Universidad Zaragoza
1542

Trabajo Fin de Grado

# Desarrollo de un tipo de vehículo con asistencia a la conducción

# Development of a type of vehicle guided in driving


Autor

Enrique Hernández Murillo

Director

Daniel Mercado Barraqueta

ESCUELA DE INGENIERÍA Y ARQUITECTURA

UNIVERSIDAD DE ZARAGOZA


2017

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).

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

D./Dª. ENRIQUE HERNÁNDEZ MURILLO

, con nº de DNI. 73130746S 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 EN ING. DE TECNOLOGÍAS INDUSTRIALES, (Título del Trabajo) DESARROLLO DE UN TIPO DE VEHÍCULO ASISTIDO EN LA CONDUCCIÓN

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

Zaragoza, 10 DE AGOSTO DE 2017

Fdo: ENRIQUE HERNÁNDEZ MURILLO

# Agradecimientos

*A mí padre Enrique.*

*A mí madre Luisa y mi novia Marta.*

*A mi familia y amigos, por ser los pilares fundamentales en mi proyecto de vida.*

*A Juan por sus inestimables aportaciones.*

*A mi director Daniel, sin cuyos consejos y orientaciones este trabajo no hubiese sido posible.*

# DESARROLO DE UN TIPO DE VEHÍCULO CON ASISTENCIA A LA CONDUCCIÓN

## Resumen

Desde el siglo pasado, se ha venido produciendo una auténtica revolución, impulsada por la robótica, en los paradigmas de modelos productivos, hasta desembocar en la actual Industria 4.0. Así es posible destacar cuatro aportaciones de sus aportaciones más fundamentales: el aumento de productividad en las tareas, una alta flexibilidad, excelente calidad y una sustancial mejora en la seguridad. Una de las áreas de la robótica, que mayor expansión ha tenido, es el control de vehículos autónomos.

En este proyecto se ha realizado una aplicación móvil, que permite el guiado de un vehículo de ruedas, en los modos manual y automático, así como la disposición electrónica adecuada para el control del mismo. Para la implementación de la aplicación móvil se ha elegido, como sistema más apropiado para la transmisión de la información entre el dispositivo móvil y el vehículo, la comunicación inalámbrica por radiofrecuencia basada en la tecnología para redes personales Bluetooth. El sistema operativo adoptado, para la programación en el móvil, ha sido ANDROID por ser universalmente conocido y tener una implantación extensa en modelos y marcas. En cuanto a la monitorización, al objeto de detectar y evitar obstáculos, se ha llevado a cabo utilizando sensores de ultrasonidos y para el control del vehículo y guiado, se ha empleado el microcontrolador Arduino UNO, por sus características de comunicación serie. Cabe destacar, que el coche venía provisto del sistema mecánico, los motores, además del puente H y la batería. Finalmente, las pruebas realizadas en el laboratorio, han dado como resultado un correcto funcionamiento del sistema.

# DEVELOPMENT OF A TYPE OF VEHICLE GUIDED IN DRIVING

## Abstract

Since the last century, an authentic revolution has been taking place, driven by robotics, in the paradigms of productive models, until leading to the current Industry 4.0. Thus it is possible to highlight four contributions of its most fundamental contributions: the increase in productivity in the tasks, a high flexibility, excellent quality and a substantial improvement in security. One of the areas of robotics, which has had the most expansion, is the control of autonomous vehicles.

In this project a mobile application has been made, which allows the guidance of a wheeled vehicle, in manual and automatic modes, as well as the adequate electronic layout for the control of it. For the implementation of the mobile application has been chosen, as the most appropriate system for the transmission of information between the mobile device and the vehicle, radio-frequency wireless communication based on technology for personal Bluetooth networks. The operating system adopted, for mobile programming, has been ANDROID because it is universally known and has an extensive implementation in models and brands. In terms of monitoring, in order to detect and avoid obstacles, it has been carried out using ultrasonic sensors and for the control of the vehicle and guidance, the Arduino UNO microcontroller has been used, due to its serial communication characteristics. It should be noted that the car came with the mechanical system, the engines, as well as the H-bridge and the battery. Finally, the tests carried out in the laboratory have resulted in the correct functioning of the system.

# Índice

# 1. Introducción.

## 1.1. Robótica móvil

La Robótica es un campo multidisciplinar que pertenece al ámbito de la ingeniería y que posee un extenso campo de aplicaciones, desde manipulación hasta navegación. Por un lado los robots pueden adquirir una elevada complejidad según sus características esperadas y su funcionalidad, permitiendo un área de investigación muy grande. Por otro el libre acceso a librerías de "código abierto" y a los bajos precios del hardware electrónico, hacen posible la construcción de robots sencillos por prácticamente cualquier persona.

Una de las áreas en la que más desarrollo se está produciendo, tanto a nivel industrial como de ocio es el de la robótica móvil. A diferencia de los robots manipuladores cuya base es fija, en los robots móviles el movimiento de su base está habilitado, lo que les dota de gran autonomía. A pesar de las ventajas que presenta, hay que destacar que trabajar en entornos hostiles, no estructurados y con grandes incertidumbres en cuanto a la posición y orientación, conlleva una dificultad añadida con respecto a los robots estáticos cuyos entornos, en definitiva, son conocidos.

Se dice de un vehículo es holonómico cuando la libertad de sus movimientos es igual a la libertad global del prototipo. Un coche es noholonómico porque posee tres grados de libertad globales, sin embargo no puede desplazarse de forma lateral, si quiere girar tiene que avanzar o retroceder, lo que introduce una cierta complejidad en la generación de trayectorias para desplazarse en un entorno con obstáculos. Dentro del campo de los vehículos no holonómicos podemos encontrar varios tipos de configuraciones, si bien este proyecto está centrado en una sola: los vehículos con ruedas, los cuales utilizan una serie de motores para mover dos o más ruedas según su morfología. Será necesario pues un sistema de monitorización y guiado, capaz de percibir las características del entorno, es decir la posición y orientación del robot respecto de un sistema de referencia absoluto o relativo, de forma que sea capaz de trazar trayectorias y evitar obstáculos. A este fin, Se pueden encontrar dos tipos de medición: de la posición relativa, esto es, mediante sensores incluidos en el robot, que

calculan velocidad, distancia y aceleración; de la posición absoluta, a través de estaciones de transmisión fijas o móviles de posiciones conocidas.

Este proyecto centra su objetivo en el control remoto vía Bluetooth mediante la interactuación del usuario, y en el control necesario para el seguimiento de la situación de cada obstáculo. Para ello se utilizan sensores ultrasónicos en el vehículo móvil, un módulo de comunicaciones Bluetooth, un dispositivo móvil y la plataforma Arduino.
En lo que se refiere al control del vehículo móvil, se hace una descripción del esquema general de éste y explica las tres jerarquías de control, y sus funciones. Además, se expone como es capaz de reaccionar, ante los cambios inesperados en el entorno, de forma rápida.

## 1.2. Instrumentación y Control

En lo que se refiere al control de un robot móvil, el Capítulo 5 hace una descripción del esquema general de éste y explica las tres jerarquías de control, y sus funciones. Además, se expone como el robot es capaz de reaccionar ante los cambios inesperados en el entorno de forma rápida. Este proyecto focaliza su objetivo en el control remoto vía Bluetooth mediante la interactuación del usuario, así como, del control necesario para el seguimiento de la pared. Para ello se utilizan sensores ultrasónicos, el prototipo móvil, un módulo de comunicaciones Bluetooth, un dispositivo móvil y la plataforma Arduino.

Una parte indispensable en la robótica móvil, son los sensores, al igual que el control para producir la trayectoria esperada. Así mismo, dotan al controlador del robot de información acerca del entorno y del estado de él mismo. Para ser más específicos, se realiza una clasificación de los sensores, en función de si son propioceptivos, que obtienen información interna del prototipo como la temperatura, batería, etc. o exteroceptivos, que son aquellos que toman información del entorno permitiendo que el robot evite chocarse con obstáculos por ejemplo.

Además, la función principal de la instrumentación es la de obtener la posición y orientación del robot respecto de un sistema de referencia, absoluto o relativo, de forma que el sistema sea capaz de trazar trayectorias y evite obstáculos. Se pueden encontrar dos tipos de medición. En primer lugar, la medición de la posición relativa, esto es, mediante sensores incluidos en el robot, que calculan velocidad, distancia y aceleración. Por otro lado, se encuentra la medición de la posición absoluta a través de estaciones de transmisión fijas o móviles de situaciones conocidas.

## 1.3. Motivación del trabajo

La idea de la realización de este Trabajo Fin de Grado surge en la asignatura de Simulación y Análisis de Sistemas Mecánicos en Mecatrónica, más concretamente de Daniel Mercado, actual profesor de la asignatura y director de mi proyecto. Las motivaciones para la realización del presente trabajo han sido varias:

- Por una parte la necesidad inherente, en base al avance tecnológico, de estudiar, asimilar e interiorizar un lenguaje de programación como Java y también Android, de la forma más directa y rápida posible.
- En segundo lugar encontrarme inmerso en un proyecto tecnológico sobre la industria 4.0 para la Universidad de Zaragoza, donde realizo mis actividades estudiantiles, y que me ha proporcionado el acceso a toda la información y los dispositivos tanto mecánicos como electrónicos con los que desarrollar mi proyecto.
- En tercer lugar el interés en poner en práctica los conocimientos y aptitudes adquiridas a lo largo del grado, especialmente en las asignaturas que me van a ser útiles de cara al futuro.

## 1.4. Objetivos

El presente proyecto es parte de un conjunto, formado por dos TFG, para el *Desarrollo de un Prototipo de Vehículo con Asistencia a la Conducción*. Cabe notar que el procedimiento para ambos ha sido el mismo, el prototipo ha de ser capaz de combinar acciones controladas y automáticas, apoyándose de una red de sensores y comunicaciones vía Bluetooth, para el correcto desarrollo y su objetivo final.

El objeto del presente trabajo es la realización de un vehículo controlado a distancia, que sea capaz de detectar y esquivar obstáculos de forma autónoma. Para ello, este vehículo estará basado en un coche de radiocontrol modificado para ser controlado desde un teléfono móvil.

La problemática asociada al proyecto viene dada por la generación de un sistema de control que gestione el movimiento del coche y que permita detectar y esquivar los posibles obstáculos en la trayectoria del mismo mediante guiado manual y automático a través de una aplicación de telefonía móvil.

Dicho prototipo debe incluir un controlador que interprete las órdenes de un dispositivo externo, para el funcionamiento de los motores y dirección del coche, así

como un módulo Bluetooth encargado de recibir y transmitir dicha información. Por último, se deberá disponer de los sensores necesarios para su control.

## 1.5. Alcance

Adicionalmente, se exponen a continuación los principales puntos del alcance del proyecto:

- Realizar un prototipo, que siendo operado a distancia sea capaz de detectar y esquivar obstáculos. Para ello, se parte de un coche de radiocontrol al que se anulará la parte de radio y se dotará de un sistema de comunicación inalámbrico.
- Controlar desde un dispositivo Android el vehículo, a través del mencionado sistema inalámbrico, así como el acoplamiento de un sistema de detección y control.
- Generación de un programa para Android, desde el IDE de Processing, para la realización del emisor.
- Originar un conjunto de detección, control y recepción, mediante un microcontrolador de la familia Arduino.
- Dotar al grupo mecatrónico de los sensores correspondientes.

## 1.6. Metodología a seguir

Para el desarrollo de este trabajo se decidió seguir una metodología en cascada, donde se establecieron unos objetivos fijos, donde antes de pasar a la siguiente etapa se ejecutaba la anterior y se aseguraba del correcto funcionamiento. El presente trabajo se ha estructurado en siete capítulos:

En primer lugar, se hace una introducción general sobre la robótica y se plantean los objetivos del proyecto; hasta aquí el Capítulo 1.

En segundo lugar, en el Capítulo 2, se ha dedicado a hacer una revisión del estado del arte, sobre los robots móviles, concretamente acerca del vehículo Ackerman, así como del concepto de teledirección, es decir, el control a distancia del vehículo móvil. Además, se hace mención a las comunicaciones inalámbricas, y se aborda el tema de la transferencia de datos.

Se aborda un primer análisis mecánico del vehículo en el Capítulo 3, se detallan los elementos mecánicos que lo componen y también se elabora un modelo, sobre el

cuál se va a trabajar a lo largo de todo el proyecto. El Capítulo de desdobla en dos partes, en las que se pretende simplificar dicho análisis, y cabe decir, que se ha centrado más en la parte del control direccional, que en la motriz.

Seguidamente, se describe la arquitectura hardware del sistema móvil en el Capítulo 4. Se detallan la elección de los componentes que conforman el prototipo para finalmente acabar haciendo el montaje del mismo.

A continuación se pasa a detallar, en el Capítulo 5, el método empleado en la generación de un programa receptor y la realización de las pruebas.

En el Capítulo 6, se describe la selección llevada a cabo para determinar la comunicación entre ambos sistemas, emisor y receptor, mediante vía inalámbrica, en particular, mediante un módulo Bluetooth. El Capítulo está hecho de forma que la comunicación se explique desde ambos puntos de vista; emisor y receptor.

En cuanto al Capítulo 7, una vez descritos entrambos sistemas (recepción y comunicación), el siguiente paso del proyecto es la emisión, es decir, el Capítulo 7 explica cómo se ha llevado a cabo la aplicación móvil, que interactúa con el usuario, y desde la cual se controla el vehículo.

Finalmente, en el último Capítulo, se extraen las conclusiones derivadas del trabajo, de acuerdo a los objetivos expuestos en el primer capítulo. Por último y debido a las distintas alternativas planteadas durante la elaboración del presente trabajo, se proponen algunas posibles modificaciones futuras relacionadas con el sistema de comunicación inalámbrico.

# 2. El estado del arte

En el presente capítulo se hace una revisión de los aspectos más importantes que conciernen a esta rama de la robótica. En primer lugar se hace una revisión histórica, seguidamente se efectúa una clasificación por su forma de locución y se analizan las diferentes partes que lo componen, finalmente se abordan los temas concernientes a posicionamiento y trayectoria.

## 2.1. Los robots móviles

Se puede definir un robot móvil como "*un sistema electromecánico autónomo, capaz de desplazarse mediante algún dispositivo de locomoción, que emplea sensores que le proporcionan un cierto control sobre su posición*".

Tal y como se deduce de la definición, este tipo de robot debe de desplazarse por lo que necesitará un sistema de locomoción apropiado; también deberá sortear obstáculos que tendrá que detectar mediante algún tipo de dispositivo que le dote de sentidos; también necesitara modificar su trayectoria por lo que tendrá que tener una mínima capacidad para tomar decisiones, es decir será necesario una cierta inteligencia artificial.

### 2.1.1. Evolución histórica

Aunque la robótica tiene antecedentes en los autómatas del siglo XVIII, fue a finales de los años sesenta y la década de los setenta del siglo pasado cuando se produjo un mayor crecimiento en cuanto a la investigación en robótica móvil [1], ejemplos destacados de esta década fueron los robots: Newt desarrollado por Hollis (16) como se muestra en la figura 1, Hilare construido en el LAAS-NSR (Laboratoire d'analyse et d'architecture des systèmes) en Toulouse o Shakey desarrollado por Nilsson (14) en el que por primera vez se emplea la inteligencia artificial para el control de movimientos. En los laboratorios del JPL (Jet Propulsión Laboratory) se desarrolla el robot de exploración planetaria Lunar Rover, como se muestra en la figura 2.1.

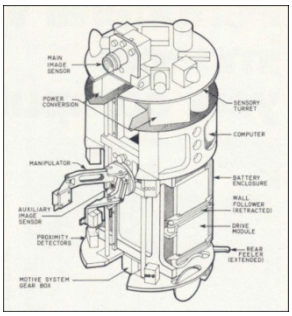


***Figura 2.1***. *Robot Newt. [2]*

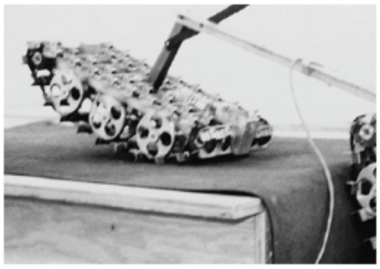

***Figura 2.2.*** *Robot de exploración planetaria Lunar. [3]*

A comienzos de la década de los ochenta; Moravec desarrolla el Standford Cart, robot capaz de seguir una línea recta dibujada en suelo y Raibert desarrolla en el MIT un robot de una sola pata, ver figura 2.8.

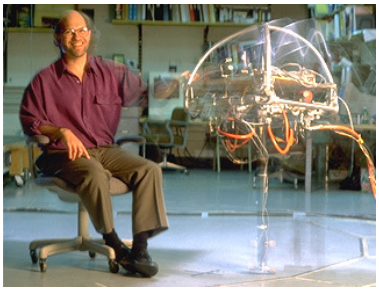

***Figura 2.3.*** *Robot que sigue una recta* [4]

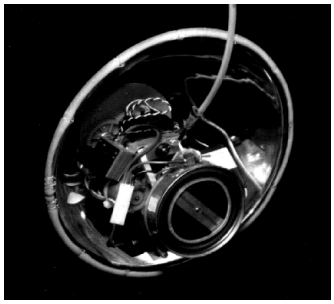

***Figura 2.4.*** *Gyrover*. [5]

***Figura 2.5.*** *Dante II.* [6]

Durante la década de los noventa Koshiyama y Yamafuji inician el estudio de los robots con sistema de desplazamiento basado en una esfera, este se desplaza en la dirección deseada, pues no tiene restricciones, consiguiendo el equilibrio dinámicamente. En la CMU (Carnegie Mellon University) se desarrollaron Dante II un sistema de seis patas, ver figura 2.5 y Gyrover un mecanismo giroscópico, sin ruedas ni patas muy estable y en el MIT se desarrolló Spring Flamingo un robot con patas que imitaba el movimiento de un flamenco.

A finales de los noventa la NASA envió a marte el Sojourner Rover, robot con ruedas mostrado en la figura 2.6 para la exploración del planeta. Por su parte Honda presenta su robot de tipo humanoide P3, mostrado en la figura 2.7. En la misma línea de robots humanoides, la universidad de Waseda en Japón emplea el robot Wabian RIII para analizar los movimientos del cuerpo humano. Zeglin en un empeño por

reducir el consumo de energía, crea un diseño que aprovecha la energía potencial que acumulaba una única pata curva.

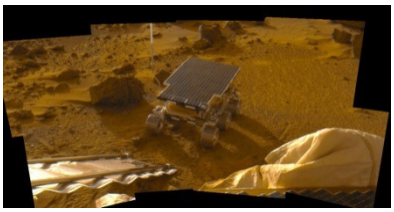

***Figura 2.6.*** *Sojourner Rover.* [7]

***Figura 2.7.*** *Robot humanoide P3.* [8]

***Figura 2.8.*** *Zeglin.* [9]

En 2005 el Prof. Ralph Hollis del CMU desarrollo un robot de tamaño humano resistente a perturbaciones tales como patadas, empujones y colisiones. A partir de este El Prof. Masaaki Kumagai desarrolló *BallIP* en 2008 [10], demostrando su utilidad para llevar cargas y Tomas Arribas desarrollo un robot de bola con LEGO Mindstorms NXT en la Universidad de Alcalá.

***Figura 2.9.*** *Curiosity.* [12]

***Figura 2.10.*** *Robot LEGO Mindstorms.* [11]

Actualmente el vehículo de exploración Curiosity, que se muestra en la figura 2.9, sustituyo a los robots teleoperados Spirit y Opportunity en la exploración de la superficie del planeta Marte.

## 2.1.2. Vehículos con ruedas

Los robots móviles se clasifican por la técnica que emplean para moverse. Los medios de locomoción más empleados son; por ruedas, patas, orugas o balón [13].

Todos estos tipos de robot han sido objeto de una amplia investigación, y si bien cada uno de ellos tiene sus ventajas, cabe destacar las de los robots móviles con ruedas, porque ofrecen una serie de características que los hacen más interesantes que los demás: menor complejidad, eficiencia energética y menor incidencia en la superficie donde se mueven.

Existen diferentes tipos de robot con ruedas que se diferencian fundamentalmente por su configuración cinemática [14].

**Rueda diferencial**: este es el mecanismo de control más común para los constructores de robots, especialmente para principiantes. La diferencia de velocidad entre dos motores impulsa al robot hacia cualquier dirección requerida. De ahí el nombre de unidad "Diferencial", como se muestra en la figura 2.11.

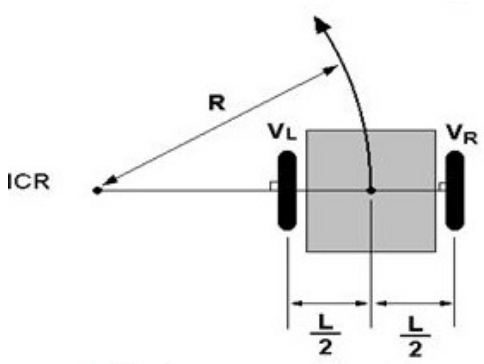


***Figura 2.11.*** *Robot con ruedas diferencial.* [14]

**Skeed steer:** es otro mecanismo de conducción implementado en vehículos con vías o ruedas que utiliza un concepto de accionamiento diferencial. El giro se realiza generando una velocidad diferencial en el lado opuesto de un vehículo ya que las ruedas o pistas en el vehículo no son dirigibles.

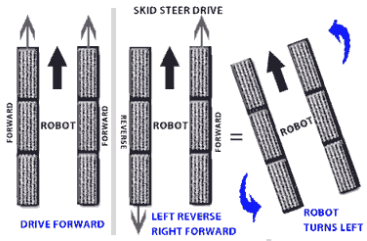


***Figura 2.12.*** *Skeed steer.* [15]

**Triciclo** : diseñado con un volante delantero controlado por un motor. Las dos ruedas traseras están unidas a un eje común accionado por un solo motor con dos grados de

libertad (ya sea hacia adelante o hacia atrás). La desventaja es que no pueden girar como un robot de accionamiento diferencial y no pueden girar 90 ° debido a que su radio de curvatura está impedido mecánicamente. Debido a las restricciones de diseño y sus inconvenientes, este diseño es menos apreciado por la comunidad de construcción de robots.

