batería. Los datos introducidos para este caso será únicamente el hueco existente(H), ya que el coche hará los cálculos de posición posteriormente (X, Y).

7. APLICACIÓN MOVIL ESTACIONAMIENTO AUTOMATICO

Todo programa que tenga más de una funcionalidad necesita de unos parámetros de entrada y una señal u orden de activación. En este caso, todas ellas se realizan a través de una aplicación móvil que envía los datos mediante Bluetooth al Arduino. Éste recibe los datos en forma de byte, lo analiza y les asigna los valores adecuados a las variables correspondientes a dichos parámetros.

En primer lugar, se han de fijar las partes del byte que se utilizan para transmitir la información. Puesto que únicamente se manda la información de tres variables, se seleccionan los tres bits menos significativos de tal manera que el byte enviado tiene la siguiente estructura:

0	0	0	0	0	1/0	1/0	1/0

- Bit 0: indica el momento en el que se manda la señal de activación del programa:
    - ON → bit0 = 1.
    - OFF → bit0 = 0.
- Bit 1: selecciona el tipo de estacionamiento:
    - Cordón →bit1 = 1.
    - Batería → bit1 = 0.
- Bit2: selecciona el lado en el cual se estaciona:
    - Izquierda → bit2 = 1.
    - Derecha → bit2 = 0.

Para la asignación de dichos valores en la lógica, se utilizan tres variables booleanas: *v_acionamiento, IZDA y FILA.*

- **V_accionamiento** toma valor de TRUE, cuando se manda la señal de activación del programa, y FALSE, cuando no se envía información o el programa acaba de finalizar.
- **IZDA** toma valor de TRUE, cuando a través del bit2 se interpreta la selección del lado *Izquierda*, y FALSE, si se interpreta la selección del lado *Derecha*.
- **FILA** toma valor de TRUE, cuando a través del bit1 se interpreta la selección del estacionamiento en *Cordón*, y FALSE, si se interpreta la selección del estacionamiento en *Batería*.

8. PROGRAMACIÓN FINAL

9.1. DESARROLLO DE FUNCIONES

En primera instancia, se presenta el desarrollo de las funciones utilizadas para la creación de la lógica con una breve descripción de cada una de ellas.

Función	Breve descripción
Aparca	Función principal de estacionamiento
Posiciona	Posicionamiento inicial del vehículo
Busca_sitio	Estado de búsqueda de hueco para estacionar
Selección maniobra	Selección de la maniobra necesaria para estacionar en función de los parámetros iniciales y datos obtenidos en *busca_sitio*
Maniobra_fila_1	Maniobra con fases arbitrarias de estacionamiento en cordón
Maniobra_fila_2	Maniobra con fases dinámicas de estacionamiento en cordón
Estado_mani_1	Fase 1 de *Maniobra_fila_2* : retrocede/derecha
Estado_mani_2	Fase 2 de *Maniobra_fila_2* : avanza/izquierda
Maniobra_bateria	Maniobra con fases arbitrarias de estacionamiento en bateria
Inicializar_var	Inicialización de variables globales
PWM	Envío de señal analógica al pin seleccionado
Lee_ultrasonidos	Lectura del valor obtenido del sensor tras ejecutar *Lanza_pulso*
Lanza_pulso	Envío de señal al sensor para que se lance un tren de pulsos
Calcular_angulo_ref	Lectura del valor de posición angular del vehículo tras posicionamiento inicial y asignación de dicho valor a *angulo_ref*
Calcular_angulo	Lectura del valor de posición angular del vehículo respecto a valor de referencia de *angulo_ref*

Fig. 26

9.2. ESTRUCTURACION DE LA LOGICA

Puesto que el programa representa una cadena de sucesos, la forma de representación será la de un árbol de sucesos que contendrá el programa principal en el *TOP* y las sucesivas funciones ramificadas partiendo de este. A continuación, se explica estructuradamente el desarrollo del programa incluyendo la declaración de variables, el programa principal y todas las funciones que este contiene:

- Declaración de variables, tablas, estructuras, constantes, bytes y otras definiciones.

- SETUP, donde se inicializan las variables y se asignan las entradas y salidas del Arduino.

- MAIN o programa principal, esencialmente definido por un LOOP en el cual se introducirán los comandos que se desean ejecutar.

- Funciones auxiliares, estas son las correspondientes a los elementos externos al Arduino y que se llamaran a lo largo del programa. Las funciones auxiliares de este programa son:

  - ***PWM (X, Y):*** función cuyo objetivo es mandar una señal PWM de valor 'X' al pin 'Y' correspondiente. La principal utilidad de esta función es variar la velocidad del vehículo en función del estado de estacionamiento en el que se encuentre. Para ello se han definido variables referentes a la velocidad para posibles modificaciones en la fase de pruebas del prototipo.

    Al llamar a *PWM (X, Y),* se realiza una conversión del valor de 'X', en función de la velocidad máxima del vehículo, a un valor de entre 0 y 255. De esta manera, el valor de la señal de salida del pin 'Y' será adecuado para su funcionamiento.

  - ***Inicializar_var*** *():* función creada con el fin de juntar todas las variables, registros de tablas, estructuras, bytes que posteriormente se utilizaran como comparación, en una sola localización.

    Se llama una sola vez al iniciar el Arduino y contiene los valores de todos los tiempos de ejecución (T_acc[i]), 'distancias' de seguridad (T_seg[i]), velocidad del vehículo en cada una de las fases (v_arranque, v_pos_ini, v_desplaz, v_b_sitio, v_check_sitio, v_pos_man, v_maniobra, v_maniobra_bat), el número de los sensores correspondientes al lado de estacionamiento (s_a, s_b) y el estado de lectura de la señal bluetooth (estado).

  - ***Calcular_angulo_ref ():*** función ligada a *'Calcular_angulo'* en la cual se obtiene el valor del ángulo de referencia, una vez el vehículo se ha situado paralelo al objeto de referencia.

    Se llama cada vez que se ejecuta el programa, tras el posicionamiento del vehículo con el fin de fijar el valor de la variable '*angulo_ref*', ya que a partir de esta se desarrollaran los cálculos posteriores en la función '*Calcular_angulo ()*'.

  - ***Calcular_angulo ():*** función necesaria para la obtención del ángulo del vehículo en la que se compara los valores obtenidos de la función *'Calcular_angulo_ref*' con los obtenidos de la brújula, de tal manera que se pueda calcular la diferencia de ángulo entre estos.

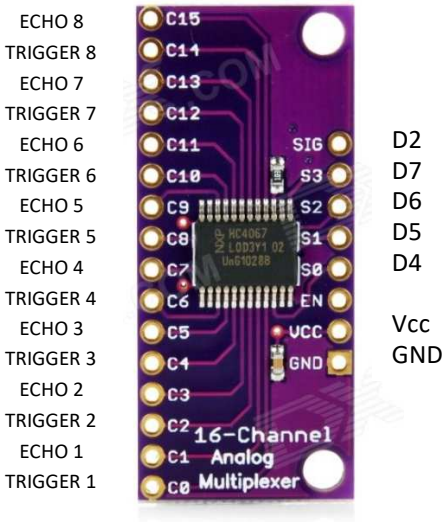
  - ***Lee_ultrasonidos(x,y):*** función encargada de obtener las distancias medidas por los sensores en el instante de analizar el entorno en busca de obstáculos. El funcionamiento de esta función se basa en el envío de un tren de pulsos, que realiza el propio sensor al recibir una señal del Arduino (esta señal se crea a través de la función '*Lanza_pulso ()'*), y la medición del tiempo que transcurre entre el lanzamiento y la recepción de dicho tren de pulsos. De esta manera, la distancia que han recorrido estas ondas puede calcularse como:

$$Distancia = \frac{Tiempo_{sensor}}{1000}[ms] * \frac{340\left[\frac{mm}{ms}\right]}{2}$$

El tiempo medido por el sensor será dos veces la distancia entre este y el objeto en el cual se han reflejado las ondas enviadas previamente. Puesto que el medio por el que circulan será el aire se puede asumir la velocidad del sonido como 340 metros por segundo.

Este sería el funcionamiento en el caso de tener dos pines asignados a cada sensor, ya que únicamente habría que enviar la señal por el pin conectado al TRIGGER y habría que medir dicho tiempo en el pin asignado al ECHO. En este caso, se ha escogido la opción de utilizar un multiplexor para realizar dicha tarea utilizando una menor cantidad de pines y facilitando el desarrollo de la función de lectura.