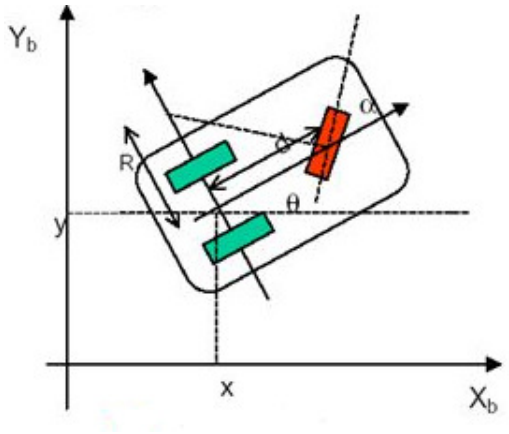


***Figura 2.13.*** *Triciclo.* [14]

**Ackermann:** una de las configuraciones más comunes que se encuentran en los automóviles es la configuración de Ackerman, que coordina mecánicamente el ángulo de dos ruedas delanteras que están fijas a un eje común utilizado para la dirección y dos ruedas traseras fijadas en otro eje para la conducción.

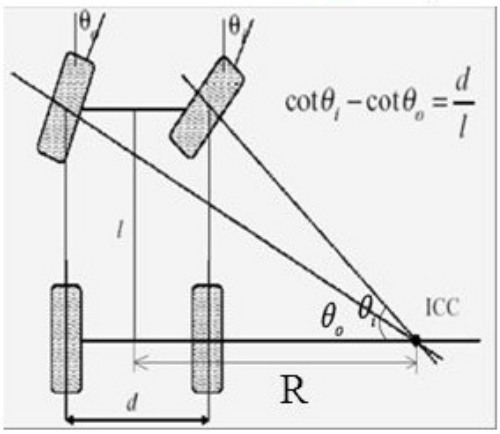


***Figura 2.14.*** *Ackermann.* [14]

La ventaja de este diseño es un mayor control, una mejor estabilidad. La dirección de Ackerman está diseñada de tal manera que cuando hay un viraje, el neumático interior gira con un ángulo mayor que el neumático exterior y evita el deslizamiento del neumático.

Este enfoque se puede usar generalmente para robots rápidos al aire libre que requieren una excelente distancia al suelo y tracción. Aunque hay desventajas, la desventaja es que se principal es que se requiere un diseño más complejo.

**Tracción síncrona**: para superficies uniformes, este diseño se puede considerar el mejor ya que la navegación es muy precisa. Puede construirse con tres, cuatro o cualquier número de ruedas, pero todas las ruedas deben girar juntas en la misma dirección y a la misma velocidad. Más de tres ruedas significarían una complejidad adicional y, por lo tanto, este método es mejor para los robots con tres ruedas en una superficie pareja.

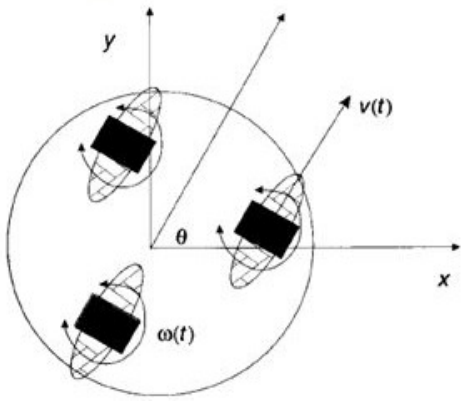


***Figura 2.15.*** *Tracción síncrona.* [14]

**Transmisión omnidireccional:** los robots omnidireccionales están fabricados con ruedas Omni. Como las ruedas Omni tienen ruedas más pequeñas unidas perpendicularmente a la circunferencia de otra rueda más grande, permiten que las ruedas se muevan en cualquier dirección al instante. La principal ventaja es que no necesitan girar para moverse en ninguna dirección a diferencia de otros diseños. En otras palabras, son robots holonómicos y pueden moverse en cualquier dirección sin cambiar la orientación.

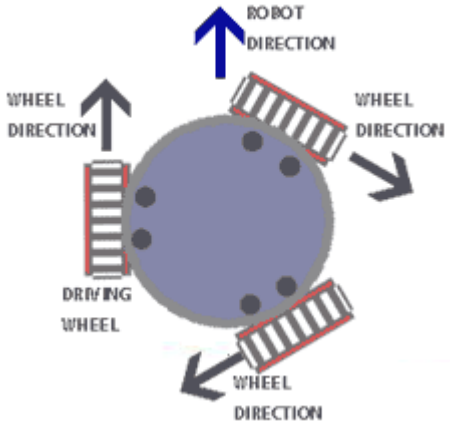


***Figura 2.16.*** *Transmisión omnidireccional.* [15]

## 2.2. Posicionamiento

Leonard y Durrant-Whyte [17] resumieron el problema de navegación de un robot móvil mediante tres preguntas : “¿Dónde estoy ?”, “¿A dónde voy ?” y “¿Cómo debo llegar allí?”. En su análisis indica que a falta de una solución global al problema, existen soluciones parciales que pueden categorizarse en dos fundamentales, que normalmente se combinan; las basadas en la medición de la posición relativa, y las empleadas en la medida de la posición absoluta.

### 2.2.1. Medición de la posición relativa

Los métodos que se emplean fundamentalmente en este tipo de medición son:

- **Odometría,** emplea codificadores para medir la rotación y orientación de la rueda, necesitan una referencia periódica para evitar la acumulación de error.
- **Navegación inercial,** que emplea giróscopos y acelerómetros de manera que además permite medir la velocidad, pero tiene el mismo inconveniente que en el caso anterior..

Al medir errores de odometría, uno debe distinguir entre errores sistemáticos, que son causados por imperfecciones cinemáticas del robot móvil (por ejemplo, diámetros de rueda desiguales), y errores no sistemáticos, que pueden ser causados por el deslizamiento de la rueda o irregularidades del piso. Los errores sistemáticos son una propiedad del robot en sí, y se mantienen casi constantes durante períodos prolongados de tiempo, mientras que los errores no sistemáticos son una función de las propiedades del piso. Borenstein y Feng [23], presenta un método para medir errores de odometría en robots móviles y para expresar estos
errores cuantitativamente.

### 2.2.2. Medición de la posición absoluta

Se basan en la percepción del entorno

- **Balizas activas**, que emiten luz o frecuencias de radio ubicadas en sitios estratégicos.
- **Reconocimiento de puntos de referencia**, que pueden ser artificiales, estos se colocan en ubicaciones conocidas y se pueden definir por características como la forma o el área; o puntos de referencia naturales, son puntos característicos del entorno, su precisión es menor que en el caso anterior.

- **Comparación de modelos**, en este método la información adquirida por los sensores del robot se compara con un mapa o modelo del entorno, de manera que se puede estimar la ubicación absoluta del vehículo.

Los robots autónomos deben poder aprender y mantener modelos de sus entornos. La investigación en la navegación de robots móviles ha producido dos paradigmas principales para mapear ambientes interiores: basado en rejillas y topológico. Mientras que los métodos basados en rejillas producen mapas métricos precisos, su complejidad a menudo prohíbe la eficiencia planificación y resolución de problemas en entornos interiores a gran escala. Los mapas topológicos, por otro lado, se pueden usar de manera mucho más eficientes, pero son considerablemente difíciles de aprender en entornos de gran escala. Bucken and Thrun [24], proponen un enfoque que integra ambos paradigmas, empleando redes neuronales artificiales, aprendizaje bayesiano y sensores sonar.

### 2.2.3. Sensores de medición de la posición absoluta

Los sensores empleados en este tipo de medición son [25]:

• **Transceptores infrarrojos (IR)**: un LED IR transmite un rayo de luz IR y si encuentra un obstáculo, la luz simplemente se refleja hacia atrás y es capturada por un receptor IR. Se pueden usar para la medición de la distancia.

• **Sensores ultrasónicos:** estos sensores generan ondas de sonido de alta frecuencia; la medida del tiempo que tarda el eco, nos indica a que distancia se encuentra un obstáculo. Los sensores ultrasónicos también se pueden usar para la medición de distancia.

• **Fotoresistores:** es un sensor de luz; se puede usar como un sensor de proximidad. Cuando un objeto se acerca mucho al sensor, cambia la cantidad de luz que a su vez cambia la resistencia del fotorresistor. Este cambio puede ser detectado y procesado.

- **Sensores visuales**: ayudan a los robots a identificar el entorno y tomar las medidas adecuadas. Los robots analizan la imagen del entorno inmediato importado del sensor visual. El resultado se compara con la imagen intermedia o final ideal, de modo que se pueda determinar el movimiento apropiado para alcanzar el objetivo intermedio o final.

## 2.3. Control y Procesado

### 2.3.1. El microcontrolador

A los circuitos integrados cuyos elementos son los de un procesador digital secuencial síncrono programable se les llama microcontroladores. Los microcontroladores están orientados a tareas de comunicación y control. Su costo reducido y consumo de energía y velocidad adaptables, resultan apropiados para numerosas aplicaciones. Un microcontrolador contiene un CPU (unidad central de procesamiento), una memoria ROM (memoria de sólo lectura), ó RAM (memoria de acceso aleatorio), entradas y salidas para comunicación externa y periféricos los cuales pueden ser PWMs (modulación de ancho de pulso), convertidores analógicos digitales, timers, etcétera.

### 2.3.2. Entornos de programación

Se llama Entorno de Desarrollo Integrado (IDE) a los programas informáticos formados por elementos y herramientas de programación. Un IDE puede entenderse como un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

#### MPLAB-IDE de MICROCHIP

La empresa Microchip, fabricante de microcontroladores, distribuye de manera gratuita desde su página web [12], un entorno de desarrollo integrado llamado MPLAB-IDE. El MPLAB-IDE es un programa que se ejecuta sobre un ordenador con el fin de desarrollar aplicaciones para los microcontroladores fabricados por Microchip [12]. Este programa software es específico de esta marca, por tanto no es posible emplearlo con un microcontrolador de otro fabricante.

#### SKETCH-IDE de ARDUINO

El lenguaje de programación y el entorno de desarrollo necesario para programar el microcontrolador de cualquier placa que pertenezca a la familia Arduino es Arduino. Gracias al entorno de código abierto se ha convertido en una plataforma de trabajo muy intuitiva, en la que escribir código y cargarlo a la placa es muy fácil.

Además funciona tanto en Windows, como en Mac O s X y Linux.El entorno está escrito en Java y basado en Processing, AVR-gcc y otros programas también de código abierto. El software de desarrollo es abierto y es posible descargarlo gratis. [15]

En cuanto al entorno de desarrollo, cabe decir que está formado por un área de mensajes, un editor de texto para crear el código, una consola de texto y una serie de menús. Adicionalmente habilita la conexión con hardware Arduino para cargar lso programas y también permite la comunicación con ellos. Arduino utiliza para escribir el código lo que denomina "sketch" (programa).

### 2.3.3. Arduino UNO (Atmega328P)

La placa electrónica Arduino UNO (como se muestra en la Figura 2.17) dispone del microcontrolador ATmega328. Además esta placa está dotada de catorce entradas y salidas digitales (6 de ellas pueden emplearse como salidas PWM), 6 entradas analógicas, conector USB, clavija hembra tipo Jack, y botón de reset. Algunas de sus características más relevantes se pueden apreciar en la tabla 2.1.

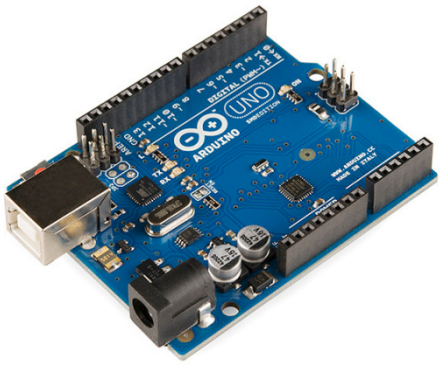


***Figura 2.17.*** *Placa de Arduino UNO. [27]*

| Microcontrolador | ATmega328 |
|---|---|
| Voltaje de funcionamiento | 5V |
| Alimentación | 7-12V |
| Límites voltaje de entrada | 6-20V |
| Pines digitales I/O | 14 (6 dan salida PWM) |
| Pines analógicos | 6 |
| Intensidad de corriente por Pin I/O | 40 mA |
| Intensidad de corriente para el Pin 3,3V | |
| Memoria Flash | 32 KB (ATmega328) |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Velocidad del reloj | 16 MHz |

***Tabla 2.1.*** *Características del microcontrolador Arduino UNO.*

### 2.3.4. Texas Instruments MSP430 LaunchPad

Esta plataforma pertenece a Texas Instruments, la cual en términos generales es el competidor directo de Arduino UNO. Es solo una de la también buena variedad de Texas Instruments. A continuación se añade una lista con sus propiedades más relevantes.

Microcontrolador: MSP430G2553
- Frecuencia: 16Mhz
- Flash: 16KB
- RAM: 512B
- 8 canales 10-bit ADC
- Comparador
- 2 contadores de 16-bit

Protocolos comunicación: I2C, UART

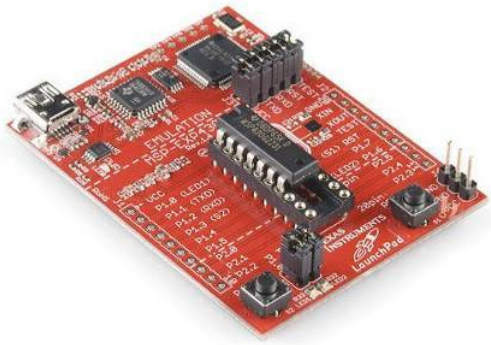

***Figura 2.18.*** *Placa MSP430 de Texas Instruments. [28]*

## 2.4. Comunicaciones

Actualmente es posible la transferencia de información, entre dos o más terminales, sin la necesidad de una conexión por cable, este tipo de conexión se denomina inalámbrica. Su ventaja fundamental es la versatilidad, ya que la instalación de estas redes no requiere de ningún cambio significativo en la infraestructura existente, en el sentido de no tener que perforar paredes, ni de instalar portacables o conectores, como pasa con las redes cableadas. Esta ha sido la causa de la enorme rapidez con la que se han extendido.

Con las redes inalámbricas el dispositivo puede mantenerse conectado cuando se desplaza dentro de una determinada área geográfica, permite a los usuarios tener acceso a otras computadoras, bases de datos e internet con total movilidad sin perder la conexión.

Los medios de transmisión más usuales en las redes inalámbricas son:

- Radiofrecuencia, cuyo espectro de frecuencias va desde los 3 KHz a los 3 GHz. Es la tecnología inalámbrica más común, la banda de frecuencias más utilizada es la de 2,4 GHz, el problema fundamental es la distancia que pueden alcanzar. Los vehículos móviles son muy sensibles al consumo de energía que se produce en la transmisión de señales. Otros problemas son el ruido por interferencias electromagnéticas y los obstáculos físicos que suelen atenuar el alcance de la señal, a pesar de todo es posible conseguir comunicaciones fiables.
- Microondas, desde 1 a 300 GHz, se necesitan antenas parabólicas y que el emisor y receptor estén alineados.
- Infrarrojos, desde los 300 GHz a 348 THz, necesitan alineación entre receptor y emisor o emplear algún tipo de reflexión.
- Ultrasonidos, se utilizan en aquellos medios donde no es posible emplear radiofrecuencia, sus características son similares a esta.
- Laser, se emplea fundamentalmente en comunicaciones espaciales.

Las redes inalámbricas permiten conexiones remotas, sin dificultad desde unos metros de distancia hasta varios kilómetros. Existen muchas tecnologías diferentes que se diferencian por la frecuencia, alcance y velocidad de sus transmisiones. Se clasifican en varias categorías, de acuerdo al área geográfica que abarca la conexión posible del usuario a la red, o área de cobertura. Cada grupo posee sus propios estándares o conjunto de reglas que deben cumplir los protocolos de comunicaciones, para permitir la comunicación entre dispositivos de distintos fabricantes.

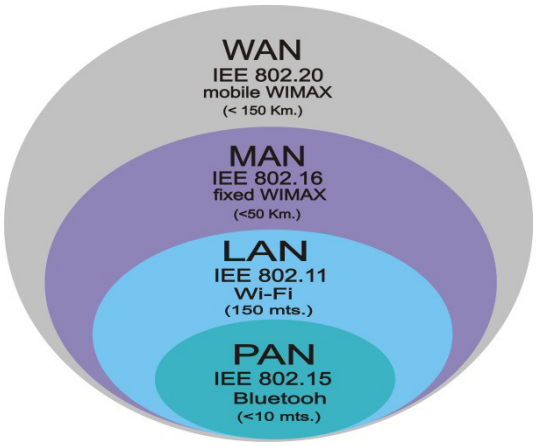


***Figura 2.19.*** Categorías de las redes inalámbricas. [26]

- **WWAN (Wireless Wide Area Networks).** Interconecta redes LAN, abarca áreas geográficas ilimitadas: una región, un país, el mundo. Son de carácter público. El ancho de banda debe ser alquilado o rentado. El estándar de esta rede es el IEEE 802.20. El dispositivo de red que determina este tipo de red es el ROUTER.
- **WMAN (Wireless Metropolitan Area Networks ).** Red de área metropolitana. La tecnología más representativa es WiMax, dentro de las conocidas como tecnologías de última milla, permite la recepción de datos por microondas y retransmisión por ondas de radio, basada en el standard IEEE 802.16MAN. Utiliza las bandas 2,3 GHz, 2,5 GHz y 5,4 GHz.
- **WLAN (Wireless Local Area Networks,).** Red de área local. Son de carácter privado. Utilizan el ancho de banda "gratis. Emplea las bandas de 2,4 y 5,4 GHz. Su alcance operativo es hasta los 500 m. El sistema más popular es WI-FI basado en el estandard IEEE 802.11. Los dispositivos habilitados con WI-FI pueden conectarse a internet a través de un punto de acceso de red inalámbrica.
- **WPAN (Wireless Personal Area Networks).** Red de área personal. Basada en el estándar IEEE 802.15. Se emplean en automatización de recintos, aplicaciones publicitarias o dispositivos conectados a un punto de acceso de un corto alcance [19][20].

Las redes WPAN son las que tienen más interés desde el punto de vista de este trabajo y se analizarán más en detalle, en concreto los sistemas Bluetooth y Zigbee [12]. Existen otros que emplean distintos medios físicos de transmisión; como HomeRF que se utiliza para telefonía inalámbrica, RFID que utiliza la radiofrecuencia para la identificación de objetos, NFC emplea para la transmisión un campo electromagnético y se usa para transacciones comerciales con tarjetas, IrDA cuyo medio de transmisión son los infrarrojos e Insteon que se utilizan sobre todo en domótica y aprovecha la instalación eléctrica.

### 2.4.1. Bluetooth

Se trata de un standard de hecho, para la conexión inalámbrica de corto alcance entre numerosos dispositivos [19][20]. Soporta varias conexiones simultáneas, tanto punto a punto como multipunto, en la última especificación hasta siete dispositivos esclavos pueden comunicarse con uno maestro, a estas redes se las denomina piconet y se permite dejar a los esclavos en "stand by", a la escucha del maestro, para los mensajes de sincronización y transmisión, lo que supone un ahorro de energía [22].

Emplea trasmisión por radio de alta frecuencia, 2,45 GHz ISM (Industrial, Scientific and Medical) que permite la transmisión de voz, datos e imágenes, empleando microchip, integrados en el dispositivo que se desee (teléfonos móviles, ordenadores personales, etc.). Disponible en todo el mundo sin necesidad de licencia, con un ancho de banda de 1MHz, soporta hasta 79 canales, utilizando modulación binaria por desplazamiento en frecuencia FSK (Frecuency Shift Keying). empleando microchip, que los trasmisores y receptores, integrados en el dispositivo que se desee (teléfonos móviles, ordenadores personales, etc.).

Soporta tanto servicios síncronos como asíncronos, por lo que es fácil su integración con TCP/IP, por lo que ha adoptado este protocolo y el de OBEX (intercambio de objetos).

Se trata de un estándar que soporta entornos con muchas interferencias electromagnéticas utilizando tecnologías de salto de frecuencias como CDMA-FH (Code Division Multiple Acces-Frecuency-Hop).

Consume una cantidad baja de energía por lo que se considera adecuado en aplicaciones tales como sensores, videoconsolas, juguetes, automoción, carnets inteligentes, controles remotos, domótica y dispositivos móviles que funcionen con pilas o baterías [19]. Dependiendo el alcance de la potencia de transmisión, así se puede hablar de:

Clase 1: si el alcance es de 100 m y potencia 100mW. Necesita de un amplificador opcional

Clase 2: alcance es de 25 m y potencia 2,5 mW.

Clase 3: alcance entre 1 y 10m y potencia 1 mW. Cubre la distancia de una habitación y es el de uso más común.

Clase 4: alcance 0,5 m y potencia 0,5 mW.

En cuanto a la velocidad de transmisión, su mejora ha sido notoria en cada versión:

Bluetooth v1.1, es la primera y aparece en 1994 desarrollado Ericsson, incluía en el hardware, de forma obligatoria, la dirección del dispositivo de transmisión, lo que impedía el anonimato y por lo tanto su uso en algunas aplicaciones [18][21].

Bluetooth v1.2, permite la coexistencia con WI-FI en el espectro de 2,4GHz, mejora la eficiencia y la seguridad en la transmisión mediante la técnica AFH (Adaptative Frecuency Hopping).

Bluetooth v2.0 y 2.1, Se trata de una nueva especificación, incorpora la técnica EDR (Enhanced Data Rate), que permite alcanzar velocidades de transmisión de hasta 3Mbps. La versión v2.1 reduce el consumo a la quinta parte.

Bluetooht v3.0: puede trabajar con WiFi con lo que se consiguen velocidades de transferencia de 24 Mbps.

Bluetooht v4.0: la velocidad aumenta hasta los 32Mbps. Incorpora en su especificación el protocolo Wibree, sistema de comunicación inalámbrica desarrollado para conseguir un ultra bajo consumo en los dispositivos que lo incorporan, esta implementación se denomina BLE (Bluetooth Low Energy) y su velocidad de transferencia es de 1 Mbps.

Bluetooht v5.0: mejora la eficiencia en la transmisión de datos, limitando el consumo de los dispositivos. Está orientado al desarrollo de aplicaciones basadas en el IoT (Internet of Things).

Tiene soporte para seguridad mediante encriptación y autentificación. Recientemente se ha detectado una amenaza bajo el nombre de BlueBorne en las conexiones a internet, a la fecha de realización de este trabajo ya se está trabajando para eliminarla.

El costo de su utilización se limita al del producto en el que se integra, no existe un registro del servicio relacionado con el uso de esta tecnología, funciona con un espectro de radio sin licencia, lo que significa que no existe un coste adicional para la comunicación [18].

Los dispositivos se pueden comunicar mediante esta tecnología sin necesidad de estar alineados, funciona aunque existan obstáculos intermedios.

### 2.4.2. ZigBee

Es el nombre de un conjunto de protocolos basados en el estándar IEEE 802.15.4, cuyo objetivo son las aplicaciones seguras con una tasa baja de transmisión de datos para conseguir un mínimo consumo de energía y la máxima duración de las baterías. Se engloba dentro de las llamadas redes inalámbricas de área personal.

El desarrollo de esta tecnología se centra en su sencillez, ya que necesita un 10% menos de hardware que un nodo típico de Bluetooth clásico o WI-FI, aunque a nivel de software necesite un 50% más de código. Otro factor importante es su bajo coste.

Desde un punto de vista de consumo de potencia Bluetooth ULP o BLE y ZigBee son similares, no así en la velocidad de transmisión que en ZigBee es de unos 163 kbps mientras que en Bluetooth ULP ronda los 35 kbps.

Zigbee se emplea fundamentalmente en aplicaciones domóticas por su flexibilidad en las topologías de red: en malla o estrella.

### 2.4.3. WI-FI

Es una marca de la WI-FI Alliances, organización global sin ánimo de lucro, integrada por cientos de empresas, que promueve esta tecnología, certificando que los productos que la llevan se ajustan a sus normas de inteoperabilidad y son compatibles con IEEE 802.11. La Alianza WI-FI ha impulsado la adopción generalizada de esta tecnología en todo el mundo.

Pertenece al ámbito de las redes WLAND o redes de área local, por lo que pueden alcanzar los 200m con buena cobertura. La energía consumida depende del uso particular que se haga de ella, numerosos dispositivos utilizan WMM Power Save función que sirve para ahorrar energía en los dispositivos que funcionan con baterías.

Sus aplicaciones más habituales son en conexiones a internet, transferencia de datos, seguridad, controles remotos. Es como una Ethernet sin cables, por lo que necesita una configuración previa. Utiliza el mismo espectro de frecuencias que Bluetooth con una potencia mayor y por lo tanto mayor consumo de energía, se adecua a redes de propósito general y cubre necesidades distintas en el entorno doméstico

### 2.4.4. Coexistencia de Bluetooth y WI-FI

Los dispositivos Wifi y Bluetooth operan en la misma banda de frecuencias de 2.4Ghz. Bluetooth utiliza muy baja potencia para funcionar por lo que un equipo Wifi (Estándar 802.11) puede provocar una gran interferencia en el enlace Bluetooth, en lo que se conoce como "interferencia persistente". Existen técnicas que permiten la coexistencia colaborativa entre redes WPAN y WLAN [22], estas son: la selección de paquetes adaptativos y la mencionada anteriormente saltos de frecuencia adaptativos.

También es posible el acceso a internet por parte de dispositivos Bluetooth llamados Gateway Inalámbricos Bluetooth (BWG), que permite el acceso a transmisores WI-FI, en lo que se conoce como arquitectura Bluestar que garantiza que no haya interferencias entre enlaces WI-FI y Bluetooth. También pueden acceder a internet a través de redes WWAN como LTE, WiMAX, UMTS etc.

# 3. Diseño e Implementación Mecánica

En este capítulo se aborda la descripción del sistema mecánico, detallando así los aspectos más técnicos del vehículo e introduciendo algunos de los conceptos clave que han permitido modelizar en el papel el sistema y por tanto han facilitado la comprensión del mismo. Inicialmente se hace una introducción de los distintos componentes que lo conforman para acto seguido dividirlos en dos grupos: los pertenecientes al sistema motriz, y los que pertenecen al sistema de control o direccional. Para finalizar se esclarece el término de automóvil robotizado o vehículo Ackerman, concluyendo la sección del control direccional mecánico del prototipo.

## 3.1. El Prototipo de Automóvil

### 3.1.1. Introducción

En el vehículo automóvil controlado por radiocontrol empleado en este proyecto se pueden diferenciar dos grandes conjuntos mecánicos: el chasis, y la carrocería anclada al mismo. Adicionalmente lleva las partes del equipo de radio necesarias para su control (antena, receptor, baterías, e interruptor), componentes que se han sustituido por el control mediante una aplicación de telefonía móvil por lo que se han omitido en el prototipo objeto de este proyecto.

También lleva un motor, eléctrico, con la correspondiente fuente de energía que habilita el movimiento (batería). Además el vehículo consta de suspensión independiente a las cuatro ruedas con su correspondiente transmisión. Finalmente, cabe mencionar que el automóvil empleado dispone de una opción "turbo propulsión", de la cual no se ha hecho uso.

El chasis está formado por los siguientes elementos:

- Estructura Resistente
- Motor y elementos de la transmisión
- Ejes delanteros, traseros y ruedas

- Suspensión, une las ruedas o ejes al bastidor
- El sistema de dirección

Actualmente, la carrocería se encuentra acoplada a la estructura resistente en lo que se conoce como carrocería auto portante. Es la configuración empleada en vehículos turismos, en los industriales ligeros y algunos todoterrenos. A continuación, podemos observar las diferentes partes que conforman el prototipo (Figura 4.1).

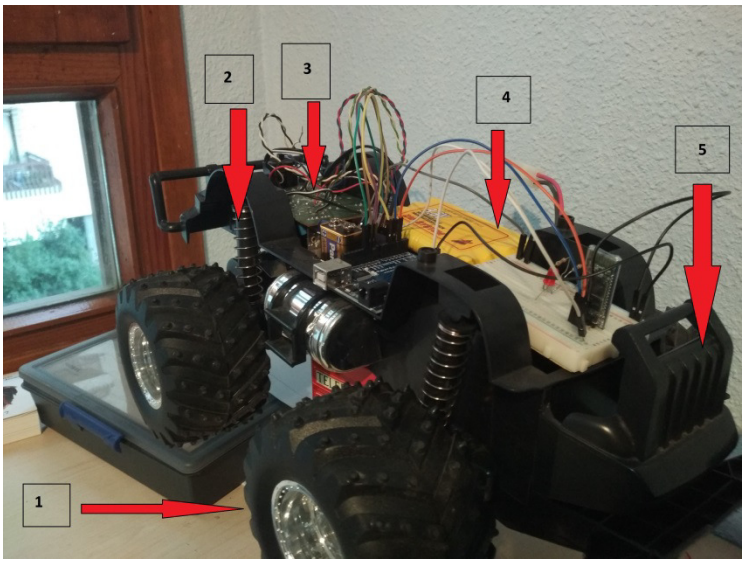


***Figura 3.1.*** *Esquema principal del vehículo de radiocontrol.*

1. Ruedas (Neumático y Llanta)

2. Suspensión (Amortiguador / Muelle)

3. Puente H

4. Batería

5. Estructura Resistente

En un coche eléctrico:

- El motor es eléctrico (sin carburador, embrague ni filtros)
- Las baterías de tracción sustituyen a depósito, filtros y escape.
- Regulación de velocidad por regulador electrónico (sin servo de acelerador/freno)
- El freno es eléctrico (no mecánico)

## 3.1.2. Características Técnicas. Dimensiones

DIMENSIONES DEL AUTOMÓVIL (SEGÚN UNE 26-192-87)

*"Longitud del vehículo:* Distancia entre los planos verticales perpendiculares al plano medio del vehículo, que tocan al vehículo por delante y por detrás."

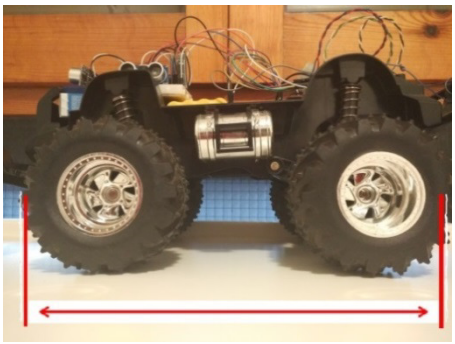

***Figura 3.2.*** *La longitud del vehículo son 30 centímetros.*

*"Anchura del vehículo:* Distancia comprendida entre dos planos paralelos al plano longitudinal medio del vehículo que tocan al vehículo en los dos costados."

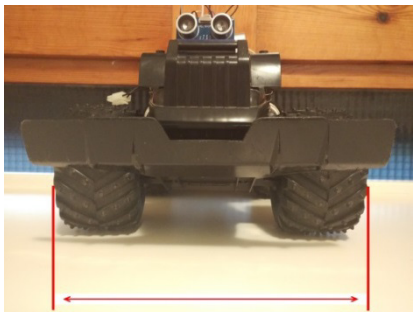

***Figura 3.3.*** *La anchura del vehículo son 15 centímetros.*

*"Altura del vehículo:* Distancia entre el plano de apoyo y un plano horizontal que toca la parte superior del vehículo."

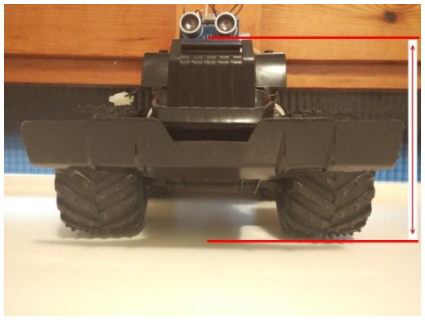

***Figura 3.4.*** *La altura del vehículo son 12 centímetros.*

## 3.2. El Sistema Motriz

- **MOTOR ELÉCTRICO CONTINUA**

Con respecto al vehículo, posee un motor conectado a las ruedas de dirección y otro a las motrices. La mayoría de los vehículos robotizados son capaces de generar movimiento siendo los más usuales los motores de continua (DC) y los servos motores. Los primeros son utilizados junto con un tren de engranajes que reducen la velocidad y le dotan de mayor fuerza; el motor está compuesto de un rotor y un estator. El motor funciona con voltajes que van de 7.2 V a 16 V.

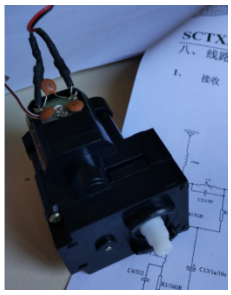

***Figura 3.5.*** *Motor DC.*

- **RUEDAS MOTRICES**

En lo que se refiere al tren inferior, las ruedas tiene una llanta forrada de esponja acrílica de forma que disminuya la fricción conforme al suelo, así como una banda mejorada de rodadura. Por lo que concierne al proyecto, el prototipo dispone de dos ruedas de dirección, 2WD. En cuanto a las medidas, son: radio 32 milímetros y ancho 22 milímetros.

***Figura 3.6.*** *Rueda del vehículo.*

# 3.3. La Dirección y el Control Direccional

## 3.3.1. Introducción

Todos los elementos que integran la orientación de las ruedas, en función de los movimientos realizados por el piloto sobre un mando de dirección, es conocido como dirección de un vehículo. El objeto de la dirección es situar al vehículo en la trayectoria de las curvas, la evasión o adelantamiento de obstáculos presentes en su camino y las maniobras de aparcamiento. El proyecto se centra especialmente en el control direccional, en sus aspectos: mecánico, electrónico e informático.

## 3.3.2. Tipos de Sistemas de Dirección

Para la unión entre la dirección y el varillaje se emplean varias configuraciones, que se agrupan en dos grandes grupos: los sistemas de bolas recirculantes (Figura 3.7) y los sistemas de piñón cremallera. En el caso que concierne al prototipo objeto del proyecto, este dispone de un sistema de bolas recirculantes y es servo dirigido; es decir, las fuerzas que actúan sobre la dirección proceden de una fuente externa de energía que es la batería.

Las ventajas de los sistemas de bolas frente a los de piñón cremallera son las siguientes:

- Pueden usarse en eje rígido
- Transmiten fuerzas elevadas
- Se consiguen ángulos de giro importantes en la rueda

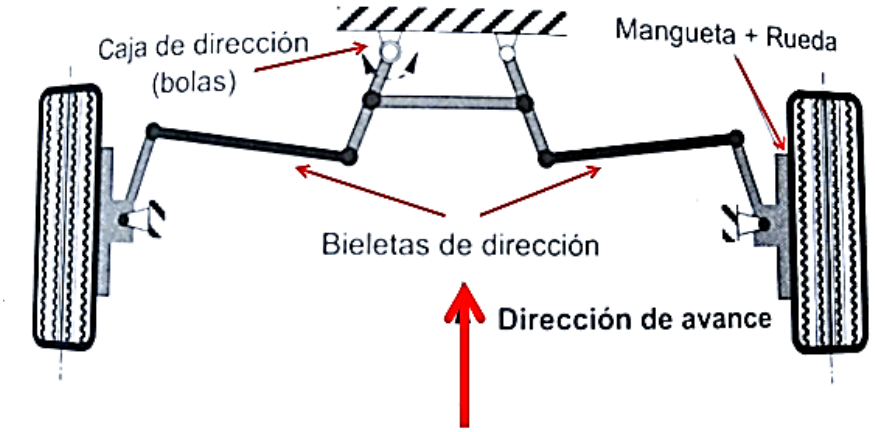


***Figura 3.7.*** *Cuadrilátero articulado [31].*

### 3.3.3. El Trapecio de Dirección

El trapecio de dirección está formado por el eje delantero, junto con el brazo de dirección (marcado con un 1) y la barra estabilizadora (marcado con un 2), tal como podemos observar en la Figura 3.8. En la imagen se describen dos situaciones diferentes, en ellas se muestra los diferentes ángulos que forma el trapecio de dirección en las curvas. La barra estabilizadora es paralela al eje delantero en línea recta. Cuando se gira, el eje de pivoteo (marcado con un 3) debe moverse.

Abajo se muestra el trapecio de dirección del vehículo objeto del proyecto, Figura 3.9.

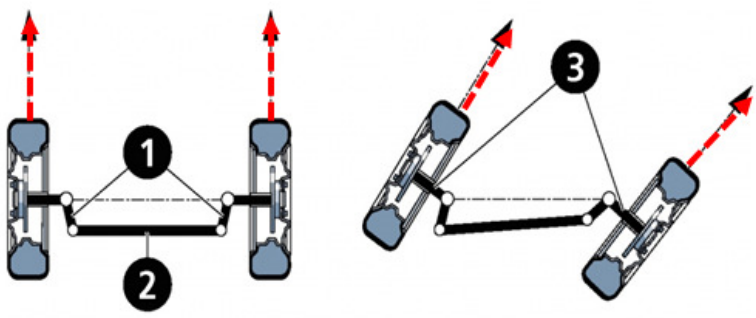


***Figura 3.8.*** *Trapecio de dirección del vehículo [29].*

***Figura 3.9.*** *Sistema de bolas re circulantes del vehículo.*

## 3.3.4. Fundamentos Geométricos – Condición de Ackerman

*"El principio de Ackerman enuncia que cuando un vehículo gira en una curva, los ejes de todas las ruedas deben concurrir en un punto, el centro instantáneo de rotación."* [3.3]

Es la configuración que estamos habituados a ver en los vehículos convencionales, y por lo tanto es una configuración muy probada y estable. Se basa en una estructura de cuatro ruedas colocadas en dos ejes, donde sólo las dos ruedas delanteras permiten un giro sobre el eje vertical.

Esta condición permite obtener la geometría adecuada para la dirección del vehículo; el ángulo de giro de la rueda más alejada es menor que el de la rueda interior, tal y como se muestra en la Figura 4.14.

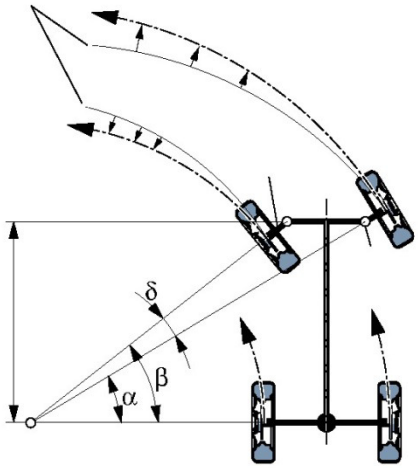


***Figura 3.10.*** *Esquema de los ángulos que intervienen en la dirección [29].*

### 3.3.4.1. Ángulo de Guiado (δ) y Control Direccional del Vehículo

Para que el diseño del sistema de dirección asegure una cierta estabilidad y un plausible control direccional, es indispensable que el eje pivotado por el conjunto mangueta – rueda, con relación al vehículo, adquiera una situación espacial conveniente. Los ángulos característicos son: guiado (δ), salida (σ), caída (ε) y avance (τ). **No obstante se ha considerado únicamente relevante el ángulo de guiado (δ) para el proyecto.**

La rueda directriz forma un ángulo con el plano longitudinal del vehículo, este ángulo recibe el nombre de ángulo de guiado (Figura 3.11). Como se ha mencionado anteriormente, el ángulo de guiado es función del radio de la curva descrita, así como

de la geometría del coche, en particular del ancho del carril y la distancia entre ejes. En condiciones normales se tiene que hacer una distinción entre los ángulos de guiado formados por la rueda exterior e interior. Sin embargo, para este trabajo, **se ha considerado una dirección paralela, por lo que se cumplirá que ambos ángulos son iguales. Para ello, en el diseño se incorpora una geometría basada en la condición de Ackerman, aunque algo modificado.**

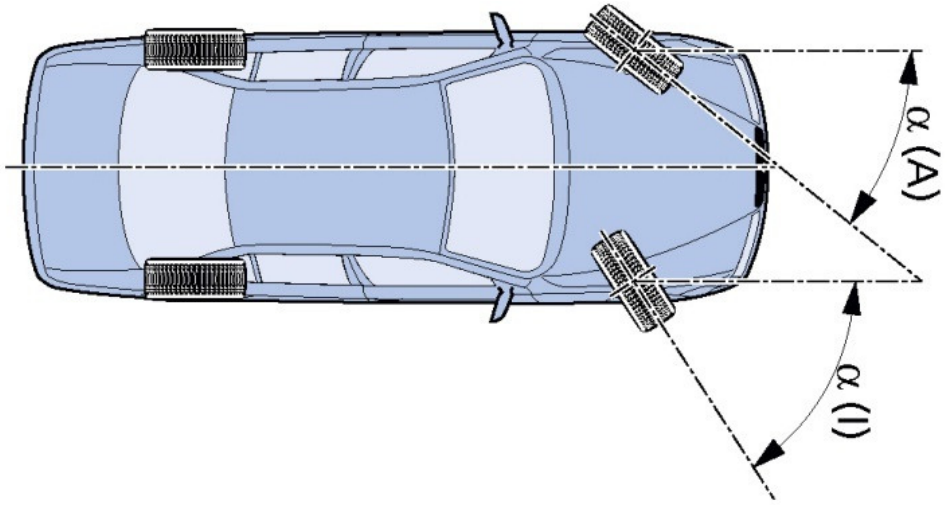


***Figura 3.11.*** *Ángulos de guiado exterior e interior [29].*

### 3.3.1.2. Vehículo Ackerman

El vehículo robotizado o vehículo de Ackerman es el más sencillo de entre todos los tipos de robots móviles con ruedas. Lo conforman dos ruedas traseras fijas, y dos delanteras estándar dirigibles. Para ser más específicos, una bicicleta sigue el mismo modelo pero con sólo una rueda de cada tipo. Ambos tipos de ruedas restringen la cinemática del desplazamiento lateral, esto es, el coche no se mueve en la dirección perpendicular al plano de las ruedas. Si se traza la intersección de las líneas punteadas, véase Figura 3.12, se obtiene un punto determinado que se llama Centro Instantáneo de Rotación (ICR en inglés o CIR); dependiendo el ángulo de guiado (δ) que se introduzca, se obtendrá un CIR en cada instante, si mantenemos constante el ángulo, el vehículo traza una trayectoria circular de radio R, cuyo centro es el CIR. Según este principio, el robot se mueve en línea recta cuando se supone un R infinito y por tanto un δ nulo.

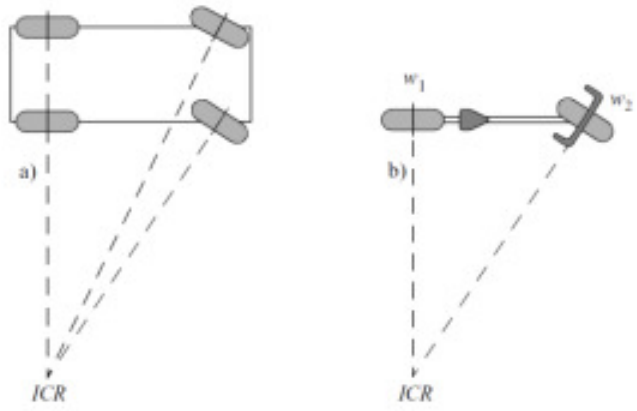


***Figura 3.12.*** *Centros Instantáneos de Rotación de un coche y una bicicleta [30].*

La disposición geométrica del CIR depende del número de impedimentos del movimiento del vehículo y no del número de ruedas. Así por ejemplo, se obtiene idéntica solución si se considera una bicicleta con una rueda trasera y una delantera. Por cada rueda se contribuye a una restricción o línea de "movimiento nulo", las cuales intersectan en un punto (CIR). Ambas condiciones son independientes lo que significa que constituyen las restricciones del movimiento del robot.

No obstante, esto no sucede con el vehículo Ackerman, ya que a pesar de tener más de una rueda no añade más restricciones independientes. Respecto a las ruedas traseras, estas solo introducen una restricción al estar alineadas y fijas; así pues, con la restricción una de las ruedas delanteras, se obtiene una solución del CIR. La otra rueda delantera en cambio, se encontrará de forma que su línea de "movimiento nulo" se encuentre con el CIR, es decir, no introduce ningún impedimento al movimiento. En definitiva, la consecuencia es que las ruedas de delante tengan una ligera variación en el ángulo de dirección entre ellas mismas, impuesto gracias a la condición de Ackerman.