En primer lugar, se inicializa el puerto ECHO en modo salida OUTPUT ya que la primera acción es emitir el tren de pulsos. Para seleccionar el pin correspondiente de cada sensor y para que la programación sea lo más sencilla posible, la disposición de los pines correspondientes a cada sensor es la siguiente:



La posición del TRIGGER de cada sensor viene determinada por la fórmula: $sonar = 2 * (sensor - 1)$. Para que el multiplexor pueda interpretar este valor se crea un byte con el número del sensor en la parte alta y 0 en los cuatro bites bajos, se saca la dirección del multiplexor para el TRIGGER sin cambiar la parte baja del puerto y se ejecuta la función '*Lanza_pulso()'* para mandar el tren de pulsos. Tras esto se cambia el puerto ECHO a INPUT para recibir las ondas emitidas previamente.

Se realiza el mismo proceso para seleccionar el puerto del ECHO correspondiente en el multiplexor, esta vez, añadiendo 1 al valor de 'sonar'. Una vez se ha seleccionado comienza el cronometrado del tiempo el cual se realiza mediante la función estándar '*pulseIn(X, Y)'* , para medir el tiempo que esta el pin 'X' en el estado 'Y'. En este caso X es el pin correspondiente al ECHO e Y es el estado HIGH, puesto que el sensor permanece en alto hasta que recibe el tren de pulsos de vuelta.

Todo lo ejecutado da como resultado un valor de 'distancia' que, teóricamente, varía entre 20 y 4.000 mm (medida de sensor 0,1-22 ms). Para que se obtengan estos valores, evitando '*erratas'* en las medidas se ha de colocar un filtro a la hora de capturar el valor medido. El filtro escogido es un bucle condicionado en el cual, si el valor obtenido es 0 o mayor que la distancia máxima que puede medir no se captura y se re ejecuta el proceso.

Una vez obtenido un valor adecuado con el rango de aceptación, se le asigna al registro de la tabla de tiempos de sensores correspondiente.

- ***Lanza_pulso ():*** función ejecutada en '*Lee_ultrasonidos(x,y)'* para mandar una señal al sensor correspondiente a través de la puerta del TRIGGER de manera que este lance un tren de pulsos que, posteriormente, se utilizara para medir la distancia.

  En primer lugar, se manda una señal de valor 0 para limpiar la salida; después de ejecuta un pulso de 10 µs y, finalmente, se vuelve a limpiar la salida para continuar con la ejecución del programa padre.

Una vez se ha presentado el esquema general con las funciones auxiliares utilizadas en el programa, se procede a explicar el funcionamiento de programa principal:

Para empezar, al ejecutar el programa, este entre en un bucle infinito definido por la funcion '*loop'* en la que se incluye la lectura de datos desde la aplicación, la selección de los sensores correspondientes a la lectura y la llamada a la función '*aparca ()'* que inicia el estacionamiento.

- ***Aparca ():*** esta función separa la fase de posicionamiento inicial, donde prima la precisión de los sensores, de la fase de búsqueda de hueco, donde la medición experimental de tiempos tiene una gran importancia.

- ***Posiciona (X):*** para que los cálculos sean correctos y precisos es necesario que esta función se realice lo más perfecta posible. Todo sistema comienza en un estado de reposo conocido a partir del cual realiza las maniobras programadas. En este sistema la situación inicial es desconocida, por lo que es necesario utilizar un objeto en situación de reposo con posición conocida como referencia y situar el dispositivo respecto a este, tanto en distancia como en ángulo.

  Para ejecutar las maniobras posteriores es necesario que el vehículo se situe paralelo a la referencia con una distancia previamente calculada, por lo que los pasos a seguir serán los siguientes:

- <u>Calculo y corrección del ángulo relativo a la referencia</u>.

  Mediante la medida de los dos sensores laterales colocados a una distancia conocida se obtiene dicho ángulo mediante la aplicación de la siguiente formula:

$$\beta[\mathrm{rad}] = \mathrm{atan}\left(\frac{|x2 - x1|}{\mathrm{b}}\right)$$

donde β es el ángulo existente respecto a la referencia, x2 es la distancia medida por el sensor 2, x1 la distancia medida por el sensor 1 y b la batalla del vehículo que coincide con la distancia entre los sensores. Todas las distancias han de introducirse con las mismas unidades para obtener un resultado correcto.

Dado que en la práctica los sensores no proporcionan tal precisión de cálculo se ha adaptado otra manera de realizar este proceso. En lugar de obtener el ángulo de posicionamiento inicial, se analizan las distancias obtenidas por los sensores (s_a, s_b) y se calcula la diferencia entre ambos, considerando cuál de los dos tiene un valor mayor. Se giran las ruedas en función de la comparación anterior, se ejecuta la maniobra de giro a la vez que los sensores siguen midiendo sus respectivas distancias y se calcula la diferencia entre ambas. En el momento que la diferencia calculada sea menor que un valor de referencia o nula, finaliza el proceso posicionamiento angular.

- Desplazamiento lateral.

En función del tipo de estacionamiento, el vehículo se ha de situar a una distancia determinada del objeto de referencia. Este proceso se realiza al inicio ya que existe la indeterminación de que exista o no otro lugar de referencia posteriormente.

Lo primero de todo, es determinar cómo se va a ejecutar la maniobra ya que, de esta decisión, dependerán los cálculos que se van a realizar. En este caso, la maniobra consta de un doble giro de ángulo β, que calcularemos en función de la distancia a la que esté situado el vehículo respecto a la referencia. Esta distancia es la medida por el sensor *'s_a'* al finalizar el proceso de posicionamiento angular.

La distancia correspondiente al estacionamiento en cordón, como ya se ha calculado anteriormente, es de $37{,}44$ cm. Dicha distancia se compara con la medida con el sensor y, mediante la fórmula:

$$d[cm] = 2 * (R[cm] + \frac{e}{2}) * (1 - \cos(\beta[\text{rad}]))$$

se calcula el ángulo con el que se debe ejecutar la maniobra de desplazamiento lateral.

La distancia correspondiente al estacionamiento en batería se obtiene de la simulación. Suponiendo que la maniobra no es arbitraria, sino que depende del tamaño del hueco existente, se ha de elegir una de todas las existentes en la simulación con la condición de que sea adecuada para cualquier situación. Es decir, se han de cumplir los requisitos de que exista el suficiente espacio para realizar dicha maniobra y que el hueco disponible sea de un tamaño lo más cercano a la realidad posible.

- ***Busca_sitio ().***

Tras haber posicionado correctamente el vehículo, el siguiente paso es ordenarle que encuentre un hueco de suficiente tamaño para poder ejecutar las maniobras. Este proceso se compone de tres fases definidas a continuación:

- Estado de búsqueda.

El vehículo se encuentra en un estado de detección de huecos, sin importar el tamaño de este. El mapeo de las distancias/tiempos se realiza con el sensor delantero (s_a) correspondiente al lado de estacionamiento. En el instante en que se detecta un valor mayor a la distancia que existe entre la anterior referencia y el vehículo, se pasa a la siguiente fase.

- Comprobación de hueco.

La velocidad del vehículo se ve reducida para mejorar la precisión de las medidas y evitar problemas por la inercia de este en fases posteriores. Durante un tiempo *'T_acc []'*, definido por el hueco máximo y mínimo que debe existir para que sea posible el estacionamiento, el vehículo avanza lentamente hasta que:

- Transcurre un periodo de tiempo máximo, si el hueco es lo suficientemente grande como para ejecutar la maniobra arbitraria (cordón y batería).

- Detecta un obstáculo, en cuyo caso puede ocurrir que:

→ Si el recorrido es mayor al espacio mínimo predefinido, el vehículo puede seguir con el proceso de estacionar.

→ Si el recorrido es menor al espacio mínimo, volverá a la fase de estado de búsqueda.

- Posicionamiento pre maniobra.*

Una vez se ha comprobado que el hueco encontrado es suficiente para proseguir con el proceso, el vehículo debe posicionarse respecto una referencia sobre el eje X.

Para el estacionamiento en cordón, la referencia utilizada es el parachoques trasero del vehículo delantero (estacionamiento en cordón cuando hueco existente es mayor que el mínimo y menor que el máximo).

En caso de no existir o de haber avanzado la distancia suficiente (estacionamiento en batería o en cordón cuando hueco es mayor que el máximo), se comienza a cronometrar a partir del momento que se cambia de fase.