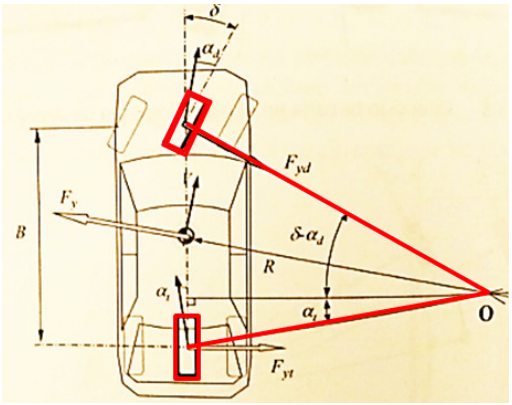


***Figura 3.13.*** *Modelo del comportamiento direccional [31].*

# 4. Instrumentación y Sistema de Control. Implementación Hardware.

El objetivo principal de este Capítulo es uno de los temas más trascendentes en el mundo de la Robótica: el control. El control hace posible que el robot obedezca las órdenes que le precisamos para que ejecute de forma óptima las tareas, ya que si lo hiciera fuera de control, la tarea se haría de forma incorrecta. Póngase por caso los brazos manipuladores industriales, si no se controla de manera precisa su movimiento, ciertamente no obtendremos el resultado buscado y podría ser peligroso.

En el mundo de la robótica móvil es fundamental la información que procede de los sensores para que el control del prototipo ejecute de forma autónoma un plan o realice tareas reaccionando a sucesos en su entorno.

- Generador Global de Trayectorias (GGT): Nivel superior de decisión de las coordenadas del punto de destino e intermedios de la trayectoria.
- Generador Local de Trayectorias (GLT): Nivel intermedio, aquí se simula el robot para evitar obstáculos, modificando la trayectoria y corrigiendo la velocidad a los movimientos. Básicamente se encarga del control dinámico del robot móvil e informa al GGT de los resultados.
- Control Local de Sistema de Tracción y Dirección (CL): Nivel inferior que se encarga de la lectura de las referencias dadas por el GLT y produce acciones de control sobre el motor de dirección y de tracción.

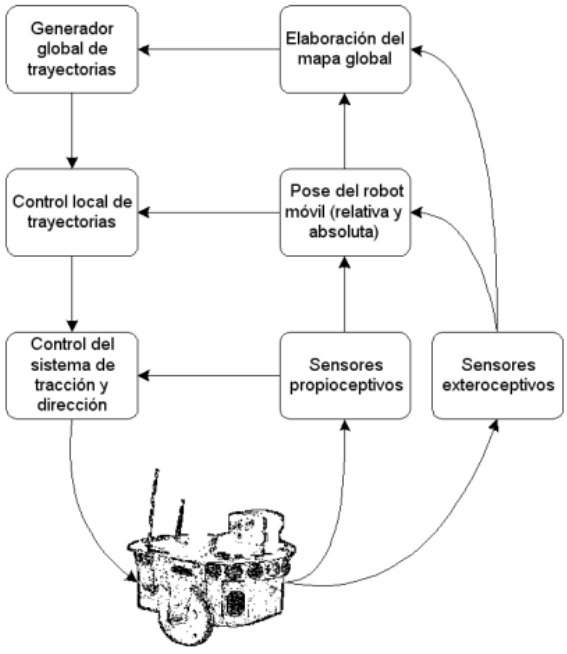


***Figura 4.1.*** *Esquematización del sistema de control de un robot móvil [36].*

Cómo se puede ver en la Figura 4.1, un esquema de control básico para un robot móvil autónomo presenta dos bucles, englobando ambas partes, GLT y CL. El bucle de la izquierda controla la dirección, es decir, ordena a los actuadores para que sigan la dirección correcta en cualquier momento. Por otro lado, el bucle de la derecha es el control de la velocidad, generando órdenes a los actuadores responsables de la propulsión del vehículo. Por lo que respecta a este proyecto, en el caso de vehículos convencionales tipo Ackerman, los bucles están fuertemente ligados.

Dentro de las estrategias de control, nos encontramos con dos en particular: por una parte la estrategia deliberativa, es aquella que razona la percepción del robot (datos de la instrumentación) y se encarga de generar un modelo del entorno (memoria) organizando las acciones objetivo, es similar a la inteligencia artificial; por otra, el control reactivo, que se presenta como la otra cara de la moneda, es aquél que elimina el conocimiento, es decir, se suprime la planificación y el modelado del entorno. Los datos de los sensores (estímulos) generan acciones "reflejas", lo que deriva a respuestas rápidas y de bajo coste computacional, a diferencia del control deliberativo. En definitiva, las estrategias reactivas instruyen al robot a reaccionar ante sucesos en el entorno.

## 4.1. Implementación Hardware

En lo que concierne a este apartado, se procede a la descripción detallada del montaje, así como los materiales y conexiones, que han sido necesarios para desarrollar físicamente el proyecto, esto es, se explica cada elemento que compone el sistema hardware del robot móvil, desde los sensores empleados, pasando por el microcontrolador, el puente H y el circuito que lo comprende, y finalizando con el módulo Bluetooth.

### 4.1.1. Descripción General

A continuación, en la Figura 4.2, se muestra el diagrama de bloques llevado a cabo en este proyecto. En el diagrama se aprecia como todas los componentes del sistema de engloban dentro de una misma unidad alimentada la tensión de 9.6 V aproximadamente.

En particular, el módulo Bluetooth, el microcontrolador y la aplicación móvil son las partes más destacadas, ya que representan la comunicación, el control y el guiado; no obstante se han de destacar los sensores de ultrasonidos y el circuito del puente H, implementado ya en el sistema mecánico.

De acuerdo con la elección de cada componente se han realizado una serie de ensayos para asegurar un correcto funcionamiento de cada uno. En concreto, ha sido así para cada parte con la excepción del puente H, que se encontraba integrado en el coche teledirigido, y se decidió aprovecharlo.

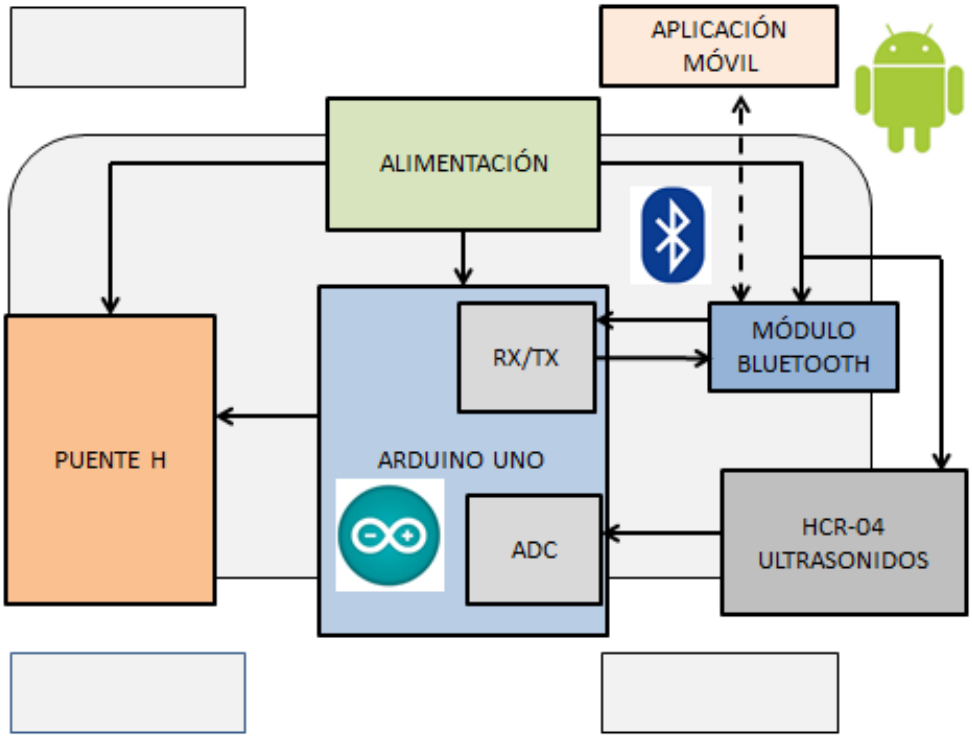


***Figura 4.2.*** *Diagrama de bloques del sistema.*

### 4.1.2. Elección del Microcontrolador

Tras una ardua búsqueda de entre una amplia gama de modelos disponibles de microcontroladores de la plataforma Arduino, se acordó la realización de una tabla comparativa (Tabla 4.1) de todos ellos, para la posterior elección de la versión que más se adecuaba al prototipo objeto de este proyecto.

Se examinan brevemente los parámetros de mayor trascendencia a la hora de conseguir eficiencia en el control del vehículo.

| Modelo Arduino | NANO | LEONARDO | UNO | MEGA 2560 |
|---|---|---|---|---|
| Entradas Analógicas | 8 | 12 | 6 | 16 |
| E/S Digitales | 14 | 20 | 14 | 54 |
| Microcontrolador | ATmega168(SMD) | ATmega32u4(SMD) | ATmega328 | ATmega2560(SMD) |
| Precio (IVA no incluido) | 33 € | 18 € | 20 € | 39 € |

***Tabla 4.1.*** *Características de los modelos candidatos.*

De acuerdo al número de entradas y salidas digitales disponibles en las diferentes alternativas, todas disponen del número mínimo necesario para el vehículo, por tanto, no se excluye ninguna. Por otra parte, respecto a los pines analógicos, no supone ningún motivo de exclusión.

En cuanto a las pruebas y ensayos a las que se someterá el candidato, las plataformas Arduino presentan una clara desventaja frente a los microcontroladores de Texas Instruments, en el caso de avería y sustitución del microcontrolador, no solamente sería necesario sustituir el montaje superficial, sino que también habría que cambiar la plataforma Arduino. La versión UNO es la única con microcontrolador intercambiable; esta es la razón principal por la que hemos elegido este modelo; otra razón de peso es que la tarjeta UNO integra el microcontrolador ATmega16U2, programado como puente de USB a serie con los drivers necesarios ya instalados.

### 4.1.3. Características

Para el desarrollo del proyecto se ha empleado el microcontrolador Arduino UNO, que difiere de todos los demás modelos en que no usa el chip driver FTDI USB a serie, sino que, funciona con un ATmega16U2 programado como convertidor USB a serie, como se ha mencionado anteriormente. Sus principales características son:

- Microcontrolador basado en el ATmega328.
- Tiene 14 pines con entradas/salidas digitales (6 de las cuales pueden ser usadas como salidas PWM), 6 entradas analógicas.
- Un cristal oscilador a 16Mhz.
- Conexión USB.
- Entrada de alimentación.
- Un botón de reset.

Además, para el prototipado e integración de los sensores se ha empleado una protoboard.

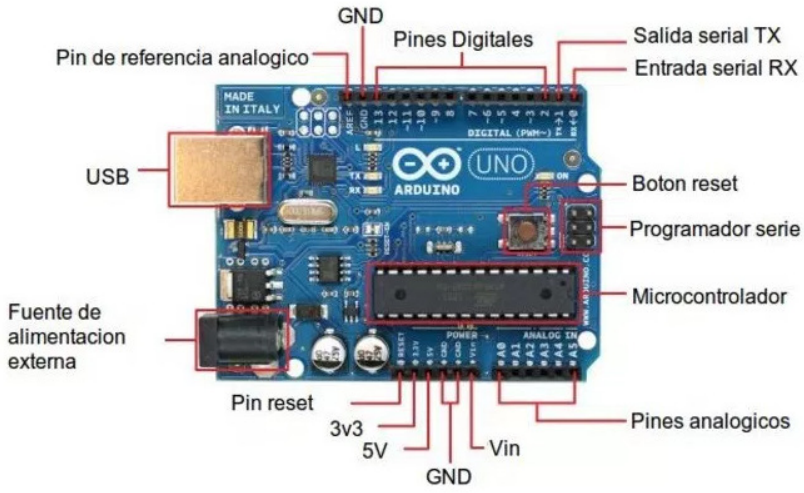


***Figura 4.3.*** *Placa de UNO [32].*

### 4.1.4. Sensor de Ultrasonidos HC-SR04

Cuando se presenta un sistema autónomo parece inevitable focalizar nuestra atención en su interacción con el entorno, en ocasiones similar a la de un humano, y es que una habilidad indispensable en un robot es la de obtener información del entorno en el que se encuentra, gracias a los sensores incorporados. Generalmente los robots móviles se dividen en dos tipos:

***Figura 4.4.*** *Sensor de ultrasonido HC-SR04 [33].*

- Propioceptivos, o aquellos que proporcionan información del estado interno del robot, tales como el voltaje de la batería o la velocidad de los motores.
- Exteroceptivos, obtienen la información del entorno exterior, ya sea sonido, intensidad de luz o medidas de distancia; el prototipo pertenece a este último subgrupo (exteroceptivos) y mediante sensores de ultrasonidos toma medidas de distancia.

El HC-SR04 es un sensor piezoeléctrico de ultrasonidos de gama media-baja, pero que cumple satisfactoriamente las prestaciones que se requieren en este proyecto y su relación calidad-precio es bueno, se pueden encontrar una gran variedad de precios, desde cincuenta céntimos de euro hasta los cuatro euros.

El uso de este sensor permite saber la distancia, más o menos exacta, a la que se encuentran los obstáculos y por tanto habilita el control del vehículo. Consta de cuatro pines, VCC, GND, TRIG (disparo de ultrasonido), y ECHO (recepción del ultrasonido). La secuencia de trabajo es la siguiente, el sensor envía un tren de pulsos de alta frecuencia no perceptible por el oído humano, el pulso rebota en el obstáculo más próximo y es recibido por la parte receptora del sensor. Para conocer la distancia, se mide el tiempo entre los pulsos (función *pulseIn (Pin_ECHO, HIGH)* en la plataforma Arduino) y conociendo la velocidad del sonido se calcula la distancia a la que se encuentra el objeto. El cálculo para medir la distancia se presenta a continuación:

$$343\text{ms} \cdot 100cmm \cdot 11 \cdot 106\mu s \ = 0.0343\ cm\mu s \ \rightarrow 1\ cm\ son\ 29.1545\ \mu s$$

$$Distancia\ [cm] = Tiempo[\mu s]\ 29.1545 \div 2$$

El tiempo se divide por dos porque lo que se mide es el tiempo que tarda el pulso en ir y volver, por tanto, sino se realizase esta división se estaría calculando el doble de la distancia a la que está el objeto. Este cálculo se conoce como técnica impulso-eco (Figura 4.5).

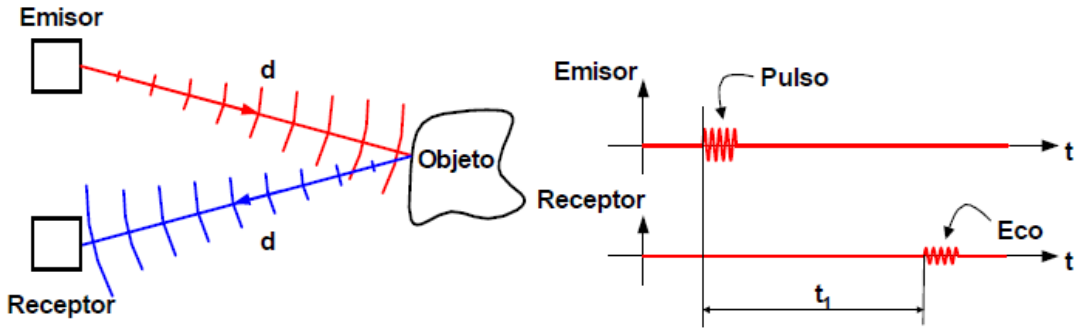


***Figura 4.5.*** *Técnica Impulso-eco. [37].*

**ESPECIFICACIONES**

- Voltaje de alimentación: 5V DC
- Corriente: $\leq$ 2 mA
- Ángulo de cobertura: $<$ 15°
- Rango de medición: 2cm - 500cm
- Resolución: 0.3 cm

**MEDICIÓN**

- Poner el pin TRIG a nivel alto durante 10 $\mu s$
- Esperar un pulso de retorno en el pin ECHO
- Vel.sonido = 340 m/s = 1/29 cm/$\mu s$

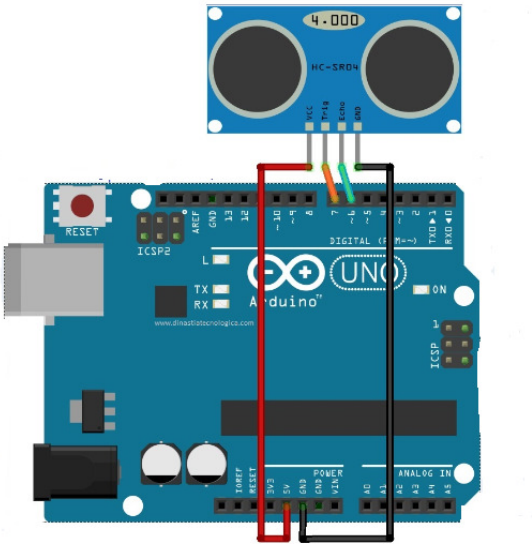


***Figura 4.6. (Derecha)*** *Conexión de un HCSR-04 con la placa UNO.*

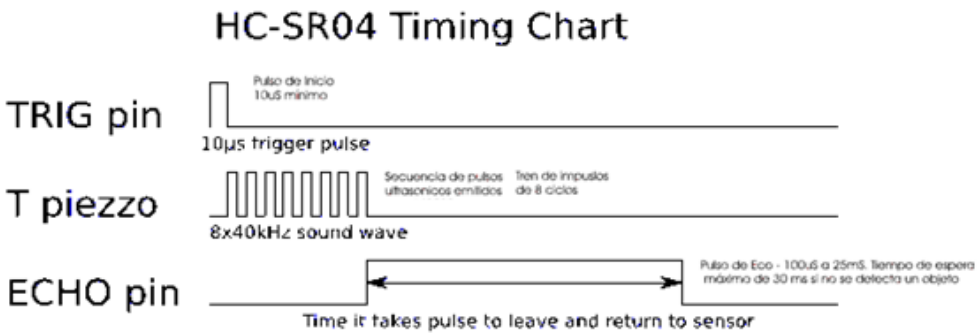


***Figura 4.7.*** *Diagrama de pulsos del HCSR – 04. [34]*

### 4.1.5. Puente H

A la hora de implementar físicamente el prototipo y llevar a cabo la puesta en marcha, aparece un problema, las señales PWM generadas desde el microcontrolador, no invierten la polaridad de los motores, es decir, siempre giraran en un sentido, eliminando la opción de girar a la izquierda y de ir marcha atrás. Esto ha llevado a la búsqueda de soluciones para invertir el sentido de giro de los motores y desbloquear por tanto las restricciones. Para ello, se planteó la posibilidad de incorporar al circuito un puente H, concretamente el L293D, muy utilizado para toda clase de vehículos teledirigidos electrónicos. Sin embargo, el coche del que se disponía en el laboratorio, ya incluía un puente H además de todo el circuito para su acondicionamiento, por tanto se optó por sacar el máximo rendimiento a éste.

El puente H del vehículo teledirigido pertenece a un conjunto llamado receptor del coche radiocontrolado de n canales. El esquema del circuito depende de los canales que tenga el coche. Los canales son las acciones independientes que puede realizar. El prototipo presenta un esquema de cinco canales, es decir, presenta las funciones atrás-adelante-turbo e izquierda-derecha. En este tipo de casos no es útil emplear distintas frecuencias para cada opción, de manera que se emplea modulación digital. En la figura 4.8 se han coloreado algunas secciones con el fin de facilitar la compresión del esquema. Se ha de añadir que se ha suprimido la zona del circuito equivalente a la antena de radiofrecuencia, ya que se ha implementado la comunicación Bluetooth.

**Sección A: Amplificación de audio.**

El integrado SCRX2B incorpora dos amplificadores inversores. Las pastillas exteriores se conectan con lo que sería equivalente a las entradas inversoras. Además las resistencias y condensadores actúan como redes de alimentación de los dos amplificadores. Se ha de añadir que no se hace uso de esta etapa y por tanto no se va a pasar a detallar las características de dicho integrado.

**Sección B: Alimentación.**

Esta sección corresponde a la alimentación del circuito. Se ha de destacar el diodo que previene contra la inversión de las baterías, así como algunos condensadores de filtrado. En el esquema no se aprecia pero la parte que alimenta a la etapa de la antena va desacoplada mediante una resistencia de 100Ω y un condensador. Sirve para que ninguna señal residual de RF pueda filtrarse a la línea de alimentación e interferir con el integrado. En algunos circuitos esta parte no está bien

diseñada, y se acopla la RF con la alimentación, también puede pasar por medio de las capacidades parásitas entre las pistas por ejemplo.

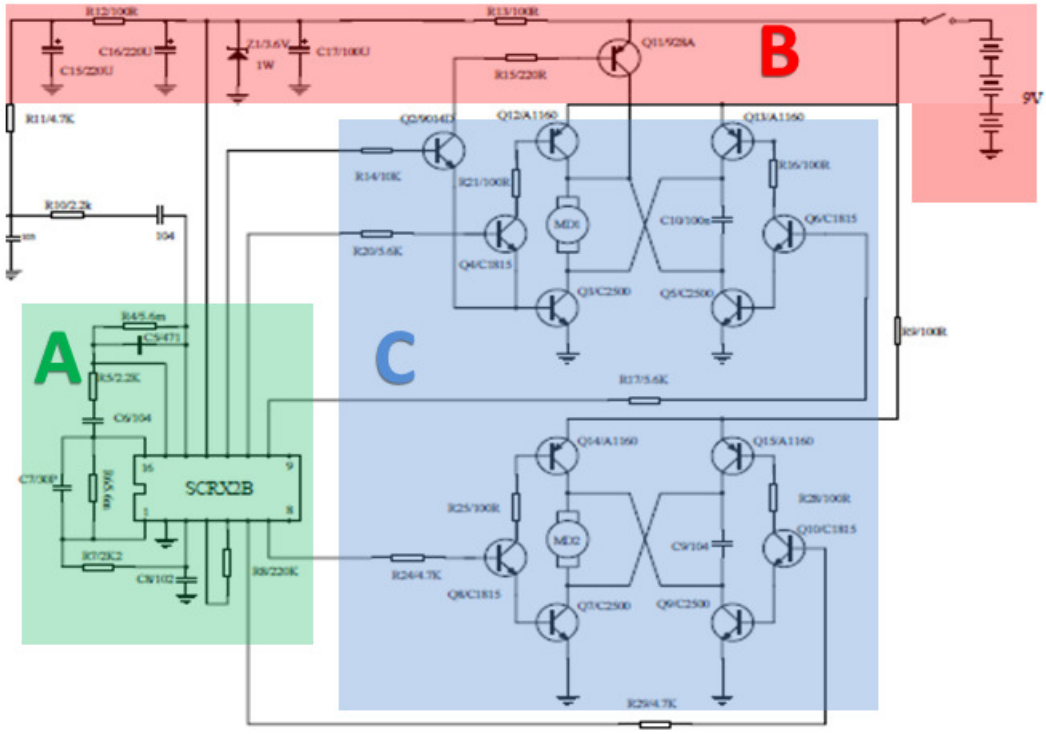


***Figura 4.8.*** *Esquema funcionamiento del conjunto receptor de 5 canales. [35]*

**Sección C: Puente H.**

Cuando aplicamos tensión a un motor este gira en una dirección determinada. Si lo que se quiere es que gire e un sentido o en otro a voluntad tenemos que usar una disposición especial de transistores para alimentarlo. Este circuito se llama puente H. Cuando el integrado aplica tensión a la patilla equivalente al avance, un transistor 1A pasa a conducción. Con él como una reacción en cascada también conmutan los transistores siguientes, 2A y 3A, poniendo a masa el terminal izquierdo del motor y suministrando tensión positiva al derecho. Y el motor girará en un sentido.

En cambio, cuando se activa la patilla de retroceso se activa el transistor 1B que a su vez activa 2B y 3B. En están condiciones, el terminal izquierdo del motor recibiría tensión positiva mientras que el derecho se conecta a masa. Justo la situación inversa a la anterior, y el motor girará en sentido contrario.

## CONSIDERACIONES

Se debe tener en cuenta que al apagar un motor, se produce un pico de tensión de polaridad opuesta, debido a esto, no es recomendable cambiar bruscamente el sentido de giro del motor, es más adecuado frenarlo antes.

Para la realización de las pruebas, antes de conectar y realizar el montaje final, el prototipo se alimenta mediante el ordenador a través del puerto serie del microcontrolador. Más tarde, se sustituirá el ordenador por una batería de 9V colocada sobre el coche para la alimentación autónoma.

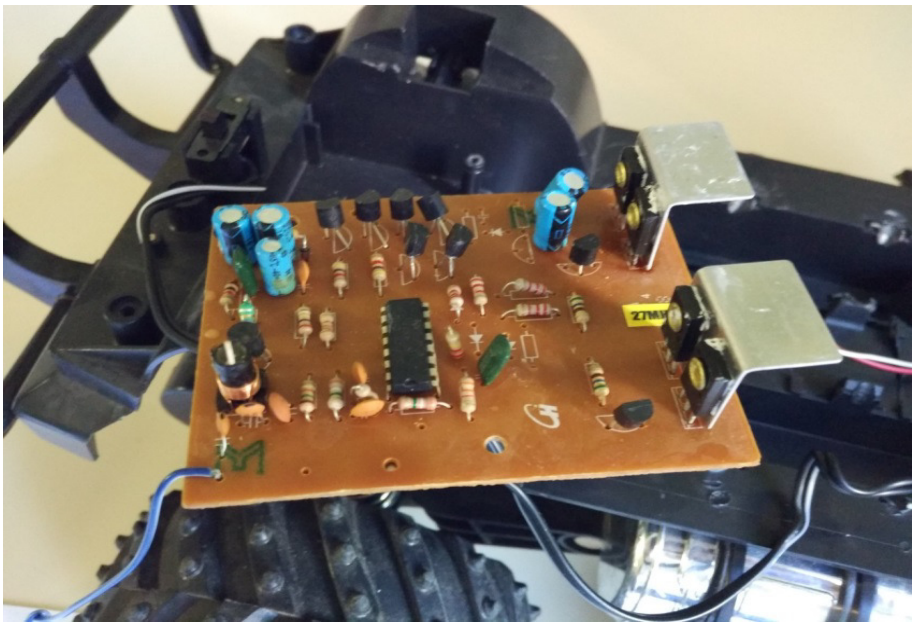

***Figura 4.9.*** *Circuito acondicionador y puente H del prototipo.*

### 4.1.6. Comunicación Bluetooth

Las comunicaciones Bluetooth son un protocolo estándar de comunicación inalámbrica, emisión y recepción de información, entre distintos dispositivos. Presenta código abierto de manera que no se requiere licencia, opera en la banda ISM (médico-científica-internacional) a una frecuencia de 2.4GHz. El Bluetooth también es un protocolo seguro, de bajo coste e ideal para pequeñas distancias. Estos son los parámetros más importantes:

- Nombre: cadena de caracteres del dispositivo definido por el fabricante.
- Dirección MAC: identificador numérico del dispositivo, es única para cada módulo. Los 24 primeros bits son establecidos por el IEEE y los últimos por el fabricante. El módulo empleado en el proyecto se describe a continuación:

| | |
|---|---|
| Nombre | HC-06 |
| MAC | 98:D3:31:FC:31:14 |

***Tabla 4.2.*** *Módulo Bluetooth empleado*.

Respecto al control de cuándo y a quién envían los datos se emplea el modelo maestro/esclavo, de forma que cada maestro se pueda conectar a cualquier esclavo para enviar o recibir datos, mientras que el esclavo solo pueda realizar conexión con un único maestro. Concretamente, la aplicación móvil actuará de maestro y el esclavo será nuestro dispositivo Bluetooth HC-06, que se encargará de recibir datos y enviarlos a la placa.

Cabe decir, que para que dos dispositivos sean compatibles, deben soportar el mismo perfil; un perfil es un protocolo adicional que define el tipo de información a transmitir entre ambos elementos, es decir, que se ha de habilitar un puerto serie para el Bluetooth, para cada uno de los lenguajes (Processing y Arduino), como se menciona en los siguientes capítulos. El perfil empleado es el SPP (*Serial Port Profile*), que emula una línea de comunicación serie, se ha utilizado a partir del puerto UART del microcontrolador mediante la línea de comunicación serie RX y TX.

## 4.2. Prototipo

### 4.2.1. Esquemático

Para el diseño del esquemático y la placa de circuito impreso, se optó por el programa de diseño FRITZING, la versión más actualizada, en lugar de otros como por ejemplo EAGLE. El esquemático se presenta a continuación:

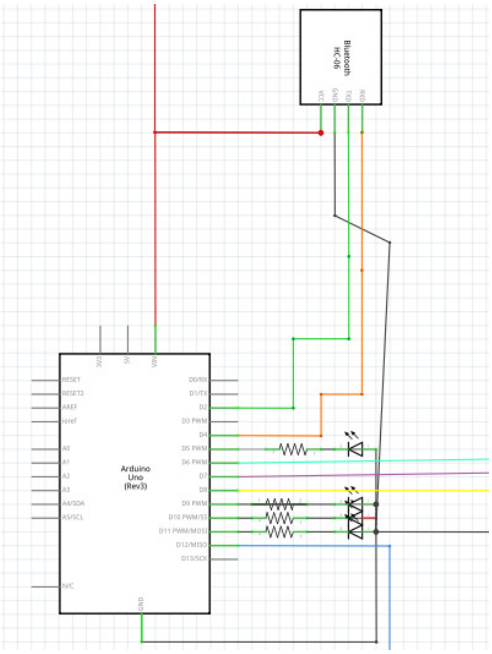


***Figura 4.10.*** *Esquemático del prototipo. Parte I.*

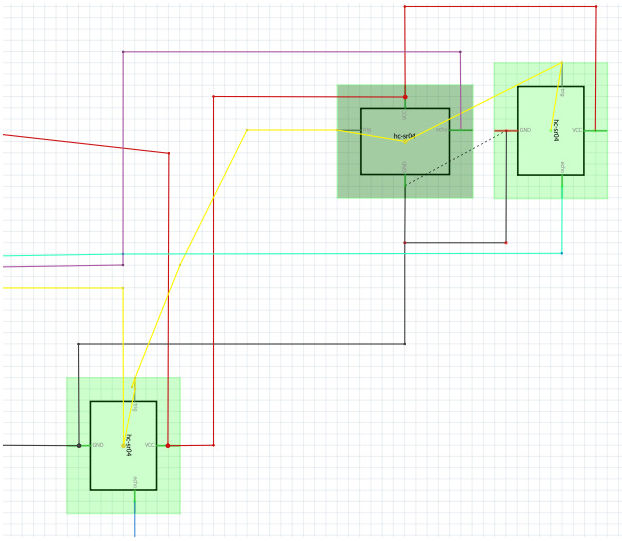

***Figura 4.11.*** *Esquemático del prototipo. Parte II.*

Cabe destacar la ausencia del puente H, el esquemático de éste se ha representado anteriormente, en el apartado 4.1.6.; en su lugar se han sustituido con resistencias y leds, lo que equivaldría a los pines de entrada del puente. En consecuencia tampoco se representan ni el motor de dirección y ni el de las ruedas motrices, en el siguiente apartado se describe el montaje final (Figura 4.16).

## 4.2.2 Montaje Final

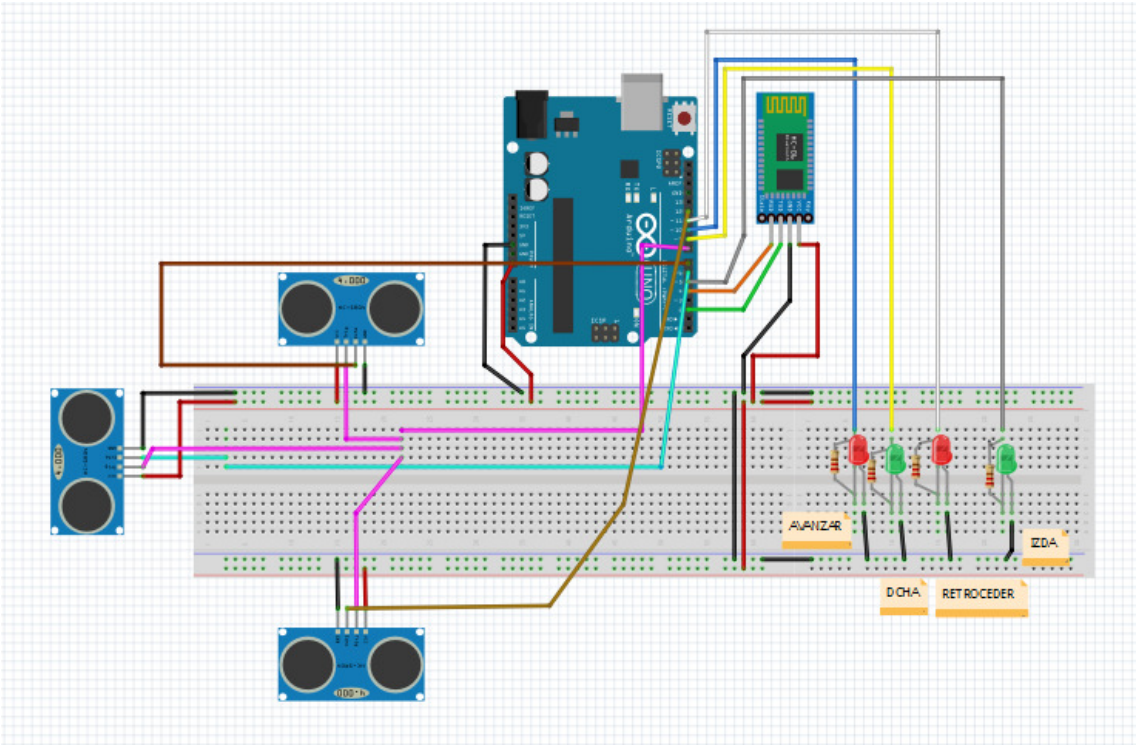


***Figura 4.12.*** *Captura de la protoboard con las salidas PWM derivadas a Leds.*

### ASIGNACIÓN PINES

La alimentación de la placa de Arduino se realiza mediante el pin Vin a través de una fuente de tensión externa (batería del coche, 9.6V); no obstante también puede ser alimentado mediante el puerto serie. Para la alimentación del módulo Bluetooth y de los sensores se ha empleado el pin 5V, que como su propio nombre indica nos proporciona cinco voltios.

Respecto a la conexión con el módulo Bluetooth, la salida Tx del dispositivo es asignada al pin 2, que equivale al Rx del Arduino, y por otra parte la entrada Rx del Bluetooth se conecta al pin 4 de la placa de Arduino.

En cuanto a la conexión con el puente H, los pines a asignar son cuatro, dos de ellos equivalen al motor que controla el giro dcha/ida, y los otros dos corresponden a la marcha adelante o atrás. Se describen en la Tabla 4.3.

| | Pines |
|---|---|
| DCHA | 9 |
| IZDA | 5 |
| ADELANTE | 10 |
| ATRÁS | 11 |
| RXArduino | 2 |
| TXArduino | 4 |
| Pecho_av | 6 |
| Pecho_dcha | 7 |
| Pecho_izda | 12 |
| Ptrig_general | 8 |

***Tabla 4.3.*** *Tabla de asignación de los pines a la placa de Arduino.*

Por último, se ha de mencionar la conexión de los sensores HCR-04; el trigger de los tres sensores va conectado a un mismo pin, el pin ocho del Arduino, mientras que las salidas ECHO de los sensores, se conectan a los pines seis, siete y doce; esto es así porque nos interesa obtener la información de la distancia de manera independiente para cada sensor.

# 5. Instrumentación y Sistema de Control. Implementación Software.

## 5.1. Implementación Software

Por lo que se refiere a este apartado, se va a describir la programación del módulo receptor (microcontrolador). Para ello, se ha expuesto en el capítulo anterior la implementación Hardware. En primer lugar se comenzará exponiendo el programa principal y acto seguido la comunicación serie establecida entre el microcontrolador y los sensores por un lado, y por otro la establecida con el módulo Bluetooth.

### 5.1.1. Programa Principal

El funcionamiento del código en el que se basa la aplicación se aprecia de manera esquemática en la figura 5.1. Es un programa diseñado para que se mantenga a la espera hasta que recibe una acción externa. Dicha acción es que se pulse alguno de los botones de la aplicación móvil y se produzca la transmisión y posterior almacenamiento de datos en el buffer del puerto serie. Esto hace que se inicie el proceso de dirección del vehículo. Cada vez que se finaliza uno de los ciclos mencionados se devuelve el programa a su estado de reposo inicial.

En esta primera parte se presenta un bucle que continuamente comprueba el puerto serie, es decir, es un bucle infinito, y del que únicamente se puede salir reseteando la placa del microcontrolador. Como se ha mencionado en el párrafo anterior, el bucle de esta primera parte se mantiene a la espera comprobando asiduamente si el puerto serie se encuentra disponible, esto es, si se ha recibido algún dato y se ha almacenado en el buffer.

El bucle comienza comprobando si hay algún usuario conectado. Una vez conectado alguien espera hasta que este haga alguna acción a través de los botones de la aplicación móvil que se ha creado, en la que se puede seleccionar el modo de funcionamiento del vehículo, modo manual o automático.

El siguiente diagrama muestra de forma gráfica el procedimiento descrito:

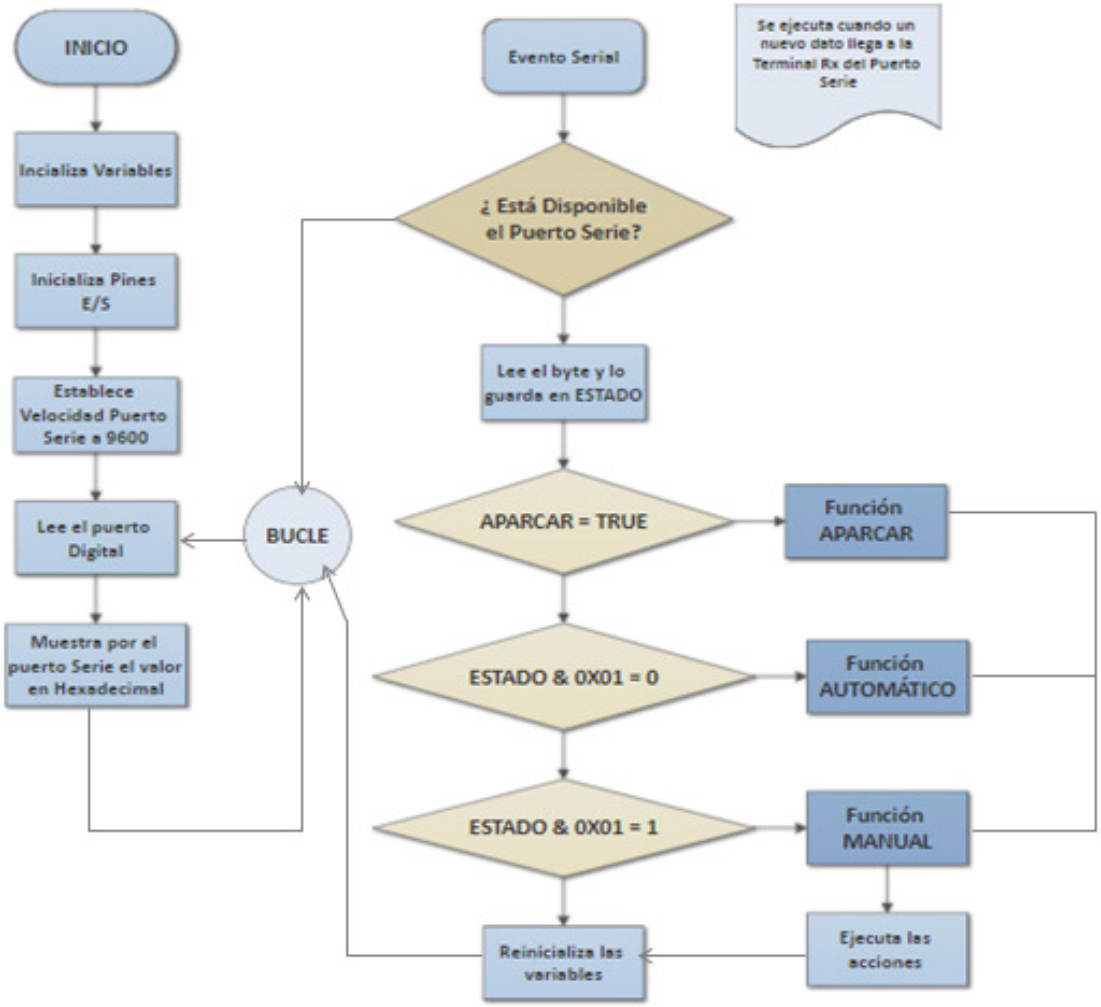


***Figura 5.1.*** *Diagrama de flujo del programa principal.*

A continuación, se explicarán cada uno de los estados mostrados en este diagrama.

## I. Reposo

Inicialmente el programa se encuentra en un estado de reposo. El programa consiste en un bucle en el que continuamente se comprueba el valor del pin asignado al RX de la placa, el pin 2. Previamente a este estado de reposo, se procede a la inicialización de las variables, los pines y de la comunicación serie (UART).
La configuración de los pines es sencilla, se va a utilizar un pin para introducir la duración del eco proveniente de cada sensor (pines 6, 7, 8 y 12) y los dos pines de la comunicación serie UART (pines 2 y 4). Aunque estos son indispensables para el desarrollo del programa, también se configuran otros pines con propósitos de comprobación y activación. Este es el caso de los pines asociados a los leds rojos y a los leds verdes (pines 5, 9, 10 y 11), que durante el proceso de creación del código se han ido utilizando a modo de comprobantes. Adicionalmente se inicializan las variables booleanas que posteriormente se utilizarán como referencia para el guiado del coche, así se podrá saber cuál es la pared lateral más cercana y por tanto, el lado hacia el cual está permitido el giro.

## II. Recepción de datos

Dentro del programa, se encuentra el bucle infinito, es aquí donde se ha introducido la petición de información como se ha comentado anteriormente. Cuando el buffer del puerto serie está disponible, es decir, existe almacenado algún dato recibido, es entonces cuando se procede a la lectura del mismo. Una vez leída la información, se asigna a la variable "*estado*", leída a lo largo de todo el código, a partir de ella se obtienen la velocidad, la dirección del coche y el modo de actuación.

## III. Acciones

Para acceder a la función aparcar se comprueba que el estado tiene en el cuarto bit menos significativo un uno, si es cierta la variable booleana "*aparcar*" toma valor verdadero y permite el acceso a esta modalidad. De forma similar ocurre con los otros dos modos; si el bit menos significativo de la variable "estado" toma el valor de cero, en este caso se activa el modo automático; si por el contrario toma un uno, la función que actúa es manual.

### 5.1.2. Comunicación Serie

Para iniciar la comunicación serie será necesaria una petición de información por parte del microcontrolador, que la aplicación móvil recibirá en forma de mensaje a través del puerto serie. Estos mensajes serán analizados siempre y cuando haya datos que poder enviar, y constan de diferentes partes (Capítulo 6) que aportan una información al receptor tanto sobre el propio mensaje como sobre la información que pide.

Las funciones principales que se deben saber emplear ara manejar el puerto serie son: begin(), read(), write(), print() y available().

- Begin(): Establece la velocidad de la UART (baudios) para la transmisión serie, es posible configurar el número de bits de datos, aunque por defecto por defecto son 8 bits de datos, sin paridad y un bit de stop.
- Read(): Se encarga de leer el primer byte entrante del puerto serie.
- Write(): Escribe datos en binario en el puerto serie.
- Print(): Imprime datos al puerto serie en forma de texto ASCII, para las pruebas de laboratorio se dispuso para que imprimiera en formato hexadecimal.
- Available(): Proporciona el número de bytes (caracteres) disponibles para su lectura en el puerto serie, son datos que se almacenan en el buffer serie, que tiene un tamaño de 64 bytes.