**El proceso de posicionamiento pre maniobra se realiza en la función selección_maniobra ().*

Este proceso contiene la función adicional que ocasiona la detención del vehículo en el caso de detectar un obstáculo delante en la fase de *Estado de búsqueda*

- ***Selección_maniobra (X).***

  En esta función se realiza la selección de la maniobra correspondiente al tipo de estacionamiento seleccionado al inicio (cordón/batería) teniendo en cuenta, además, si el vehículo ha detectado o no que el hueco es lo suficientemente grande (si X = true, el hueco es mayor al máximo; si X = false, el hueco es menor que el máximo y mayor que el mínimo{cordón}).

- ***Maniobra_fila_1***

  La maniobra por defecto que se realiza para estacionar el cordón cuando el hueco es mayor que el máximo calculado. La ejecución consta de:

  1. Giro de 45 grado hacia el lado seleccionado en el inicio.
  2. STOP, parada para realizar los cambios de dirección, detener el vehículo y reinicializar variables.
  3. Giro de 45 grados hacia el lado contrario.

  Los sensores de ultrasonidos no realizan ninguna función una vez comienza este proceso. Sin embargo, la brújula digital es utilizada para calcular constantemente el ángulo del prototipo en cada instante mientras se realiza la maniobra.

- ***Maniobra_fila_2***

  Variante del proceso de *maniobra_fila_1,* la cual incluye la utilización de los sensores delantero (1) y trasero (8), que permite el estacionamiento del vehículo en un espacio más reducido. El proceso que se lleva a cabo es el siguiente:

  1. Giro de 45 grado hacia el lado seleccionado en el inicio (sin cambios respecto a *maniobra_fila_1).*

  2. Sucesión de dos estados de maniobra que se ejecutan hasta obtener una posición angular del vehículo nula, con un margen de error de 2 grados. Los estados de maniobra que se ejecutan son los siguientes:

a. *Estado_mani_1.* El vehículo retrocede, con las ruedas giradas hacia el lado contrario al de estacionamiento, hasta detectar un obstáculo con el sensor trasero a una distancia menor de 10 cm o detectar que la posición angular del vehículo ha llegado a un valor menor del rango previamente impuesto.

b. *Estado_mani_2.* El vehículo avanza, con las ruedas giradas hacia el lado de estacionamiento, hasta detectar un obstáculo con el sensor delantero a una distancia menor de 10 cm o detectar que la posición angular del vehículo ha llegado a un valor menor del rango previamente impuesto.

- ***Maniobra_bateria***

  El estacionamiento en batería contiene la misma estructura que el de cordón, salvo por los valores correspondientes a las distancias. En este caso, el avance en el posicionamiento antes de la maniobra es ligeramente mayor, debido a que el giro que realiza es más amplio, y únicamente existen dos fases:

  - Giro de 90 grados con el fin de ubicar el vehículo perpendicular a la referencia.
  - Retroceso del prototipo, una vez situado, hasta haber alcanzado una distancia definida por la maniobra arbitraria seleccionada o hasta detectar un obstáculo, a través del sensor trasero, a una distancia menor de 10 cm.

**La tabla de definición de variables está contenida en el Anexo 2.*

**El desarrollo completo del código se incluye en el Anexo 5.*

9. CONCLUSIONES Y TRABAJO A FUTURO

A lo largo del proyecto se ha construido un prototipo de vehículo eléctrico a pequeña escala capaz de estacionar de manera autónoma tanto en cordón como en batería, cumpliendo así los objetivos y alcances establecidos.

Para la realización de este proyecto se ha asumido una serie de suposiciones como son:

- Deslizamiento nulo.
- Peso del vehículo constante.
- Pendiente nula.
- Tiempo de aceleración despreciable.

Como se ha mencionado al principio, el objetivo de este proyecto es adaptar un vehículo eléctrico radiocontrol para que sea capaz de aparcar de forma autónoma bajo unas condiciones determinadas. Para ello, se han realizado múltiples suposiciones del entorno, se han despreciado ciertas variables por el hecho de trabajar a pequeña escala y los procesos se han llevado a cabo lentamente, ya que los sensores no tienen la precisión y seguridad que se pide para este tipo de procesos. Con todo esto, se pueden definir ciertos ámbitos de mejora para este prototipo en un futuro:

Trabajar con sistema GPS y sensores de alta precisión.*[3]

Implantando un dispositivo GPS y actualizando los sensores por otros de mayor precisión, se agilizaría la ejecución del proceso y facilitaría el análisis de los datos. A su vez, programar un sistema GPS junto a un giroscopio o una brújula de mayor precisión, mejoraría notablemente la fase de posicionamiento inicial y eliminaría la limitación de aparcar únicamente en lugares rectos.

Por otra parte, podría ser interesante la introducción de sensores medidores de posición angular de las ruedas.

Adaptación a otro tipo de vehículo.

Este proyecto se ha realizado con un vehículo eléctrico radiocontrol de pequeñas dimensiones de cuatro ruedas y chasis rígidos. Sería interesante adaptarlo a vehículos de mayor tamaño y/o articulados.

10. ANEXOS

ANEXO 1: ECUACIONES PARA CALCULO DE GEOMETRIA

1. $d[cm] = 2 * (R[cm] + \frac{e}{2}) * (1 - \cos(\beta[\text{rad}]))$ (cálculo de la distancia desplazada en el eje Y en función del ángulo de giro de la maniobra)

2. $\beta = \frac{v*t}{R}$ (cálculo del ángulo girado en función del tiempo de ejecución tomando v cte)

3. $\mathrm{R} = \frac{b}{\tan(\delta)}$ (cálculo del radio de giro a partir del ángulo de giro)

4. $\mathrm{C} = \beta * \mathrm{R}$ (cálculo del arco de giro del coche en función del ángulo)

ANEXO 2: TABLAS

ANEXO 2.1: Características Geométricas del Vehículo

Características Geométricas del Vehículo		Valores
b	Batalla del vehículo	20 cm
B	Longitud del vehículo	48 cm
a	Longitud parachoques	14 cm
Dsx	Distancia entre sensores eje X	17 cm
Dsy	Distancia entre sensores eje Y	10 cm
Xs	Posición sensor referencia eje X	16 cm
R	Radio de giro	48,7 cm
δ	Ángulo de giro de ruedas	23,5 °
e	Ancho del vehículo	26 cm

ANEXO 2.2: Listado de variables y constantes en maniobra Cordón-Bat.

Listado	de variables y constantes MANIOBRA CORDON-BATERIA
x' (cte)	Distancia en el eje X entre el parachoques trasero/lateral (cordón/batería) del vehículo subsiguiente y el eje trasero del prototipo.
x'' (var)	Distancia en el eje X que avanza el vehículo al realizar un giro de ángulo β
X"' (var)	Distancia en el eje X que resta entre el parachoques trasero del vehículo
avance max (var)	Avance máximo del vehículo al realizar los dos giros de ángulo β (= 45º) al estacionar en cordón
αi	Ángulo que resta para alcanzar la referencia (estacionamiento en cordón)
R' (var)	Radio de giro del punto situado en la esquina externa del parachoques trasero
R" (var)	Radio de giro del punto situado en la esquina externa del parachoques delantero
β (cte)	Ángulo de giro en maniobra
d (var)	Distancia en el eje Y que se desplaza el vehiculo al realizar una doble maniobra con un angulo de giro β
Dlat (cte)	Distancia en el eje Y que necesita el vehículo para estacionar en cordón con una maniobra sin colisionar con la pared o cualquier objeto situado en su lugar
L (var)	Tamaño del hueco encontrado en estacionamiento en cordón
Lmin (cte)	Tamaño mínimo del hueco, utilizado como comparación para determinar el final del proceso de búsqueda en cordón
Lmax (cte)	Tamaño máximo del hueco, utilizado como comparación para determinar el tipo de maniobras a efectuar
H (var)	Tamaño del hueco encontrado en estacionamiento en cordón
Hmin (cte)	Tamaño mínimo del hueco, utilizado como comparación para determinar el final del proceso de búsqueda en batería
Xci (cte)	Posición inicial eje X estacionamiento en cordón
Yci (cte)	Posición inicial eje Y estacionamiento en cordón
Xbi (cte)	Posición inicial eje X estacionamiento en batería

Ybi (cte)	Posición inicial eje Y estacionamiento en batería
MS (cte)	Margen de seguridad/Distancia de seguridad

ANEXO 2.3: Listado de variables correspondientes a electrónica.

Listado de variables correspondientes a la electrónica	
izda	Pin de salida correspondiente al giro de las ruedas a la izquierda
dcha	Pin de salida correspondiente al giro de las ruedas a la derecha
front	Pin de salida correspondiente al avance del vehículo
back	Pin de salida correspondiente al retroceso del vehículo
estado	Variable contenedora del byte enviado por bluetooth utilizado para obtener la variables IZDA, FILA, v_accionamiento mediante la comparación de esta con valores hexadecimales.
t1	Valor medido del sensor ultrasonido correspondiente al tiempo que se ha mantenido en HIGH el dispositivo hasta la recepción de la señal emitida
sonar	Valor transferido por el parámetro 'sensor', correspondiente a la puerta de E/S seleccionada a través de los pines S0 S1 S2 S3.
Sonar0	Contenedor del byte con el número de sensor en la parte alta y 0 en los cuatro bites bajos
sensor	Parámetro indicador del número de sensor que se desea activar
Angulo_ref	Posición angular inicial, una vez se ha colocado paralelo al objeto de referencia
Angulo	Posición angular relativa a la variable Angulo_ref a partir de la cual se realizan las comparaciones en cada una de las maniobras.

ANEXO 2.4: Definición de variables de lógica programada.

Función	Breve descripción
DEFINICION DE VARIABLES Y CONSTANTES GLOBALES DEL PROGRAMA	
S0	Pin de selección del multiplexor correspondiente al bit 0.
S1	Pin de selección del multiplexor correspondiente al bit 1.
S2	Pin de selección del multiplexor correspondiente al bit 2.
S3	Pin de selección del multiplexor correspondiente al bit 3.
R	Radio de giro del vehículo
b	Distancia entre sensores/Batalla del vehículo
c_vmax	Valor de la velocidad máxima, medida experimentalmente, del vehiculo
V_accionamiento	Variable para ejecutar el programa en el instante que el usuario desee
S_a	Variable para asignar número de sensor en función del lado en el que se desea estacionar. Toma valor de 2 ó 3.
S_b	Variable para asignar número de sensor en función del lado en el que se desea estacionar. Toma valor de 6 ó 7.

S_c	Variable para asignar número de sensor en función del lado en el que se desea estacionar. Toma valor de 4 ó 5. (Obsoleto en este programa, solo se incluye dicha variable en el caso de instalar dos sensores adicionales)
T_hueco_f	Tiempo de respuesta recibido por sensor para detectar hueco en estacionamiento en cordón
T_hueco_b	Tiempo de respuesta recibido por sensor para detectar hueco en estacionamiento en batería
T_despl_f	Tiempo correspondiente a la distancia a la que debe posicionarse el vehículo respecto del objeto de referencia en cordón
T_despl_b	Tiempo correspondiente a la distancia a la que debe posicionarse el vehículo respecto del objeto de referencia en bateria
T_acc[]	*Tiempos de ejecución, se define tipo matriz para facilitar lógica*
- T_acc[1]	Tiempo de ejecución correspondiente al máximo espacio en cordón
- T_acc[2]	Tiempo de ejecución correspondiente al máximo espacio en batería
- T_acc[3]	Tiempo de ejecución correspondiente al mínimo espacio en cordón
- T_acc[4]	Tiempo de ejecución correspondiente al mínimo espacio en batería (tiene el mismo valor que T_acc[2], pero es necesario definirla por la lógica llevada a cabo)
- T_acc[5]	Tiempo de ejecución de retroceso en estacionamiento en batería
- T_acc[6]	Tiempo de posicionamiento pre maniobra en cordón
- T_acc[7]	Tiempo de posicionamiento pre maniobra en batería
t[]	Matriz donde se almacenan los tiempos medidos por cada sensor en el momento de llamarlos a través de la funcion lee_ultrasonidos(i).
T_seg[]	*Tiempos de comparación de seguridad*
- T_seg[1]	Tiempo de comparación con el sensor delantero para activar el STOP de seguridad, en el caso de detectar un obstáculo en el estado de búsqueda de sitio
- T_seg[2]	Tiempo de comparación con sensores delantero y trasero en el proceso *maniobra_fila_2* para detectar los obstáculos correspondientes a cada estado.
V_arranque	Señal enviada a la función PWM correspondiente al impulso inicial necesario para alcanzar la velocidad objetivo lo más rápido posible
V_pos_ini	Señal enviada a la función PWM para mantener una velocidad lo baja posible para realizar las medidas de forma precisa, pero lo suficientemente alta como para que contrarreste las perdidas mecánicas del vehículo.
V_desplaz	Señal enviada a la función PWM para mantener la velocidad contante en la fase de desplazamiento, teniendo en cuenta que las pérdidas, al realizar ambos giros de ángulo β, aumentan.
V_b_sitio	Señal enviada a la función PWM para realizar la búsqueda de sitio de forma rápida y segura, puesto que las medidas de los sensores deben ser lo suficientemente precisas y se debe permitir ejecutar la acción de STOP de seguridad.
V_check_sitio	Señal enviada a la función PWM para comprobar que el

	hueco detectado es lo suficientemente grande. Está ligada directamente con los T_acc[1,2,3,4] por la formula L = v_check_sitio*T_acc[i]
V_pos_man	Señal enviada a la función PWM para posicionar correctamente el coche tras haber comprobado que el tamaño del hueco es suficiente
V_maniobra	Señal enviada a la función PWM para la correcta ejecución de la maniobra de giros, tanto en cordón como en batería
V_maniobra_bat	Señal enviada a la función PWM para ejecutar la maniobra de retroceso en el estacionamiento en batería

ANEXO 3: SIMULACIÓN DE MANIOBRAS DE ESTACIONAMIENTO

Para realizar las pruebas experimentales se ha de verificar que los datos introducidos y las suposiciones operan correctamente en un caso ideal. Para ello, se programa una simulación aplicando las fórmulas utilizadas en los cálculos y aproximando la trayectoria del vehículo según ciertas suposiciones:

1. Pendiente de terreno nula: la aceleración y velocidad del vehículo únicamente depende de la potencia aportada por el motor eléctrico.
2. Peso del vehículo constante: a pesar de poder recalcular la velocidad mediante la fórmula de conservación de momento lineal:

   $$p = m * v$$

   Se supondrá una masa constante, ya que la variación de la potencia efectiva debido a las perdidas mecánicas es un factor de gran importancia.

3. Adherencia máxima entre las ruedas y el terreno: se elimina cualquier movimiento relativo (deslizamiento) entre la rueda en la base del movimiento.

4. Tiempo de aceleración despreciable: al no realizar movimientos de precisión, sino que únicamente se trata de alcanzar una velocidad, los tiempos de aceleración del prototipo se podrán despreciar en los cálculos. Para ello, cada momento en el que el vehiculo se encuentra en reposo y se desea acelerar, se ejecutar una señal PWM de gran amplitud durante el tiempo suficiente para alcanzar la velocidad necesaria para después reducirla a la velocidad correspondiente al proceso en marcha.

5. Desviación de guiñada nula: se supondrá que el error en la guiñada es nulo, de modo que no será necesaria, con todo lo anterior, la corrección de la posición en Y durante la búsqueda de hueco.