Se ha de añadir, que no se ha hecho uso del puerto serie asociado a los pines RX Y TX (cero y uno) de la placa de Arduino, sino que se ha empleado un puerto serie virtual en la plataforma Arduino. Cada microcontrolador tiene un número de puertos serie hardware (UART), pero por si existiera algún fallo en el hardware, Arduino desarrolló la librería Software Serial para permitir la comunicación serie sobre otros pines digitales de Arduino. Concretamente en este trabajo se emplean los pines dos y cuatro que equivalen al RX Y TX del Arduino.

### 5.1.3. Integración Sensor de Ultrasonidos

En primer lugar, se ha implementado un circuito de prueba en el que se ha comprobado que todos los sensores funcionaban correctamente, probándose individualmente uno por uno; dicho circuito consiste en la alimentación al módulo y la asignación de 2 pines de interfaz con el microcontrolador (ECHO y TRIGGER). Los sensores escogidos para medición de distancia a través de ultrasonidos se dividen en dos grupos:

1. Interfaz utilizando un pulso de eco.
2. Interfaz serial (I2C o UART).

El módulo seleccionado, el HC-SR04, pertenece al primer grupo; a continuación se describe brevemente su funcionamiento.

La SDI (Interfaz Digital Serial) se consigue mediante dos pines digitales: el TRIGGER (disparo) y el ECHO (eco).

- TRIGGER: recibe un pulso habilitante proveniente del microcontrolador, el cual indica al módulo sensorial cuando debe empezar a realizar la medida de la distancia.
- ECHO: el microcontrolador recibe del sensor un pulso cuyo ancho es proporcional al tiempo que ha tardado en recibir el eco.

La programación del sensor se ha llevado a cabo en la plataforma Arduino, y puede hacerse con o sin librería; en este proyecto la programación se ha llevado a cabo sin hacer uso de ninguna librería, se ha optado por utilizar la orden pulseIN (pin del ECHO, HIGH), que lee el tiempo del eco. Además hace falta generar el tren de pulsos, en este caso, no existe ninguna función específica y por tanto se ha de hacer tal y como se muestra en la figura 5.2.

```
int LeerDistancia(byte Pecho, byte Ptrig)
{
  unsigned long duracion;
  long distancia;

  digitalWrite(Ptrig, HIGH);   // genera el pulso de trigger por 10us
  delay(0.01);
  digitalWrite(Ptrig, LOW);

  duracion = pulseIn(Pecho, HIGH);              // Lee el tiempo del Echo
  distancia = (duracion/2) / 29;             // calcula la distancia en centimetros
  delay(10);

  return distancia;
}
```

***Figura 5.2.*** *Función LeerDistancia que incluye la función PulseIn.*

Por último, hay que destacar el hecho de que son tres los sensores de ultrasonidos que se han implementado, y puesto que no se disponía de pines suficientes en el microcontrolador, se optó por enviar el mismo pulso para los tres, es decir, los tres sensores tienen el pin TRIGGER en común, se puede apreciar en la figura 5.3.

```
const byte Pecho_av=6;    // Pine
const byte Pecho_dcha=7;
const byte Pecho_izda=12;
const byte Ptrig_general=8;

void setup()  {
  Serial.begin(9600);         //
  Blue.begin(9600);
  Serial.println("Empezamos");
  pinMode(dcha, OUTPUT);
  pinMode(izda, OUTPUT);
  pinMode(front, OUTPUT);
  pinMode(back, OUTPUT);

  pinMode(Pecho_av, INPUT);
  pinMode(Ptrig_general, OUTPUT);
  pinMode(13,OUTPUT);
  pinMode(Pecho_dcha, INPUT);
  pinMode(Pecho_izda, INPUT);
```

***Figura 5.3.*** *Asignación de pines de los sensores en la plataforma Arduino.*

### 5.1.4. Función Manual

Esta segunda parte se trata de una función que actúa directamente sobre los motores del vehículo mediante control PWM (Pulse Wide Modulation). En la sección anterior se ha expuesto brevemente como se accede a ésta en el programa principal, ahora se va proceder a la descripción detallada.

Básicamente la función lee en una variable de tipo char (estado) el dato recibido, de donde obtiene la velocidad (0-255), que posteriormente almacena en otra variable (vel). Para ello, lee los cuatro primeros bits del dato y los multiplica por dieciséis para obtener un byte; si bien es cierto que se comete un error, ya que se truncan los decimales, es muy pequeño y no va a tener mucha repercusión.

Adicionalmente, también se leen los cuatro bits menos significativos de la variable estado, que se asignan a la variable dirección. Esta variable es la encargada de decir al programa que acción se va a ejecutar sobre los motores (adelante-atrás-derecha-izquierda) y con qué intensidad (vel).

A continuación se muestra el diagrama asociado a esta función:

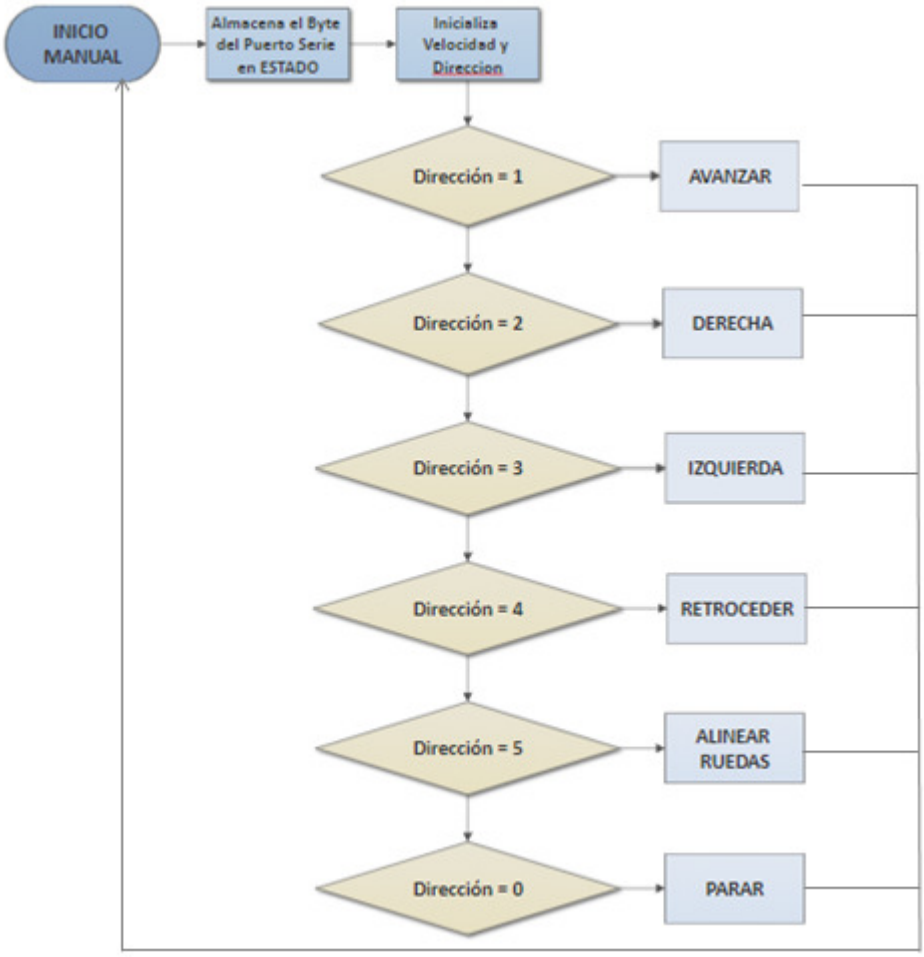


***Figura 5.4.*** *Diagrama de flujo de la función Manual.*

### 5.1.5. Función Automático

Para finalizar, se procede a la descripción de la última parte del programa. Se trata de una función más compleja que la anterior, la cual hace uso de otras funciones más simples. La primera sección se basa en los mismos principios que la función manual, por ello no se va a proceder a dar detalles, únicamente difiere en que el proceso es más largo y está condicionado por la interacción del vehículo con el medio. Una vez se ha pulsado el botón ON en la aplicación móvil la función Automático recurre a una función específica para leer las distancias enfrente, a la derecha y a la izquierda del prototipo. Una vez obtenidas, comprueba cual era el último lado al que se encontraba la pared más cercana, ya que por este lado no podrá dar la vuelta si se encuentra un obstáculo de frente; ahora bien, se ha de contemplar la alternativa de que el otro lado de la pared este lo suficientemente cerca para que no pueda girar por ese lado, en ese momento se encuentra en un callejón, y la única alternativa es dar marcha atrás durante un periodo de tiempo para ponerse en marcha de nuevo.

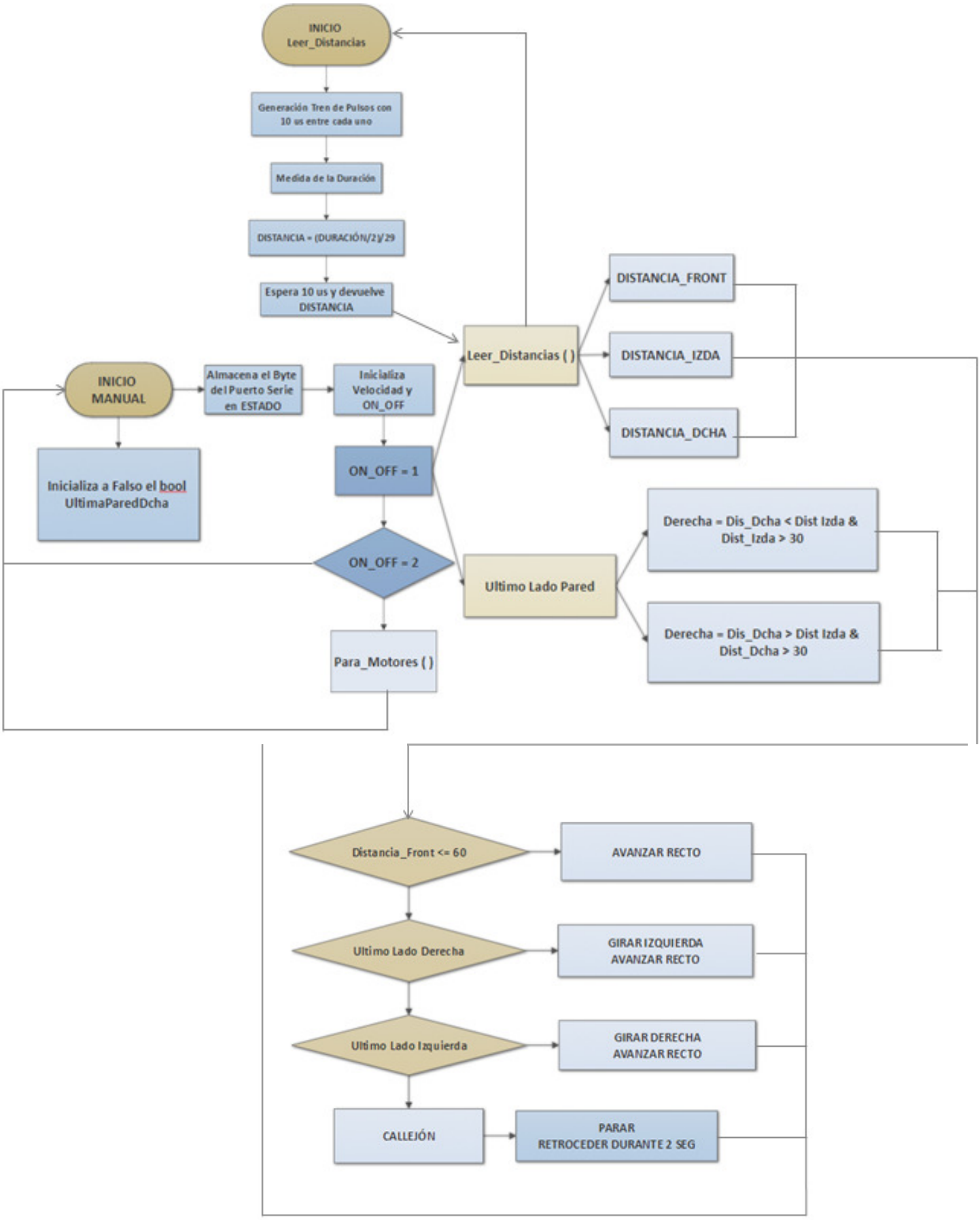


***Figura 5.5.*** *Diagrama de flujo de la función Automático.*

# 6. Aplicación Móvil

El siguiente Capítulo trata del desarrollo del módulo emisor como aplicación móvil; para ello se comienza por la descripción de la aplicación elaborada con el programa Processing, cuyo lenguaje de programación es Android. Seguidamente se complementa la descripción con un organigrama y se esclarece la misión de cada opción disponible, para el control del coche teledirigido. Así mismo, se ha creído conveniente la explicación del método de comunicación, entre el emisor, en este caso la APP, y el receptor, que será el micro de Arduino; sin embargo, no se va a centrar en el desarrollo de la comunicación Bluetooth, ya que se va a exponer en el capítulo siguiente. Finalmente, se argumenta cada modalidad de control del vehículo y se explica la forma en la que se ha hecho, a excepción de la opción Aparcar; para ello se ha desarrollado un Trabajo Fin de Grado aparte, elaborado por otro compañero, y por tanto, en este proyecto sólo se hace una mera mención.

Cabe notar, por motivos de espacio, no se ha incluido la parte completa del código, que se incluye al final de la memoria en los anexos; no obstante, se han añadido fragmentos de tareas o parámetros conforme se ha convenido necesario, de manera que revelara de forma clara el concepto que se pretende transmitir.

## 6.1. Organización y Estructura Básica

En lo que toca a la estructura de la aplicación, se divide en una serie de pantallas o pestañas, en las que se encuentran las funcionalidades principales para controlar el prototipo. Para ello, la App se ha segmentado en cuatro pantallas, como se aprecia en la Figura 6.1, estas pantallas o pestañas son:

- **CONFIGURACIÓN:** La *actividad* principal, a la que se accede automáticamente de forma predeterminada; se encarga de que la conexión con el módulo Bluetooth sea exitosa.
- **MANUAL:** Permite el manejo del vehículo teledirigido a través de la aplicación móvil. El usuario es el responsable del control, haciendo que el prototipo siga la trayectoria que se quiera.
- **AUTOMÁTICO:** Funciona con dos botones, uno de encendido y otro de apagado. Esta *actividad* restringe el control a la velocidad, es decir, el coche circula libremente ayudándose de los sensores para evitar obstáculos**.**
- **APARCAR:** Se indica a través de la aplicación el tipo de aparcamiento, cordón o batería, y el sentido, izda. o dcha., una vez hecho el vehículo aparca solo. Esta *actividad*, se explica en el trabajo de mi compañero.

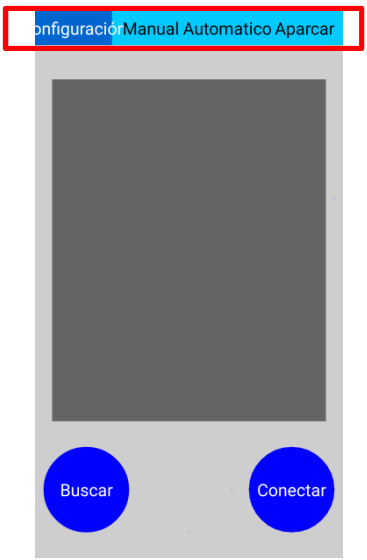


***Figura 6.1.*** *Pantalla principal de la aplicación con las diferentes pestañas.*

A continuación se muestra en la Figura 6.2, un organigrama de acceso a las distintas actividades representado mediante sus layouts.

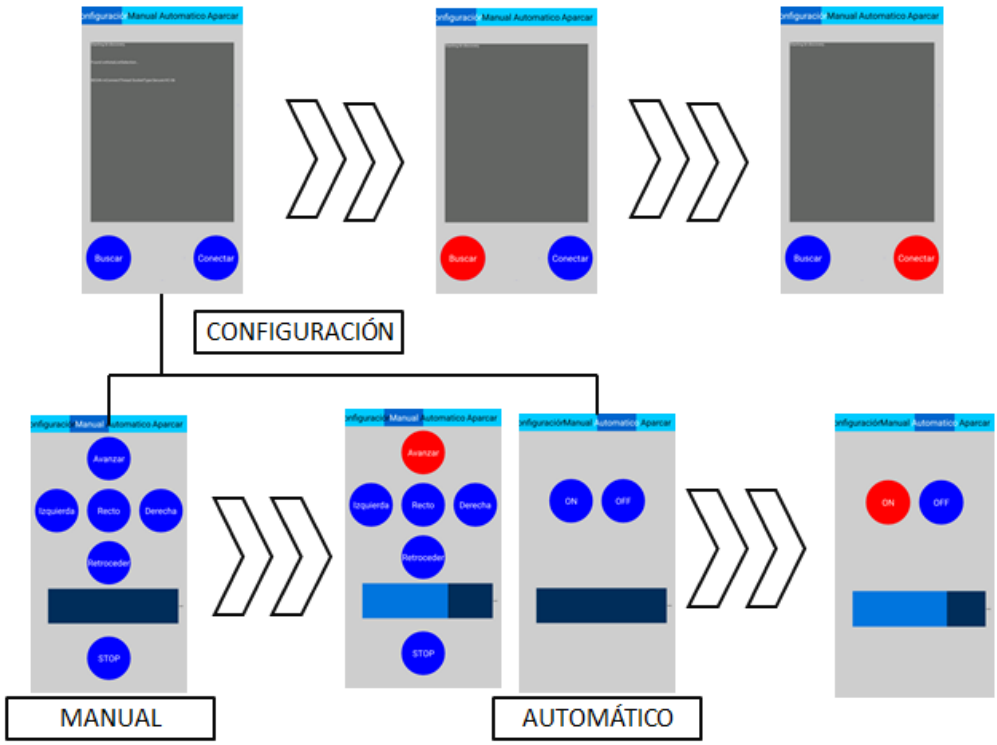


***Figura 6.2.*** *Organigrama de las funciones Configuración, Manual y Automático.*

El primer paso para empezar a desarrollar la aplicación, es saber que vamos a trabajar con una resolución 16:9 y que es indispensable saber la resolución del equipo donde se vaya a instalar la APDE (Figura 6.3); en el caso del dispositivo empleado la resolución son 1920x1080. Seguidamente se procede a desarrollar nuestro programa, para ello Processing trabaja con dos funciones principales que son:

- Función ***void setup()***, es la primera que se ejecuta cuando se empieza a correr la aplicación, aquí se definirán todos los datos que necesitemos antes de que comience el programa, cabe notar que solo se ejecuta una vez.
- Función ***void draw()***, es la función encargada del procesamiento del programa.

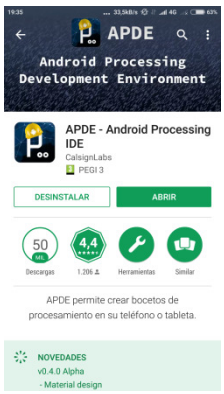


***Figura 6.3.*** *Pantalla principal de descarga de la aplicación para móvil de Processing*

En segundo lugar, se describe el sistema de coordenadas de la pantalla pretendiendo trasladar una hoja cuadriculada a ésta; las pantallas se dividen en píxeles, y en ocasiones la resolución de éstas puede ser muy elevado y resulta difícil trabajar con los interfaces gráficos. Por tanto, se dibuja un rectángulo sobre la pantalla del móvil de nueve centímetros ancho por dieciséis centímetros de alto, al hacer esto se tienen dieciocho cuadrados de ancho y treinta y dos cuadrados de alto; básicamente se pretende trabajar con una unidad mínima de resolución de pantalla, que será una zona de píxeles cuadrada con una superficie de cinco milímetros cuadrados, para saber dónde se van a colocar los elementos gráficos, como por ejemplo los botones.

Así mismo, si tomamos la cantidad de píxeles horizontales de nuestro dispositivo móvil y la dividimos entre 18, tenemos la unidad mínima de trabajo; de igual forma, tomamos los píxeles verticales y los dividimos, esta vez entre 32.

***uX = CantidadDePixelesEnX / 18 = 1080 / 18 = 60***

***uY = CantidadDePixelesEnY / 32 = 1920 / 32 = 60***

***uX = uY = 60***

Esto quiere decir que se está trabajando con una unidad de 144 píxeles por milímetro cuadrado, o lo que es lo mismo 12 píxeles/mm.

Para ilustrar mejor esta sencilla técnica, se muestra un fragmento de código de la pestaña configuración, más concretamente, la incorporación del botón “Buscar” en el programa (Figura 6.4 y Figura 6.5).; sus coordenadas son 3*uX en horizontal y 28*uY en vertical. El origen de coordenadas de la pantalla se encuentra en la esquina superior izquierda.

```
42
43  Boton1 = new BotonRedondo(3*uX,28*uY,5*uX,color(0,0,255),color(255,0,0),
44                            "Buscar",uY,color(255,255,255));
```

***Figura 6.4.*** *Código de la disposición de un nuevo botón en la aplicación móvil.*

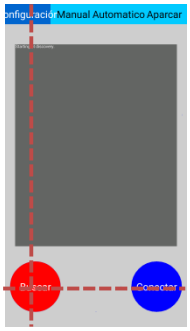


***Figura 6.5.*** *Situación del botón “Buscar”.*

Finalmente, se ha optado por simplificar el código completo del programa. Para ello, además de las funciones principales *setup* y *draw,* se ha creado para cada elemento gráfico de la APP, una clase distinta, es decir, que para el elemento botón, citado anteriormente, se elabora una clase pública llamada "BotonRedondo", de acuerdo con la librería ControlP5.

### 6.1.1. Librería ControlP5

El siguiente punto va a describir la implementación de elementos como los botones, el área de texto, así como la forma de trabajo de Processing con éstos. Primero de todo, se debe hacer una breve introducción a la librería que nos va a proporcionar los elementos, se llama ControlP5 y es de código abierto, lo que significa que se puede descargar desde la página web de Processing.