6. El tamaño de los vehículos circundantes u obstáculos tienen las mismas dimensiones que el prototipo.

SIMULACION DE MANIOBRA EN ESTACIONAMIENTO EN CORDON

Referencia parachoques delantero del coche trasero (X) y la pared base (Y)											
Necesidades para programa (fijamos angulo β = 45 y variamos maniobra en funcion del espacio)											
L (cm)	Pos.ini.X	Pos.ini.Y	Dist. Seg. (cm)	α1(rad)	Avance 1	α2 (rad)	Avance 2	α3 (rad)	Avance 3	α4 (rad)	Avance 4
58	73,97046	37,4436153	5	0,745488174	36,5856825	#NUM!	#NUM!	#NUM!	#NUM!	#NUM!	#NUM!
60	75,97046	37,4436153	5	0,709634859	38,5856825	1,415720773	52,26518	#NUM!	#NUM!	#NUM!	#NUM!
62	77,97046	37,4436153	5	0,674853654	40,5856825	1,168484691	50,96129	#NUM!	#NUM!	#NUM!	#NUM!
64	79,97046	37,4436153	5	0,641015043	42,5856825	1,020399076	49,65741	1,500451924	38,81022	0,748141688	23,3602
66	81,97046	37,4436153	5	0,608010373	44,5856825	0,901849591	48,35353	1,038893971	42,5741	0,882487739	38,21437
68	83,97046	37,4436153	5	0,575747305	46,5856825	0,798963788	47,04965	0,812721207	46,33798	0,754966081	44,34631
70	85,97046	37,4436153	5	0,544146452	48,5856825	0,705996228	45,74576	0,631650633	50,10186	0,589256744	48,41029
72	87,97046	37,4436153	5	0,513138846	50,5856825	0,619896257	44,44188	0,472141641	53,86574	0,412715655	51,25073
74	89,97046	37,4436153	5	0,482663998	52,5856825	0,538803295	43,138	0,324791947	57,62962	0,230030628	53,19284
76	91,97046	37,4436153	5	0,452668388	54,5856825	0,461473185	41,83412	0,184473071	61,3935	0,039703538	54,39358
78	93,97046	37,4436153	5	0,423104275	56,5856825	0,387015303	40,53024	0,047763682	65,15738	0	62,83217
80	95,97046	37,4436153	5	0,393928743	58,5856825	0,314756169	39,22635	0	62,35233	0	62,35233
82	97,97046	37,4436153	5	0,365102928	60,5856825	0,244161836	37,92247	0	55,98068	0	55,98068
84	99,97046	37,4436153	5	0,33659139	62,5856825	0,174790244	36,61859	0	49,60904	0	49,60904
86	101,9705	37,4436153	5	0,308361591	64,5856825	0,106259912	35,31471	0	43,23739	0	43,23739
88	103,9705	37,4436153	5	0,28038346	66,5856825	0,038227845	34,01082	0	36,86575	0	36,86575
90	105,9705	37,4436153	5	0,252629021	68,5856825	0	34,14958	0	34,14958	0	34,14958
92	107,9705	37,4436153	5	0,225072087	70,5856825	0	36,14958	0	36,14958	0	36,14958
94	109,9705	37,4436153	5	0,197687981	72,5856825	0	38,14958	0	38,14958	0	38,14958

L + X' T_seg[2] Estado de maniobra 1 Estado de maniobra 2 Estado de maniobra 1 Estado de maniobra 2

En la simulación anterior se lleva a cabo el proceso de estacionamiento en cordón mediante cálculos geométricos con las distancias existentes entre el vehículo y el entorno, las posibles trayectorias de este, los diferentes valores de ángulo que va tomando en cada fase y la distancia de seguridad tomada a la hora de aproximarse a un obstáculo.

El proceso comienza una vez el vehículo ha girado los 45 grados correspondientes a la primera maniobra. Tras esto, entra en el bucle de estados los cuales están condicionados por la detección de un obstáculo en el lado correspondiente o que el ángulo final sea nulo.

Tal y como se observa, la cantidad de maniobras necesarias viene definido por el valor de las columnas de los ángulos αi, correspondientes con el ángulo que resta por realizar al finalizar dicha maniobra. Por lo tanto, para huecos mayores de 78 cm la cantidad de maniobras necesarias es de 2, para huecos mayores de 88 cm solo sería necesaria 1 maniobra, aproximadamente, y así sucesivamente. En el caso de que el ángulo αi aumente, conforme se desarrolla el proceso de maniobras, el estacionamiento no se puede llevar a cabo.

De esta manera se calculan tanto el hueco máximo como el hueco mínimo para el estacionamiento en cordón.

A continuación, se describen los diferentes parámetros y variables de la tabla:

El parámetro principal de entrada es la longitud del hueco que se calcula mediante la ecuación:

$$L = v(\text{velocidad del vehículo}) * \text{t (tiempo de ejecución del proceso de avance)}$$

en la que el tiempo de ejecución es el tiempo medido por el dispositivo en los instantes de ejecución del proceso.

Las columnas de la tabla contienen los valores de las siguientes variables respectivamente:

1. **L:** longitud del hueco para estacionar. Se calculará a partir de los datos obtenidos por los sensores. El proceso termina cuando recorre una distancia máxima determinada o encuentra un obstáculo tras recorrer una distancia mínima.

2. **Pos. Ini. X**: distancia que avanza desde que detecta hueco. Se reseteará cuando detecte que no es lo suficientemente grande.

3. **Pos. Ini. Y**: distancia a la se situará respecto a la pared en el posicionamiento inicial.

4. **Dist. Seg.**: distancia de seguridad que guardará el vehículo a la hora de realizar las maniobras de estacionamiento en cordón.

5. α[i]: ángulo final restante tras realizar la maniobra i.

Su cálculo se realiza mediante la diferencia entre el ángulo que debe avanzar para alcanzar la referencia (β, $\alpha(1)$, $\alpha(\mathrm{i}-1)\ldots$) y el ángulo que puede avanzar.

Maniobras impares.

El vehículo se desplaza marcha atrás, por lo que, tomando el punto situado en la rueda externa del eje trasero, la fórmula que describe su posición será:

$$\mathrm{L} - (\mathrm{x}''' + e * \mathrm{sen}(\alpha[\mathrm{i}-1]) + a + MS) = \ (\mathrm{R}+\mathrm{e}) * (\mathrm{sen}(\alpha[\mathrm{i}-1]) - \mathrm{sen}(\alpha \mathrm{i}))$$

Maniobras pares.

Los cálculos se realizan en el mismo punto, aunque se ha de tener en cuenta que las limitaciones no son respecto a este.

$$\mathrm{x}''' - \big((\mathrm{b}+\mathrm{a}) + MS\big) = \ \mathrm{R} * (\mathrm{sen}(\alpha[\mathrm{i}-1]) - \mathrm{sen}(\alpha \mathrm{i}))$$

Siendo $\mathrm{x}''' = Avance\ [i-1] - X'$ (espacio disponible para realizar la maniobra). Para i =1 → $\alpha[0] = \ \beta = 45º = 0{,}785$ rad

6. Avance [i]: posición en eje X del punto de referencia del vehículo (rueda externa eje trasero) al realizar cada maniobra.

Su cálculo se basa en la suma del avance previo con el que se ejecuta en el estado correspondiente.

Para el primero, el avance será fijo $Avance[0] = (R+e) * \sin\beta$, ya que la condición inicial de esta maniobra es que el vehículo se sitúe a 45° respecto a la referencia. De esta manera, al realizar dos maniobras cuyos giros tienen contraria dirección, se ha de tener en cuenta:

i. El radio de giro, como el punto de análisis se sitúa en un lateral del vehículo el radio de giro varía dependiendo de donde se sitúa el centro de giro.
ii. El signo a la hora de calcular el valor del avance. Para las maniobras pares el avance disminuye ya que se aproxima al punto de posición inicial X, mientras que en las impares se aleja de dicho punto.

Maniobras impares.

$$Avance[i] = Avance[i-1] + (R+e) * (\sin\beta - \sin\alpha[\mathrm{i}])$$

Maniobras pares.

$$Avance[i] = Avance[i-1] - R * (\sin\beta - \sin\alpha[\mathrm{i}])$$

Estos dos últimos parámetros son de gran importancia ya que definen la situación del vehículo para los cálculos correspondientes al siguiente estado.

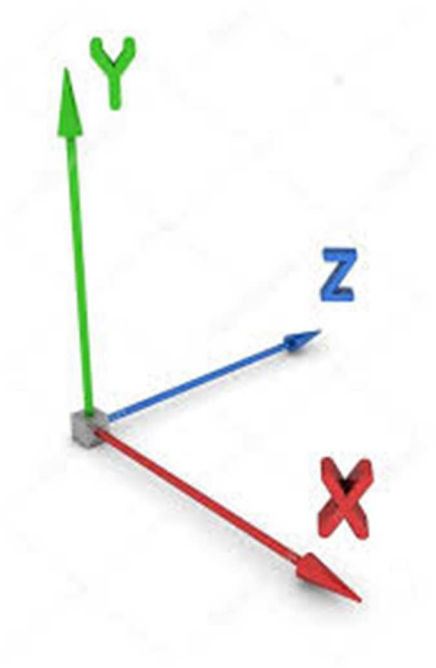
SIMULACIÓN DE ESTACIONAMIENTO EN BATERÍA			
Referencia parachoques delantero del vehículo trasero			
H	X	Y (cte)	MS
22	57,00059	No es posible	3
23	56,00059	No es posible	3
24	55,00059	No es posible	3
25	54,00059	No es posible	3
26	53,00059	No es posible	3
27	52,00059	No es posible	3
28	51,00059	40,22476519	3
29	50,00059	35,55348756	3
30	49,00059	32,21267233	3
31	48,00059	29,49497051	3
32	47,00059	27,16318636	3
33	46,00059	25,1026373	3
34	45,00059	23,24736339	3
35	44,00059	21,55524186	3
36	43,00059	19,99740293	3
37	42,00059	18,55302232	3
38	41,00059	17,20648563	3

Esta tabla simula la ejecución del proceso de estacionamiento en batería una vez el vehículo calcula el espacio disponible para estacionar. Puesto que se trata de una maniobra arbitraria, lo fundamental es hallar la posición tanto en el eje X como en Y, en la que debe posicionarse el vehículo para realizar la maniobra sin colisionar con ningún elemento, guardando siempre las distancias de seguridad indicadas.

El ancho del hueco vendrá definido desde el inicio dado que el posicionamiento del vehículo en el eje Y se realiza al iniciar el programa. Para ello, se puede buscar el valor del ancho del vehículo en una base de datos [*4] de características geométricas. Los datos del programa serán únicos para cada tipo y marca de vehículo por lo que es necesario realizar esta operación. En este caso, el vehículo posee un ancho de 26 cm, por lo que es de suponer que para valores inferiores a esa medida no se debe poder estacionar, tal y como se puede observar en la tabla. Si no muestra valor de la posición en Y al realizar los cálculos, significa que golpea a los otros elementos en la maniobra o no puede guardar el margen de seguridad impuesto.

De esta simulación se selecciona el valor de hueco que mejor se adapte al entorno en el que se desea estacionar, por ejemplo, si el espacio para maniobra es reducido se buscaran huecos de mayor tamaño y viceversa.

ANEXO 4: ORIENTACION DEL PROTOTIPO



Los datos obtenidos a través de la brújula digital son coordenadas (X, Y, Z), correspondientes a la dirección del Norte magnético de La Tierra, que se han de interpretar de forma correcta para conocer la situación del vehículo y ejecutar las maniobras sin percances.

En primer lugar, se ha de conocer el plano de trabajo en el que se mueve el dispositivo para agilizar los cálculos en el sistema operativo. En este caso, el plano de trabajo es el XZ, por lo que se excluyen los datos obtenidos del eje Y de la brújula. De esta forma se ha proyectado la dirección del Norte sobre el plano de trabajo.

Con los valores de 'X' y 'Z' obtenidos con la función *.getHeading,* correspondiente a la brújula utilizada, se calcula el ángulo existente entre el eje X (longitudinal al vehículo) y la dirección del Norte proyectado.

La forma de adaptar los datos calculados al proyecto, es utilizar esta función para comprobar, en cualquier instante, la posición angular del vehículo respecto a una referencia, impuesta por la dirección longitudinal del objeto sobre el cual se posiciona inicialmente. Siguiendo estas directrices, la formula principal utilizada es:

$$angulo_{rel} = angulo_{medido} - angulo_{ref}$$

donde angulo_rel es el ángulo medido respecto al objeto de referencia; angulo_medido es el obtenido directamente de la brújula, habiendo realizado conversión correspondiente a grados; y angulo_ref es el obtenido inicialmente en el instante en el que el vehículo se posiciona paralelamente al objeto de referencia.

Por otra parte, se han de tener en cuenta ciertas excepciones a la hora de calcular la diferencia angular entre la medida y la referencia:

- Excepción 1: Se la diferencia obtenida es mayor de 180 grados, lo cual se aparta del rango de análisis impuesto (-180° a 180°), se resta 360° al resultado, obteniendo un valor dentro del rango computable.
- Excepción 2: Si la diferencia obtenida es menor de -180 grados, lo cual se aparta del rango de análisis impuesto (-180° a 180°), se suma 360° al resultado, obteniendo un valor dentro del rango computable.

Es necesario tener en cuenta estas dos excepciones ya que los valores que se obtienen para la posición angular del vehículo en la fase de maniobras deben mantenerse en un rango de 0 a 90 grados. Por ejemplo, para un ángulo de referencia de -170 grados se desea obtener el valor relativo de un ángulo de 160 grados. Al realizar los cálculos según la fórmula anterior se obtiene:

$$angulo_{rel} = angulo_{medido} - angulo_{ref} = 160 - (-170) = 330\ grados$$

El resultado deseado debería ser de -30 grados, ya que a la hora de ejecutar la maniobra se trabaja con el valor absoluto de dicho ángulo y se compara con un valor máximo, que es donde ésta finaliza.

- Posibles fallos de medida.

  Asemejando la programación a la *'teoría'*, en el momento en que se realizan pruebas experimentales se puede observar que los valores obtenidos varían en función del ángulo de referencia escogido. Observando la *datasheet* de la brújula utilizada GY-271 se observa que, a cada eje de medida, le corresponde una amplificación de salida, es decir, si se coloca el eje del campo magnético terrestre coincidente con cada uno de los ejes, los valores obtenidos no son idénticos. Esto supone un problema a la hora de realizar las pruebas, puesto que este hecho, sumado a que campos magnéticos de otros componentes pueden alterar la medida, crea una gran incertidumbre.