Si se hace el ejercicio mental de imaginar la pared de una casa, la cual se quiere decorar, la tarea de decorar sería el programa Processing, así la pared como tal equivaldría a la pantalla donde vayamos a poner los elementos y los cuadros a colgar corresponderían a los objetos que se van a insertar en la pantalla; para incorporar esos objetos a la pared (pantalla) lo ideal es tener un molde, para que se puedan duplicar, y en caso de necesitar decorar otra pared, no tener que volver a hacerlos; estos moldes, de los cuales se va a sacar las figuras, en programación, se llaman clases. Una clase es un fragmento de código que describe un objeto que se va a emplear en programación, en este caso, las clases van a ser elementos gráficos. No obstante, todos los elementos no van a ser visibles físicamente, concretamente la conexión Bluetooth, se encontrará oculta a la vista.

Desde el punto de vista de la programación, importamos la librería con el comando import controlP5*. Acto seguido se crea una variable de tipo P5, esa variable guardará un objeto, y éste habilitará que los demás objetos puedan incorporarse a la interfaz gráfica. ControlP5 Cp5. (Figura 6.6). Ahora se va a crear el objeto de esa variable mediante la orden Cp5 = new ControlP5( this); ***this*** es un parámetro que representa el programa donde estamos creando el objeto. Una vez creado el objeto Cp5, no hay más que invocar los elementos de la librería que se quieran incorporar de la forma :

**Cp5.add "elemento"(Nombre del elemento). setPosition(Coordenadas en la pantalla).**

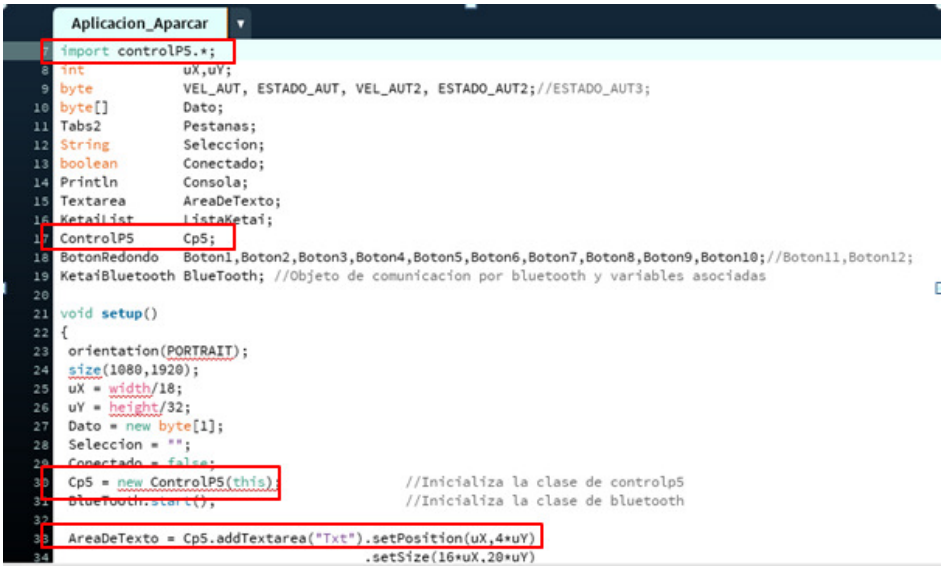


***Figura 6.6.*** *Comandos de la librería ControlP5.*

## 6.2. Configuración

Por lo que se refiere a esta primera pestaña, la actividad Configuración (Figura 6.7) presenta dos botones que permiten establecer comunicación con el módulo Bluetooth y un cuadro gris de texto en mitad de la pantalla, que muestra si se han encontrado dispositivos y si se ha logrado con éxito o no la conexión; en definitiva, el comando de texto es un elemento gráfico interactivo con el usuario que le informa del estado de las comunicaciones.

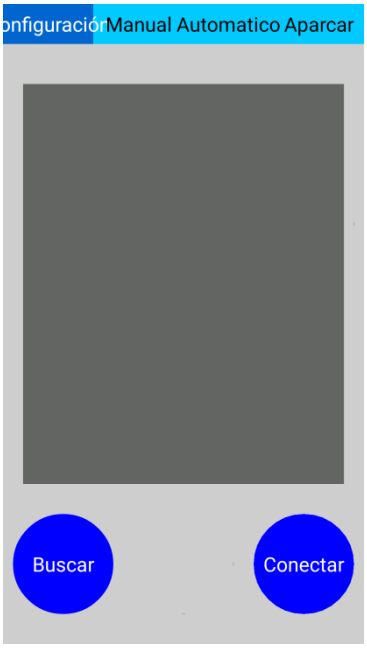


***Figura 6.7.*** *Pantalla de la actividad Configuración.*

Respecto a las funciones de búsqueda y conexión, presenta un botón redondo azul con la palabra "*Buscar*", el cual se encarga de, como el propio nombre indica, realizar una búsqueda de los dispositivos detectables y listar todos los dispositivos con sus respectivos nombres. Una vez se ha pulsado éste botón, se puede clicar en el botón "*Conectar*", que muestra la lista elaborada anteriormente y será el usuario quién seleccione el nombre del dispositivo objetivo. En caso de que la conexión sea satisfactoria saldrá un mensaje (Figura 6.8) indicando al usuario de que puede comenzar el tráfico de datos con el dispositivo. Por el contrario, si la conexión ha sido fallida, se muestra en la pantalla un texto con la frase " *Conexión Fallida*".

Se debe agregar que la consola, es un objeto que captura determinada información que Android proporciona, y que no se puede ver; por otra parte, el control que nos permite visualizar esta información es el área de texto (rectángulo gris de la pantalla, Figura6.6). Ambos elementos van unidos.

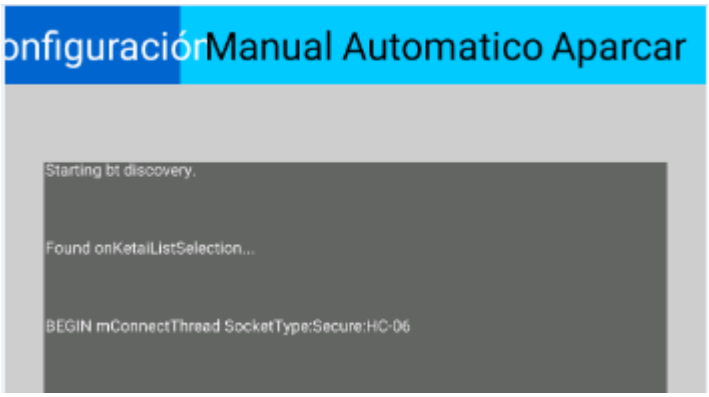


***Figura 6.8.*** *La conexión con el dispositivo ha sido un éxito.*

El diagrama de flujo general y de esta actividad se encuentra en las Figuras 6.10 y 6.11.

Respecto al diagrama de flujo de la actividad "Configuración" cabe decir, que el botón "Buscar" limpia el área de texto y obtiene la lista de los dispositivos Bluetooth emparejados; mientras que el botón "Conectar", es el encargado de crear una nueva lista de selección en pantalla para que el usuario elija un dispositivo Bluetooth. Los fragmentos de código se muestran en la figura 6.9.

```
//Acciones Botones
void AccionesBoton1()    // Boton Buscar el bluetooth
{
 AreaDeTexto.clear();
 BlueTooth.discoverDevices();  //Obtiene la lista de dispositivos bluetooth
}

void AccionesBoton2()        // Conectar con Bluetooth
{
 if (BlueTooth.getDiscoveredDeviceNames().size() > 0)
  ListaKetai = new KetaiList(this, BlueTooth.getDiscoveredDeviceNames());
 else
  if (BlueTooth.getPairedDeviceNames().size() > 0)
   ListaKetai = new KetaiList(this, BlueTooth.getPairedDeviceNames());
}
```

***Figura 6.9.*** *Acciones de los botones de "Configuración".*

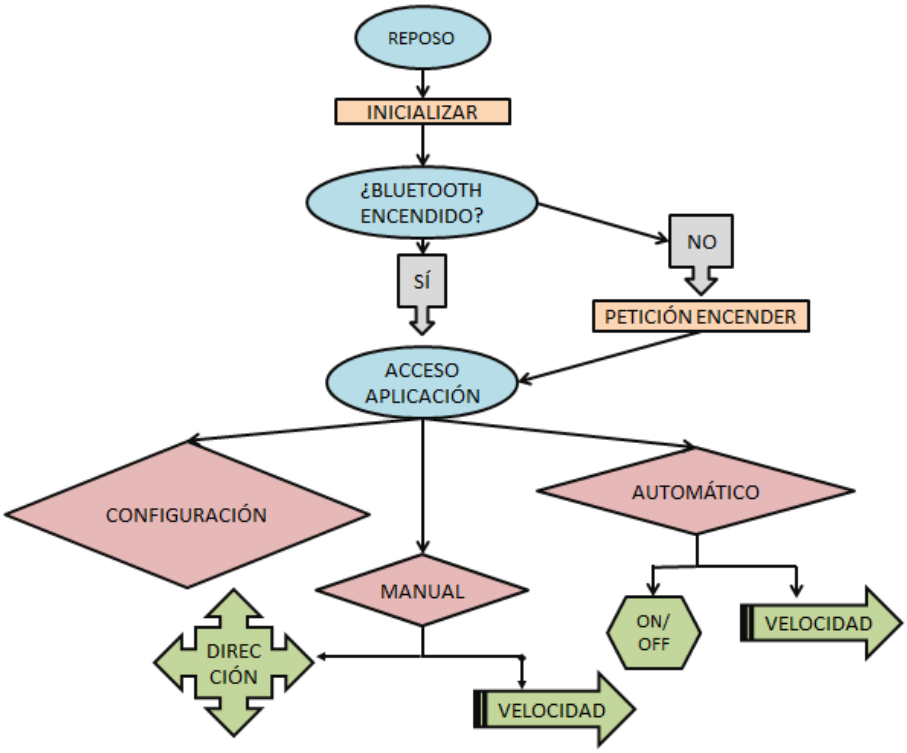


***Figura 6.9.*** *Diagrama de flujo del programa principal.*

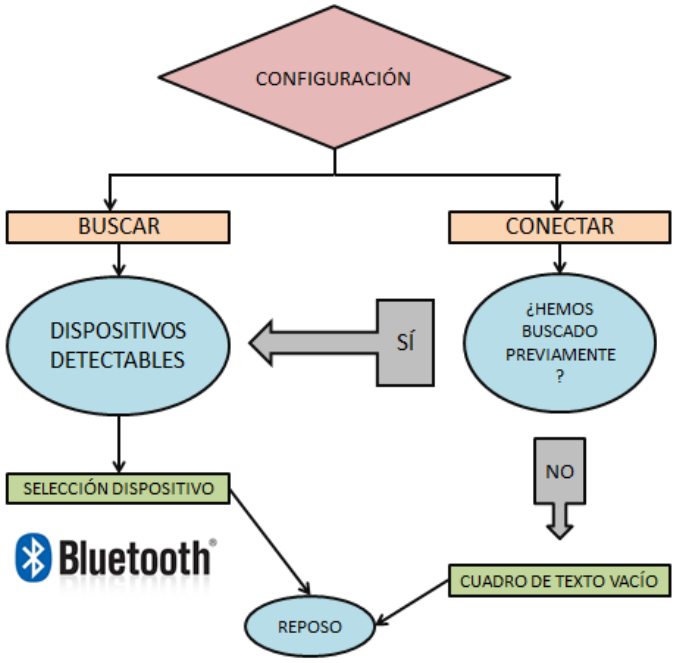


***Figura 6.10.*** *Diagrama de flujo de la actividad Configuración.*

## 6.3. Modo Manual

La actividad Manual (Figura6.11) presenta seis botones que permiten modificar el estado del vehículo; además se ha incorporado una barra deslizante, con rango de cero a doscientos cincuenta, en la que el usuario seleccione la velocidad a la que desea que el coche funcione. En este modo de funcionamiento el control del vehículo es exclusivo del usuario.

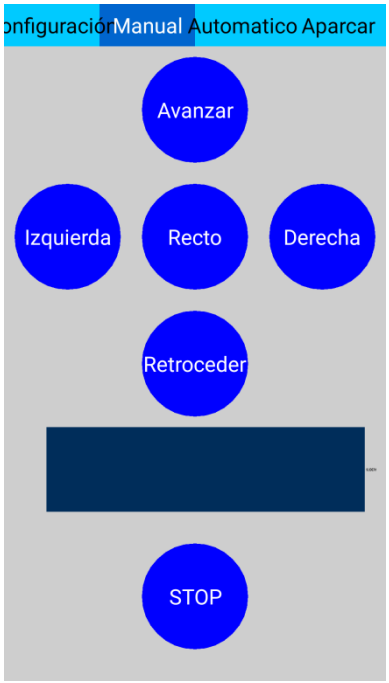


***Figura 6.11.*** *Pantalla de la actividad Manual.*

**Botones:**

La actividad posee varios controles que pueden ser usados por el usuario, cada uno de ellos asociados a una respuesta:

- AVANZAR: Acelera el motor de continua en un sentido.

- DERECHA: Pone en funcionamiento el servo hacia la derecha.

- IZQUIERDA: Lo mismo que el control a la derecha pero en sentido inverso.

- RECTO: El motor de dirección dispone de un tope que limita su giro, tanto por la derecha como por la izquierda, de forma que no existe un estado intermedio en el que las ruedas queden rectas. Para ello, hace falta incluir una orden que nos impida girar en ambos sentidos.

- RETROCEDER: Igual que la acción avanzar, pero sentido inverso.

- STOP: Finalmente, el botón Stop simplemente para los motores, es decir, envía una señal PWM a través de los pines de salida con valor cero.

Señalar que estos controles no pueden estar activos simultáneamente, sino que su atributo de visibilidad se alterna entre Visible e Invisible. También se ha de destacar, que cada vez que se hace girar un motor en un sentido, mediante PWM, se anula el giro en el sentido inverso, es decir, para ordenar un movimiento hacen falta dos órdenes, sino se podría dar lugar a fallos eléctricos en los motores.

Además, los botones tanto de la actividad Manual como Automático sólo actúan si previamente se han conectado el dispositivo móvil, con el módulo Bluetooth objetivo.

Cabe añadir, que para salir de la ventana no existe ningún botón que devuelva al usuario a la pestaña Configuración, que sería la principal, sino que en la parte superior, se han añadido unas pestañas que permiten el libre acceso de una actividad a otra. Las pestañas seleccionadas se colorean de azul marino.

## 6.4. Modo Automático

En cuanto a la última pestaña, “Automático” (Figura 6.12), está compuesta de una barra deslizante y dos botones, el botón de encendido y el de apagado respectivamente.

Este modo de funcionamiento permite al usuario un control parcial del coche, pudiendo únicamente activar, desactivar y variar la velocidad del vehículo.

**Botones:**

- ON: Activa el modo automático, el coche comienza a avanzar hasta que ve un obstáculo, entonces detecta la menor de las distancias a los lados, y selecciona el lado por el que girará; en caso de no haber espacio a los lados para dar la vuelta hace marcha atrás durante un periodo determinado de tiempo.
- OFF: Desactiva los motores, enviando una señal PWM de valor 0. Cabe decir, que aunque se pulse este botón, el modo automático sigue activo, por tanto, se puede poner en funcionamiento de nuevo accionando ON.

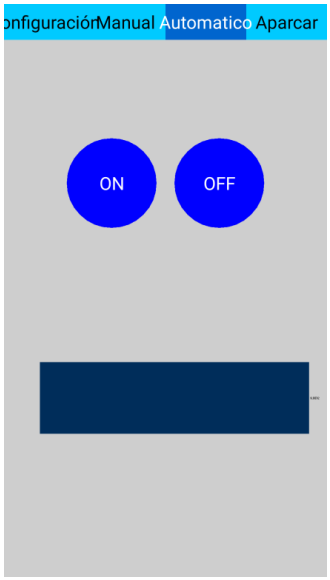


***Figura 6.12.*** *Pantalla de la actividad Automático.*

## 6.5. Protocolo de Comunicación

La estructura es tal que toda la información necesaria se codifica en un byte. La información que se quiere transmitir va a ser:

- La velocidad del vehículo. Esta velocidad se genera modulando una señal PWM que se mandará a la entrada del puente H correspondiente. Esta salida PWM se codifica en un byte (valores 0-255). A la hora de transmitir la información, ésta se codifica en los cuatro bits más significativos del byte de datos, de manera que la resolución es de 15 valores.

- El modo de funcionamiento que ha seleccionado el usuario; son dos, automático o manual (además se ha incluido un tercer estado equivalente al modo de aparcamiento, se detalla en el TFG de Javier Roche). El estado se codifica en el bit menos significativo del byte de datos.

- Por último, la dirección que se quiere que tome el coche, ya sea adelante, derecha, izquierda o hacia detrás; también se han añadido situaciones extra, con que el coche se pare, o que se alineen las ruedas una vez haya salido de un giro.

| Vel | Vel | Vel | Vel | Dir | Dir | Dir | Modo |
|---|---|---|---|---|---|---|---|

Resumiendo, debido a la velocidad, el estado y los casos que se han contemplado para el control del vehículo, se requieren un total de ocho bits, esto es, un byte, que se almacenará en la primera componente del vector Dato[]. El vector se hace necesario por exigencias de las funciones utilizadas para el control del Bluetooth.

### 6.5.1. OPCION MANUAL

| V | V | V | V | 1/0 | 1/0 | 1/0 | 1/0 |
|---|---|---|---|---|---|---|---|

- Bit 0: indica el modo de funcionamiento del prototipo, en este caso vale uno
  - Manual → bit0 = 1.
  - Automático → bit0 = 0.
- Bits 1, 2 y 3: selecciona la dirección que va a tomar el prototipo
-
  - Avanzar →bit1 = 1 (V V V V | 0 0 1 | 1)
  - Derecha → bit2 = 1. (V V V V | 0 1 0 | 1)
  - Izquierda → bit1 = 1, bit2 = 1. (V V V V | 0 1 1 | 1)
  - Retroceder → bit3 = 1. (V V V V | 1 0 0 | 1)
  - Stop → (V V V V | 0 0 0 | 1)
  - Recto → bit1 = 1, bit3 = 1. (V V V V | 1 0 1 | 1)

La velocidad se toma de la barra deslizante como un real (“ValorRango”), posteriormente se le hace un cast al tipo entero (“valorRangoInt”), también se le hace un AND (&) con el byte “11110000” con el fin de quedarnos sólo con la información de los cuatro primeros bits; finalmente se le hace un cast al tipo byte y se almacena en la variable “VEL_AUT”. Se puede apreciar en la Figura 6.13.

```
325 void Slider1(float ValorRango)              //ValorRango es un real
326 {
327  if(Conectado)
328  {
329    int valorRangoInt = (int) ValorRango;
330    VEL_AUT = (byte)( valorRangoInt & 0xF0);
331  }
332 }
```

***Figura 6.13.*** *Registro de la velocidad mediante la barra deslizadora.*

### 6.5.2. OPCION AUTOMÁTICO

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| V | V | V | V | 1/0 | 1/0 | 1/0 | 1/0 |

- Bit 0: indica el modo de funcionamiento del prototipo, en este caso vale cero
  - Manual → bit0 = 1.
  - Automático → bit0 = 0.
- Bits 1, 2 y 3: indica el momento en el que se manda la señal de activación del programa:
  - ON → bit1 = 1 (V V V V | 0 0 1 | 0)
  - OFF → bit2 = 1. (V V V V | 0 1 0 | 0)

# 7. Comunicaciones Bluetooth

A continuación se pasa a detallar el método necesario para la comunicación inalámbrica entre el sistema Arduino y la aplicación móvil, mediante el módulo Bluetooth. Para ello en primer lugar se realiza un programa sencillo en el que la aplicación emule acciones básicas, elaborando el sistema equivalente en Arduino, para comprobar que la conexión entre el móvil y el microcontrolador es correcta. Dichas acciones establecen el envío de información a través de la aplicación Android y la recepción de ésta por parte del sistema Arduino. Posteriormente se describirá en este capítulo el método empleado para la transmisión de información.

## 7.1. Estructura Básica de la comunicación Bluetooth

La comunicación inalámbrica entre la aplicación móvil en Android y el sistema Arduino debe transmitir la información que el usuario está introduciendo en su dispositivo móvil a la plataforma Arduino.

Antes de poder utilizar el sistema Bluetooth resulta imprescindible incluir la librería necesaria para su manipulación.

### 7.1.1. Configuración Básica con Arduino

El siguiente punto trata de explicar cómo se ha logrado configurar el módulo Bluetooth tanto para el sistema Arduino, como para el lenguaje de programación (Android) de la aplicación móvil. En primer lugar se describe el módulo utilizado para la dotar a la placa Arduino de este tipo de comunicación, para finalmente aclarar la conexión con el dispositivo móvil.

Para dotar al Arduino de conexión Bluetooth, se ha utilizado un módulo HC-04, mostrado en la figura 7.1. Se trata de un adaptador que permite la comunicación serie con un dispositivo de comunicación Bluetooth. El dispositivo en cuestión permite la

configuración como maestro y como esclavo pero en este poryeccto se cconfigurará como esclavo, dejand al dispositivo Android como maestro.

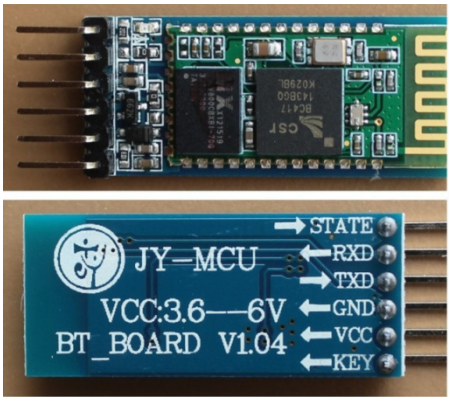


**Figura 7.1.** Módulo Bluetooth HC-04

Cabe decir, que el adaptador HC-04 es un dispositivo que se alimenta a 3.6 [V], aunque está preparado para ser alimentado hasta los 6 [V].

El alcance es muy dependiente de la situación de los transceptores y tienen mucho más alcance cuando están en la línea de visión, al aire libre más que en interior, con obstáculos como paredes y otros materiales. La distancia normal que indican los distintos proveedores para el módulo de baja potencia es de unos 50 metros.