ANEXO 5: CODIGO

```
//incluimos la brújula al sistema
#include "Wire.h"
#include "I2Cdev.h"
#include "HMC5883L.h"
#include<SoftwareSerial.h>

#define back 11       //pin de salida marcha atras
#define front 10      //pin de salida marcha frontal
#define izda 9        //pin de salida giro izquierda
#define dcha 8        //pin de salida giro derecha
#define RXARDUINO 3   //pin de entrada señal de aplicacion movil
#define TXARDUINO 12  //pin de entrada señal de aplicacion movil
#define Led 13        //pin de salida Led
#define Echo 2        //pin de E/S para señales de sensores de ultrasonidos
SoftwareSerial BT(RXARDUINO,TXARDUINO);

const byte s0=4;      //Señales para el multiplexorr
const byte s1=5;      //OJO el direccinamiento se hace en base a estas direcciones
const byte s2=6;      //NO SE PUEDEN CAMBIAR LOS PINES
const byte s3=7;
const int R = 40;  //ejemplo, cambiar a valor real
const int b = 17;
const int t_hueco_f = 200;
const int t_hueco_b = 500;
const int t_despl_f = 400;
const int t_despl_b = 700;
int long T_acc[10]; // tiempos de accionamiento/ejecucion, se definen en "inicializacion"



//VARIABLES
```

```
boolean v_accionamiento, IZDA, FILA;  //variables necesarias para la seleccion de maniobra y ejecucion de esta

char estado;                    //confirmacion de lectura de la señal de la aplicacion

boolean , flag1, flag2, flag3, flag4;


int16_t mx, my, mz;

int s_a, s_b, s_c, x, v_cont, T, T0, v, angulo, angulo_ref, angulo_abs;

int v_pos_ini, v_desplaz, v_b_sitio, v_check_sitio, v_pos_man, v_maniobra, v_arranque, v_maniobra_bat;


//MATRICES

unsigned int tt[5][8];    //Valores de tiempos para las medias, el t[4][i] será la última suma

unsigned int t[20], t_seg[20];     //Tiempos de referencia


//BYTES

byte indice[8];          //Indices para las medias. Indican donde se ha almacenado la última medida

HMC5883L compass;

void setup()

{

  Serial.begin(9600);


  pinMode(Led, OUTPUT);

  pinMode(Echo, OUTPUT);

  pinMode(s0, OUTPUT);

  pinMode(s1, OUTPUT);

  pinMode(s2, OUTPUT);

  pinMode(s3, OUTPUT);

  Wire.begin();

  compass.initialize();

  inicializar_var();

}

void loop()

{
```

```
 if (BT.available())
 {
   estado = BT.read();

 }
if ((estado & 0x01) == 1)
{
     v_accionamiento = true;
     if((estado & 0x02) == 1)
     {
       IZDA = true;
     }else
     {
       IZDA = false;
     }

      if((estado & 0x06) == 1)
     {
       FILA = true;
     }else
     {
       FILA = false;
     }
 }
 // Se pone en marcha el proceso
  if (v_accionamiento)
 {
     //Se adaptan los sensores a la señal que vas a recibir
     if (!IZDA)
     {
         //si ordenamos aparcar a la dcha
         s_a++;
         s_b++;
```

```
    }
    //Estacionamiento
        aparca();
        v_accionamiento = false;
  }
}

void PWM(int p_vel, int p_sentido)
{
        int l_vel;
        const int c_vmax = 150; //velocidad maxima del vehiculo
        l_vel = (p_vel*255)/c_vmax ;
        analogWrite(p_sentido,l_vel); //para velo... minimo 0 0% max 255 100%
}

void inicializar_var()
{
  PWM(0,izda);
  PWM(0,dcha);
  PWM(0,front);
  PWM(0,back);
```

```
  estado = 0;
  //velocidades en cm/s
  v_arranque = 140;
  v_pos_ini  = 60;
  v_desplaz  = 70;
  v_b_sitio  = 90;
  v_check_sitio = 70;
  v_pos_man  = 60;
  v_maniobra = 80;
  v_maniobra_bat = 70;
  //por defecto se estacionara a la izquierda
  s_a = 2;
  s_b = 4;
  //tiempos de distancias de seguridad
  t_seg[1] = 1000;  //STOP seguridad
  t_seg[2] = 400;  //distancia seguridad maniobra fila 2 y bateria
  //tiempos de ejecucion maniobras
  T_acc[1] = 2000;  //maximo espacio fila
  T_acc[2] = 1000;  //minimo espacio bateria
  T_acc[3] = 1500; //minimo espacio fila
  T_acc[4] = 1000;
  T_acc[5] = 1500;  //tiempo de ejecucion para introducir coche en bateria, linea recta
  T_acc[6] = 0;  //posicionamiento para maniobra fila
  T_acc[7] = 0; //posicionamiento para maniobra bateria

}
/*
*  Lee el valor de distancia del sensor
*/
void lee_ultrasonido(int sensor)
{
  unsigned char sonar, sonar0;
  unsigned int t1;
  sonar = 2*(sensor-1);
```

```
  t1=0;
  //Se asegura que los valores medidos por el sensor se situan en un rango de aceptacion
  while ((t1 == 0) or (t1 > 5000))
  {
     pinMode(Echo,OUTPUT);
     sonar0 = sonar<<4;          //Crea un byte con el numero de sensor en la parte alta y 0 en los cuatro bites bajos
     PORTD=(PORTD&15)+sonar0;    //Saca dirección de multiplexor para Trigger Sin cambiar parte baja del puerto
     sonar++;                    //Cambiamos numero de puerta de multiplexor
     lanza_pulso();
     pinMode(Echo,INPUT);
     sonar0=sonar<<4;            //Crea un byte con el numero de sensor en la parte alta y 0 en los cuatro bites bajos
     PORTD=(PORTD&15)+sonar0;    //Saca dirección de multiplexor para el Echo del sensor. Sin cambiar parte baja del puerto
     t1 = pulseIn(Echo, HIGH);  //calcula el tiempo de estado en alto del echo
  }
      t[sensor] = t1;            //Se almacenan los tiempos de cada sensor en su registro correspondiente
}
```

```
// Lanza el pulso de trigger del sensor de ultrasonidos
void lanza_pulso()
 {
   digitalWrite(Echo, LOW);  //para generar un pulso limpio pone a LOW 4us
   delayMicroseconds(4);
   digitalWrite(Echo, HIGH);  //genera Trigger (disparo) de 10us
   delayMicroseconds(10);
   digitalWrite(Echo, LOW); //enviado el pulso, pasa al estado de recepcion
 }
void calcular_angulo()
```

```
{
    //Se obtienen componentes del campo magnético (X, Y, Z)
   compass.getHeading(&mx, &my, &mz);
   //Se limpia la variable de medida
   angulo = 0;
   //Se calcula el ángulo del eje X respecto al norte
  int angulo_aux = round( atan2 (mz, mx) * 180/3.14159265 );
   if (angulo_ref != 0 )
   {
      if (angulo_ref > 0)
      {
        if (angulo_aux < 0) angulo = (angulo_aux - angulo_ref + 360)*2;
//En el caso de que el angulo sea negativo y la referencia positiva, se ha de restar 360 grados
        else angulo = (angulo_aux - angulo_ref)*2;
//En el caso de ser ambos positivos se realiza unicamente la diferencia
      }
      else
      {
        if (angulo_aux > 0) angulo =  (angulo_aux - angulo_ref - 360)*2;
//En el caso de que el angulo sea positivo y la referencia negativa, se ha de sumar 360 grados
        else angulo = (angulo_aux - angulo_ref)*2;
 //En el caso de ser ambos negativos se realiza unicamente la diferencia
      }
   }
}

void calcular_angulo_ref()
{
    //Obtener componentes del campo magnético
   compass.getHeading(&mx, &my, &mz);
   //Se limpia la variable de medida
   angulo_ref = 0;
```

```
    //Calcular ángulo el ángulo del eje X respecto al norte
    angulo_ref = round( atan2 (mz, mx) * 180/3.14159265 );
}
void aparca()
{
  if (IZDA)
  {
  posiciona(true);
  }
  else
  {
    posiciona(false);
  }
  busca_sitio();
}

void posiciona(boolean p_lado) // izda = true / dcha = false
{
  int l_dir, l_dir_aux, t_dif, t_dif_abs;
  if (p_lado)
  {
  l_dir = 8;
  l_dir_aux = 9;
  }
  else
  {
  l_dir = 9;
  l_dir_aux= 8;
  }
   //posicionamiento, se requiere baja velocidad y solo se realiza una vez, por lo que se
requiere precision
    lee_ultrasonido(s_a);
    delay(30);
    lee_ultrasonido(s_b);
```

```
delay(200);
//se obtiene la diferencia entre ambas distancias
t_dif = t[s_a] - t[s_b];
t_dif_abs = fabs(t_dif);
 //comparo tiempos y giro ruedas segun el lado que corresponda
 if ( t[s_a] < t[s_b] )
 {
        analogWrite(l_dir_aux,0);
        analogWrite(l_dir,255);
        PWM(v_arranque, front);
        delay(50);

        while (  t_dif_abs > 200  )
        {
              //señal PWM---se introduce la velocidad deseada en cm/s
              PWM(v_pos_ini, front);
              lee_ultrasonido(s_a);
              delay(30);
              lee_ultrasonido(s_b);
              t_dif = t[s_a] - t[s_b];
              t_dif_abs = fabs(t_dif);

        }
 //fijamos el angulo que marca la brujula como referencia
 }
 else
 {
        analogWrite(l_dir_aux,255);
        analogWrite(l_dir,0);
        PWM(v_arranque, front);
        delay(50);
        while ( t_dif_abs > 100 )
        {
              //señal PWM---se introduce la velocidad deseada en cm/s
```

```
                PWM(v_pos_ini, front);
                lee_ultrasonido(s_a);
                lee_ultrasonido(s_b);
                t_dif = t[s_a] - t[s_b];
                t_dif_abs = fabs(t_dif);
        }
//fijamos el angulo que marca la brujula como referencia
}
analogWrite(l_dir_aux,0);
analogWrite(l_dir,0);
calcular_angulo_ref();
PWM(0, front);
int d_despl_f, d_despl_b;
d_despl_f = abs((t[s_a] - t_despl_f)*340/1000);
d_despl_b = abs((t[s_a] - t_despl_b)*340/1000);

if (FILA)
{
  if (d_despl_b > 10)
  {
     if (IZDA)
     {
       if (t[s_a] < t_despl_f) desplazamiento_lateral(d_despl_f,false );
       else desplazamiento_lateral(d_despl_f,true );
     }else
     {
       if (t[s_a] < t_despl_f) desplazamiento_lateral(d_despl_f,true );
       else desplazamiento_lateral(d_despl_f,false );
     }
  }
}else
{
  if (d_despl_b > 10)
  {
```

```
          if (IZDA)
          {
            if (t[s_a] < t_despl_b) desplazamiento_lateral(d_despl_b,false );
            else desplazamiento_lateral(d_despl_b,true );
          }else
          {
            if (t[s_a] < t_despl_b) desplazamiento_lateral(d_despl_b,true );
            else desplazamiento_lateral(d_despl_b,false );
          }
       }
     }

}




void busca_sitio()
{
  int t_hueco;
  boolean flag_stop;
  //Se inicializan las variables de estado
  flag1 = true;
  flag2 = false;
  //Se selecciona el registro de la estructura T_acc[x] correspondiente al tipo de
estacionamiento
  if (FILA)
  {
    t_hueco = t_hueco_f;
    x = 1;
  }
  else
  {
```

```
  t_hueco = t_hueco_b;
  x = 2;
 }

 //Comienza la fase de busqueda de sitio para estacionar
  while (flag1)
   {
    //FASE 1:  BUSCA SITIO
     while (!flag2)
    {
      flag_stop = false;                //Se inicializa la variable de STOP de seguridad
      lee_ultrasonido(1);
      //condicion de frenar si encuentra obstaculos
      while (t[1] < t_seg[1])
      {
        flag_stop = true;               //Al detectar un obtaculo se da valor a la variable
        PWM (0, front);                 //STOP de seguridad
        lee_ultrasonido(1);
//Se comprueba que el obstáculo no exista para proseguir con la búsqueda
      }
      if (flag_stop)
      {
         PWM(v_arranque, front);        //Impulso de aceleracion
         delay(50);
      }
      PWM (v_b_sitio, front);
      lee_ultrasonido(s_a);
      if (t[s_a]>t_hueco)
        //ENCUENTRA HUECO
        flag2 = true;                   //Cambio de estado a checkeo de hueco

    }
    //Se inicializa el crono
    T0 = 0;
```

```
    T = 0;
    T0 = millis();
    PWM (v_check_sitio, front);
    //FASE 2: SE BUSCA QUE EL HUECO SEA SUFICIENTE
    while ((T < T_acc[x]) && (flag2) )
    {
       T = millis()-T0;
       lee_ultrasonido(s_a);
       if (t[s_a]<t_hueco)    //Deteccion de obstaculo en el lado de estacionamiento
       {
         if (T<T_acc[x+2])    //Se compara el espacio recorrido con el minimo que ha de
existir
         {
           flag2 = false;     //Se cancela estacionamiento
         }
         else
         {
           seleccion_maniobra(false);   //Se estaciona de la forma 2
         }
       }



    }
    if (flag2)
    {
    seleccion_maniobra(true);    //Se estaciona de la forma 1


    }
  }
}


void seleccion_maniobra(boolean lim_max)
{
  int t_pos;
```

```
  if (FILA) t_pos = T_acc[6];
  else t_pos = T_acc[7];
  PWM (v_pos_man, front);
  T0 = 0;
  T = 0;
  T0 = millis();
  while ( T<t_pos )
    {
       T = millis()-T0;
    }
   flag1 = false;       //Cambio de estado para salir del bucle de 'busca sitio' al terminar
   PWM (0, front);
  switch (x)           //Seleccion de maniobra en funcion del tipo de estacionamiento
pedido y del espacio medido
  {
    case 1:
      if (lim_max) maniobra_fila_1();
      else maniobra_fila_2();
      break;
    case 2:
      maniobra_bateria();
      break;
  }


}



void desplazamiento_lateral(int distancia, boolean lado) //lado: true izda, false dcha
{
  int angulo_desp;
  angulo_desp = acos((1-(distancia-(distancia*(R+b)-b)/(2*R+b)))/R)*180000/31415;
    if (!lado) analogWrite(dcha,255); //GIRA RUEDAS dcha
    else  analogWrite(izda,255); //GIRA RUEDAS izda
    calcular_angulo();
```

```
    angulo_abs = fabs(angulo);
    while (angulo_abs < angulo_desp)
    {
        PWM(v_desplaz, front);
        calcular_angulo();
        angulo_abs = fabs(angulo);
    }
    PWM(0, back);
    analogWrite(dcha,0);
    analogWrite(izda,0);

    if (!lado) analogWrite(izda,255); //GIRA RUEDAS dcha
    else  analogWrite(dcha,255); //GIRA RUEDAS dcha
    while (angulo_abs > 2)
    {
        PWM(v_desplaz, front);
        calcular_angulo();
        angulo_abs = fabs(angulo);
    }
    analogWrite(dcha,0);
    analogWrite(izda,0);

}

void maniobra_fila_1()
{ //INICIO MANIOBRA: RETROCEDE CON GIRO 45º dcha, GIRO 45º izda Y STOP
        analogWrite(dcha,0);
        analogWrite(izda,0);
        if (!IZDA) analogWrite(dcha,255); //GIRA RUEDAS dcha
        else  analogWrite(izda,255); //GIRA RUEDAS izda
        calcular_angulo();
        angulo_abs = fabs(angulo);
        PWM(v_arranque, back);
        delay(50);
```

```
        while (angulo_abs < 45)
        {
            PWM(v_maniobra, back);
            calcular_angulo();
            angulo_abs = fabs(angulo);
        }
        PWM(0, back);
        analogWrite(dcha,0);
        analogWrite(izda,0);

        if (!IZDA) analogWrite(izda,255); //GIRA RUEDAS dcha
        else  analogWrite(dcha,255); //GIRA RUEDAS dcha
        PWM(v_arranque, back);
        delay(50);
        while (angulo_abs > 2)
        {
            PWM(v_maniobra, back);
            calcular_angulo();
            angulo_abs = fabs(angulo);
        }
        analogWrite(izda,0); //STOP
        analogWrite(dcha,0); //STOP
        PWM(0, back);
}

void maniobra_bateria()
{ //GIRO 90º, MARCHA ATRAS Y STOP
  //falta avance de distancia determinada en excel
        analogWrite(dcha,0);
        analogWrite(izda,0);
        if (!IZDA) analogWrite(dcha,255); //GIRA RUEDAS dcha
        else  analogWrite(izda,255);   //GIRA RUEDAS izda
        calcular_angulo();
        angulo_abs = fabs(angulo);
```

```
        PWM(v_arranque, back);
        delay(50);
        while (angulo_abs < 87)
        {
            calcular_angulo();
            angulo_abs = fabs(angulo);
            PWM(v_maniobra, back);
        }
        analogWrite(dcha,0);
        analogWrite(izda,0);
        PWM(0,back);
        delay(500);
        T0 = 0;
         T = 0;
        T0 = millis();
        PWM(v_arranque, back);
        delay(50);
        while (T < T_acc[5] or t[8] < t_seg[2])
        {
                T = millis()-T0;
                PWM(v_maniobra_bat,back);
        }
         PWM(0,back);
}
/*
añadimos flag4:
- true para cambio de estado: atras dcha -->delante izda
-false para cambio de estado: delante izda --> atras dcha

*/
void maniobra_fila_2()
{
   if (!IZDA) analogWrite(dcha,255);//GIRA RUEDAS dcha
   else  analogWrite(izda,255);   //GIRA RUEDAS izda
```

```
    calcular_angulo();
    angulo_abs = fabs(angulo);
    PWM(v_arranque, back);
    delay(50);
    while (angulo_abs < 45)
    {
        calcular_angulo();
        angulo_abs = fabs(angulo);
        PWM(v_maniobra, back);
    }
    PWM(0, back);
    analogWrite(dcha,0);
    analogWrite(izda,0);
    flag4 = false; //empieza con ruedas a la dcha y marcha atras
    while (angulo_abs>6)
    { //condicion de que si no esta paralelo a la 'pared' no para
      calcular_angulo();
      angulo_abs = fabs(angulo);
      if (!flag4) estado_mani1(); //dcha atras
      else estado_mani2(); //izda delante
    PWM(0, back);
    PWM(0, front);
    delay(1000);
    }
}

void estado_mani1()
{
    if (IZDA) analogWrite(dcha,255); //GIRA RUEDAS dcha
    else  analogWrite(izda,255);   //GIRA RUEDAS izda
    T0 = 0;
    T = 0;
    T0 = millis();
    PWM(v_arranque, back);
```

```
    delay(50);
    while (angulo_abs > 6 and !flag4)
    {
      T = millis()-T0;
      PWM(v_maniobra, back);
      lee_ultrasonido(8);
      if (t[8]<t_seg[2])
        flag4 = true; //deteccion de obstaculo trasero, cambio estado
    }
    analogWrite(dcha,0);
    analogWrite(izda,0);
}


void estado_mani2()
{
    if (IZDA) analogWrite(izda,255); //GIRA RUEDAS izda
    else  analogWrite(dcha,255);   //GIRA RUEDAS dcha
    T0 = 0;
    T0 = millis();
    T = 0;
    PWM(v_arranque, front);
    delay(50);
    while (angulo_abs > 6  and flag4)
    {
      T = millis()-T0;
      PWM(v_maniobra, front);
      lee_ultrasonido(1);
      if (t[1]< t_seg[2])
        flag4 = false; //deteccion de obstaculo delantero, cambio estado
    }
    analogWrite(dcha,0);
    analogWrite(izda,0);
}
```

11. REFERENCIAS

(Ordenadas por orden de aparición)

1. http://seyllosa.es/blog/72-estacionamiento-autonomo-por-movil
2. https://www.luisllamas.es/brujula-magnetica-con-arduino-compass-digital-hmc5883/
3. http://www.recambiooriginal.com/blog/recambios-originales/mecanica/avances-tecnologicos-aparcamiento-autonomo/
4. http://www.medidasdecoches.com/simulador-medidas-aparcamiento.php