Para la conexión con Arduino se puede hacer de dos formas:

## I. Conexión directa a los pines de TX-RX del Arduino

Únicamente se tiene que conectar el pin Tx (Transmisión) del módulo Bluetooth al Rx (Recepción) del Arduino, y el Rx del dispositivo Bluetooth al Tx del Arduino. Uno de los inconvenientes que presenta este modo, es que para la programación del Arduino, el Bluetooth no debe estar conectado a la placa del Arduino. Otro inconveniente es el montaje, se suprime la posibilidad de utilizar el monitor del IDE del Arduino para la depuración del programa.

## II. Conexión a cualquier pin y uso de la librería *" Software Serial "*

Se pueden utilizar las funciones incluidas en la librería *"Software Serial"*, lo que permite que cualquier pin analógico de la placa de Arduino, funcione como un puerto serie.

En el proyecto se ha optado por trabajar con el segundo método, puesto que se trabaja de forma más ágil. Permitiendo dejar libre la UART para la comunicación y programación de debug.

#### 7.1.1.1. PROGRAMACIÓN

Para manejar el puerto serie se ha de crear una variable, lo que hace que la programación sea sencilla. Para ello, se ha de crear el puerto y luego inicializarlo, de esta manera establecemos la comunicación Bluetooth con el Arduino. Es importante recordar que se trabaja con el método de la librería *"Software Serial"*.

```
#include<SoftwareSerial.h>

int RXArduino = 2;
int TXArduino = 4;

// Creamos el puerto serie, le pasamos los pines que serán
// Tx (donde está conectado en Rx del bluetooth)
// y Rx (donde está conectado el Tx del bluetooth)

Software Serial Blue (RXArduino, TXArduino);

Void setup()  {

Blue.begin(9600); // Y lo inicializamos como cualquier puerto serie }
```

En cualquier caso, sea conectando los pines directamente o utilizando la librería, se leen y se escriben los datos de igual forma. Para la lectura de los datos mediante el módulo Bluetooth empleamos el siguiente comando:

```
char estado = 0 ;
estado =  Blue.read();
```

O de forma general:

```
char  unChar = miPuertoSerie.read();
```

Y para su escritura por el puerto serie empleamos:

```
Serial.println(estado, HEX);
```

O de forma general:

```
miPuertoSerie.print  ("Información a enviar");
```

### 7.1.2. Configuración Básica con Processing

A continuación se va a hacer una breve descripción de lo que es Processing y finalmente se detalla la configuración de éste para establecer comunicación vía Bluetooth. Processing es un lenguaje de programación basado en Java, específico para el desarrollo de gráficos interactivos, teniendo en cuenta el contexto de la programación de dispositivos basados en microcontroladores (MCU), resulta realmente útil para el diseño de interfaces que controlen dispositivos electrónicos.

El lenguaje Processing, al igual que ocurre en numerosos lenguajes de programación parte de una estructura de programa predeterminada, con unos recursos básicos para la programación de gráficos; por otro lado separa en librerías, las funciones menos frecuentes. Por si fuera poco, Processing dispone de distintos modos de programación, lo que implica que pueda trabajar para diferentes plataformas, generando código en diferentes lenguajes. Recientemente se ha desarrollado el modo para Android con lo que programar para este sistema operativo. Desgraciadamente, es necesario el uso de librería para incluir las comunicaciones Bluetooth; la librería Ketai es la que se ha empleado en este proyecto, por ser la más utilizada. El nombre Ketai proviene de la «katakanización» de la palabra portátil, que en japonés se emplea para referenciar al teléfono móvil.

Cabe decir, que este lenguaje de programación se hizo para trabajar con teléfonos inteligentes (Smartphone), que en definitiva son excelentes para controlar dispositivos empleando una interfaz gráfica de usuario (GUI).

#### Instalación de la librería Ketai para Android en Processing

Será necesario disponer previamente del modo de programación Processing para Android. Una vez se ha hecho esto, se procede a la instalación de la librería Ketai; dentro del menú Sketch del entorno de programación (PDE), utilizamos la entrada Añadir biblioteca, esto muestra la ventana Contribución Manager, en la que se muestran las librerías que se quieran agregar. Se aprecia en la figura 7.2.

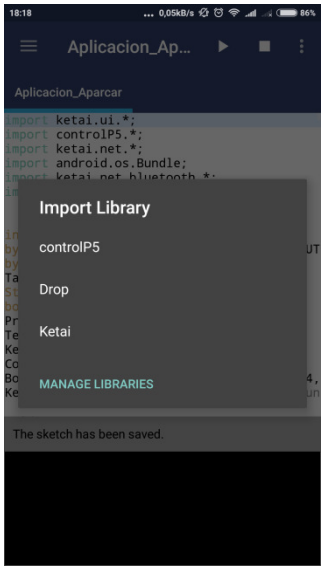


**Figura 7.2** Instalación de la librería Ketai

Para instalarla, se selecciona de la lista, y se pulsa sobre el botón Instalar. Adicionalmente se incluyen actualizaciones posteriores, que se pueden descargar con el botón Descargar.

## Modelo para comunicaciones Bluetooth con la librería Ketai

Uno de los puntos fuertes de Processing, frente a otros lenguajes, consiste en su gran potencia para la creación de los gráficos interactivos con el usuario; esta es una de las razones que la hacen indispensable en el desarrollo de aplicaciones.

El método onBluetoothDataEvent es llamado por la librería Ketai, si se reciben datos desde un dispositivo Bluetooth, de manera que será implementado en todas aquellas aplicaciones que trabajen con Ketai.

A causa de la gestión del ciclo de vida de las tareas en Android, el método para la recepción de datos Bluetooth, la inserción de la librería y de los distintos elementos de desarrollo para Android que se requieren para utilizarla, resulta una estructura de código que seguramente se usa cada vez que se incluyen comunicaciones Bluetooth en un programa Processing. Más adelante, se explicará el método para la detección de dispositivos Bluetooth, dentro de un rango de alcance, y su conexión.

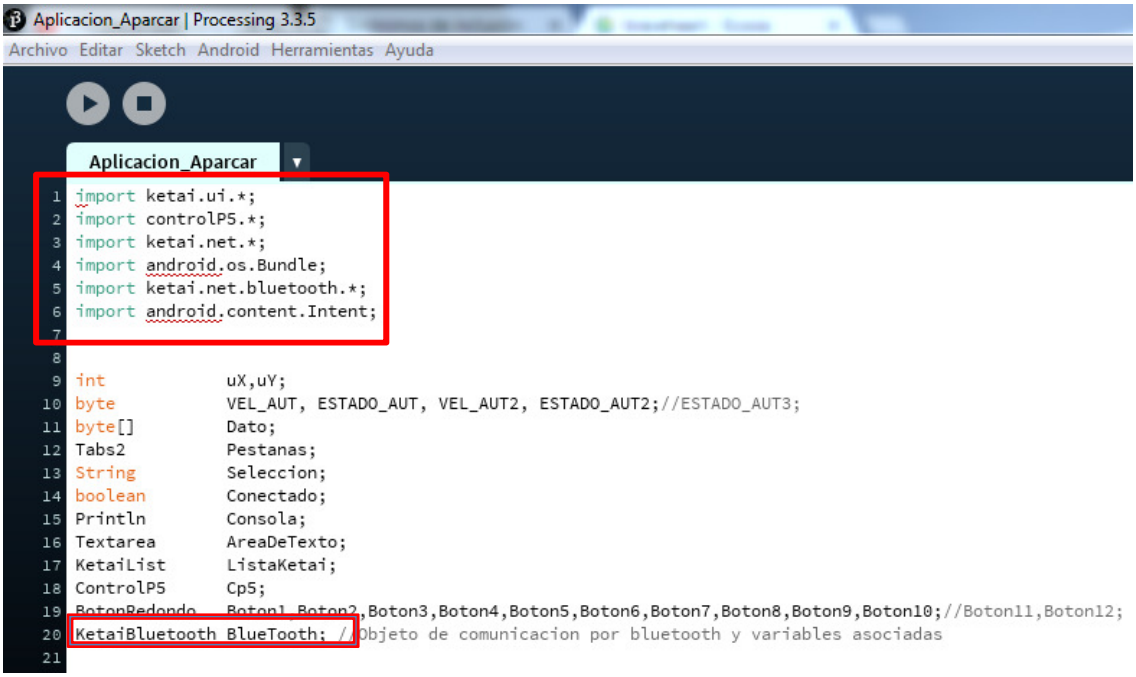


***Figura 7.3.*** *Inclusión de la librería Ketai y creación de la variable Bluetooth*

## Condiciones de un dispositivo Bluetooth con respecto a otro

Desde el punto de vista del dispositivo principal, el resto de dispositivos se pueden encontrar en alguna de estas tres situaciones:

1. Localizables, son aquellos que se encuentran dentro de un rango de cobertura (alcance) y se han configurado de manera que puedan ser localizados.
2. Conectados, una vez se haya establecido la conexión Bluetooth con ellos y se haya introducido la clave correcta necesaria.
3. Emparejados, los dispositivos han establecido una conexión anterior, de forma que son reconocidos y están listos para establecer la conexión de nuevo.

Es necesario recalcar que, si un dispositivo se encuentra ya emparejado con la aplicación, no podrá establecer conexión a menos que se encuentre dentro del rango

de alcance. Por el contrario, el dispositivo puede estar dentro del área de alcance y no ser detectado porque su configuración le impide responder a las peticiones de localización.

## Localización de los dispositivos Bluetooth libres

Por lo que se refiere a la detección con la librería Ketai de dispositivos Bluetooth disponibles, es decir, equipos que se hallan en un rango de alcance y están configurados de manera que puedan ser identificados, es decir, que sean detectables, Ketai utiliza el método discoverDevices; esta función inicia el proceso de búsqueda y captura con el que se va listando cada dispositivo detectable.

Teniendo en cuenta que el proceso dura unos segundos en finalizar la tarea, la librería Ketai dota de otro método, llamado isDiscovering, que devuelve true mientras está activo y false una vez se haya completado.

Dicho lo anterior es razonable pensar que se ha de acceder a esta lista y consultarla, aunque el proceso de búsqueda no haya finalizado todavía. Para ello, se emplea el método getDiscoveredDeviceNames, que devuelve un ArrayList<String>.

```
32  BlueTooth.start();                      //Inicializa la clase de bluetooth
33  Bluetooth.discoverDevices();            // Necesita permisos BLUETOOTH_ADMIN
```

***Figura 7.4.*** *Método discoverDevices.*

```
357 void onKetaiListSelection(KetaiList Klist)    //Obtiene la cadena recien seleccionada
358 {
359  Seleccion = Klist.getSelection();
360  BlueTooth.connectToDeviceByName(Seleccion);  //Efectua la conexion bluetooth al dispositivo seleccionado
361  if(Seleccion!="")
362   Conectado = true;
363  Klist = null;                               //Desaloja la lista, ya no es necesaria
364 }
```

***Figura 7.5.*** *Selección del dispositivo Bluetooth objetivo en la lista creada.*

Respecto a lo anteriormente mencionado hay que añadir la necesidad de la autorización del acceso por parte del usuario; para la declaración de estos permisos se emplea SketchPermisssions del menú de la aplicación Android, éste muestra una lista con los permisos con los nombres de las constantes. Será indispensable seleccionar los dos permisos Bluetooth: BLUETOOTH y BLUETOOTH_ADMIN (Figura 7.7).

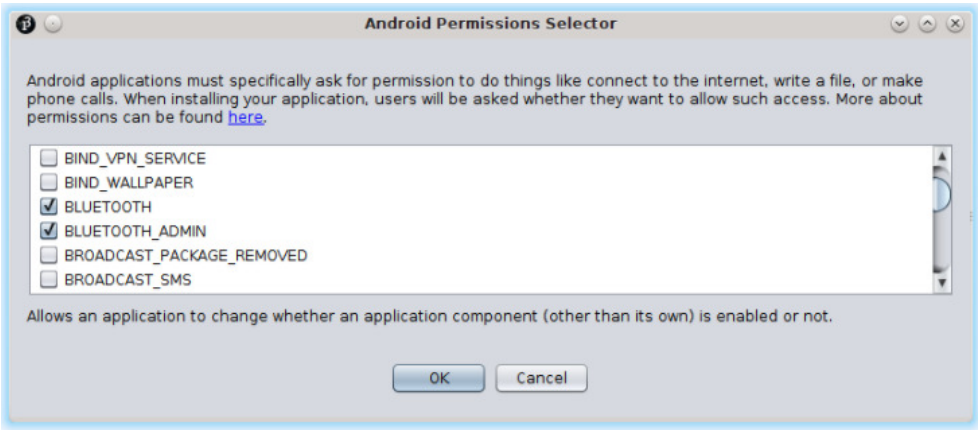


***Figura 7.6.*** *Permisos requeridos por Android para la comunicación Bluetooth.*

## Consultar los dispositivos Bluetooth conectados o emparejados

Como se ha comentado anteriormente, además de los dispositivos detectados, hay una lista con los dispositivos emparejados, a la que se accede con el método *getPairedDeviceNames* y otra con los dispositivos que están conectados que se obtiene con *getConnectedDeviceNames*. Tanto en un caso, como en otro, igual que ocurre con *getDiscoveredDeviceNames*, ambos devuelven un *ArrayList<String>* con los nombres de los dispositivos Bluetooth.

```
121 void AccionesBoton1()    // Boton Buscar el bluetooth
122 {
123  AreaDeTexto.clear();
124  BlueTooth.discoverDevices();  //Obtiene la lista de dispositivos bluetooth emparejados
125 }
```

***Figura 7.7.*** *Búsqueda de los dispositivos Bluetooth*

```
127 void AccionesBoton2()        // Conectar con Bluetooth
128 {
129  if (BlueTooth.getDiscoveredDeviceNames().size() > 0)
130   ListaKetai = new KetaiList(this, BlueTooth.getDiscoveredDeviceNames());
131  else
132   if (BlueTooth.getPairedDeviceNames().size() > 0)
133    ListaKetai = new KetaiList(this, BlueTooth.getPairedDeviceNames());
134 }
```

***Figura 7.8.*** *Creación de una lista en pantalla para que el usuario elija dispositivo Bluetooth*

## Conectar a un dispositivo Bluetooth

En relación con el inicio de las comunicaciones Bluetooth con un dispositivo desde Android, Ketai propone dos métodos: en el primero se toma como argumento la dirección MAC como una cadena de texto (*connectDevice)*; en el segundo se espera el nombre del dispositivo también como cadena de texto (*connectToDeviceByName)*. En el desarrollo de la aplicación se trabaja con el segundo método, puesto que de cara al usuario, resulta más intuitivo emplear cadenas de texto (nombres), en lugar de direcciones (números).

Si el dispositivo con el que se vaya a establecer conexión, no está emparejado, posiblemente requerirá introducir una clave o contraseña. En ese caso, Android muestra la correspondiente notificación y el cuadro de diálogo para su incorporación.

Entrambos procedimientos devuelven un booleano, mostrando si se ha establecido la conexión. Debido a que establecer la conexión no es una tarea inmediata, se infiere no utilizarlos en una estructura *if*, ya que puede resultar confuso; es por esto que se opta por establecer la conexión y seguidamente, dentro de la función *draw*, se comprueba si está activa revisando la lista de dispositivos conectados con *getConnectedDeviceNames.*

```
dispositivos_bluetooth=bluetooth.getConnectedDeviceNames ();

dispositivos_bluetooth=bluetooth.getPairedDeviceNames();
```

## Enviar datos a un dispositivo Bluetooth

Es necesario recalcar que una vez se ha establecido la comunicación con un dispositivo, ya se pueden recibir o enviar datos. Como se ha comentado anteriormente, la librería Ketai permite emplear dos métodos para referirse al dispositivo, o bien, por su dirección MAC (*write*), o bien, por su nombre

(*writeToDeviceName).* Ambos procedimientos esperan dos argumentos, el primero, que es la dirección MAC o el String del nombre; y el segundo, un vector de bytes con los datos enviados. A continuación se muestra un ejemplo de código, el cual pertenece al primer método; mientras que la figura 7.10, se presenta el segundo.

```
bluetooth.write(MAC_dispositivo_destino,mensaje);
```

```
136 void AccionesBoton3()              //Boton ON , enciende automatico
137 {
138
139 if(Conectado)
140   {
141     ESTADO_AUT2 = 1 << 1;
142     Dato[0] = byte(VEL_AUT2 | ESTADO_AUT2);
143     BlueTooth.writeToDeviceName(ListaKetai.getSelection(),Dato);
144
145    }
```

***Figura 7.9.*** *Envío de la variable Dato, que es de tipo byte.*

En el ejemplo anterior se envía un variable de tipo vector de bytes (Dato), pero que solo tiene un byte ocupado, en la posición cero, a través del módulo Bluetooth.

Habría que decir también, que anteriormente se ha consultado la lista de dispositivos conectados y se busca en ella el nombre o la cadena del dispositivo objeto de comunicación. Esta lista, se obtiene con *getConnectedDeviceNames* y la dirección de cada componente es el nombre del dispositivo y la dirección MAC entre paréntesis. A pesar de que el método de la dirección MAC parece un poco más seguro, por trabajar con direcciones en lugar de con el nombre del dispositivo Bluetooth, resulta más intuitivo el uso de nombres y letras.

# 8. Conclusiones y Líneas Futuras

Se ha desarrollado un prototipo cuyo movimiento puede ser gestionado a través del móvil, de manera que pueda ser guiado por el usuario. Se han implementado dos modos de funcionamiento, de manera que en el primero sea el usuario quien dirija el coche teledirigido, mientras que en el segundo circule de forma automática evitando obstáculos.

A través del estudio del arte de los vehículos teledirigidos se ha permitido establecer un nexo de unión con los robots móviles, de forma que no sólo fuera posible el control a distancia con un dispositivo móvil, sino que el propio sistema mecánico actuara de manera "*independiente*", con la ayuda de sensores y un microcontrolador.

Para concluir, se ha incorporado un módulo Bluetooth mediante el dispositivo HC-04, que hace posible la transmisión de datos entre el elemento emisor, que es la aplicación, y el receptor, el microcontrolador. Por medio de este módulo se ha emprendido el aprendizaje de la plataforma Arduino, así como el lenguaje ANDROID.

A lo largo de todo el proyecto se han presentado numerosos problemas debidos a la falta de experiencia en el tema, que han ido solventándose uno a uno hasta dar lugar al resultado que se pretendía. Uno de los mayores inconvenientes fue el desarrollo de la aplicación móvil, puesto que Android está basado en Java y no se había trabajado nunca con este lenguaje; se decidió trabajar con Processing que es más intuitivo.

En cuanto a los objetivos marcados se han cumplido todos, aunque se podría seguir trabajando en algunos aspectos del proyecto, como, por ejemplo:

- Incluir un módulo Wifi, en lugar de Bluetooth, y establecer las comunicaciones mediante él.
- Añadir sensores que doten al prototipo de información sobre su orientación como un giróscopo, una brújula, etc.
- Incorporar un algoritmo de control que permita al vehículo el cálculo de trayectorias.

# 9. Bibliografía


[1] Silva Ortigoza, R., García Sánchez, J. R., Barrientos Sotelo, V. R., Molina Vilchis, M. A., Hernández Guzmán, V. M., Silva Ortigoza, G. (Fecha de consulta: 12 de noviembre de 2017) Una panorámica de los robots móviles. Télématique Disponible en: <http://www.redalyc.org/articulo.oa?id=78460301> ISSN 1856-4194

[2] Cyberneticzoo. (Marzo de 2015). Disponible en: http://cyberneticzoo.com/cyberneticanimals/1977-%E2%80%93-newt-ralph-hollis-american/

[3] Jet Propulsion Laboratory. (Febrero 2, 2015). Disponible en: https://www.jpl.nasa.gov/blog/2015/2/lunar-roving-vehicle-prototype

[4] Marc Raibert. (1996). Disponible en: http://www.ai.mit.edu/projects/leglab/old-leglab/people/mxr.html

[5] A single-Wheel, Gyroscopically stabilized Robot (GYROVER). (1994). Disponible en: http://www.cs.cmu.edu/afs/cs/project/space/www/gyrover/gyrover.html

[6] The Field Robotics Center. (1994). Disponible en: http://www.frc.ri.cmu.edu/projects/danteII/

[7] NASA. (2017). Disponible en: https://www.nasa.gov/specials/pathfinder20/

[8] Robotsvoice. (Noviembre 23, 2017). Disponible en: http://www.robotsvoice.com/honda-p2-and-p3/

[9] *Tom Lauwers; George Kantor; Ralph Hollis (Mayo 2006). "A Dynamically Stable Single-Wheeled Mobile Robot with Inverse Mouse-Ball Drive" . IEEE International Conference on Robotics and Automation. pp. 2884–2889.*

[10] M Kumagai; T Ochiai (octubre de 2008). "Desarrollo de un robot equilibrado en una bola". *Conferencia Internacional de Control* . Seúl, Corea: Automatización y Sistemas: 433-438.

[11] Tomás Arribas. (2014). Disponible en: https://pbs.twimg.com/profile_images/1200346920/PenduloInvertidoEsfera2b_400x400.JPG.

[12] NASA TV. (Noviembre 22, 2017). Disponible en: https://www.nasa.gov/mission_pages/msl/index.html

[13] R. Siegwart, I. R. Nourbakhsh y D. Scaramuzza, (Octubre de 2015). Introduction to Autonomous Mobile Robots, Mit Press Ltd.

[14] City Climber: a new generation of mobile robot with wall-climbing capability Jizhong Xiao; William Morris; Narashiman Chakravarthy; Angel Calle;Proceedings Volume 6230, Unmanned Systems Technology VIII; 62301D (2006); doi: 10.1117/12.666374 Event: Defense and Security Symposium, 2006, Orlando (Kissimmee), Florida, United States.

[15] Robot Platform. (2010). Disponible en: http://www.robotplatform.com/knowledge/Classification_of_Robots/wheel_control_theory.html

[16] Leopoldo Calderón Estévez, Ramón Ceres Ruiz, José Nó Sánchez de León y José Ramón Alique López. (Marzo-Abril. 1985. )"Sensores de distancia en robótica" Revista Robótica. No.12. España.

[17] Leonard J. and Durrant-Whyte, H.F., (1991), "Mobile Robot Localization by Tracking Geometric Beacons." IEEE Transaction on Robotics and Automation, Vol.7, No.3, pp. 376-382.

[18] Google Play (Agosto 2012) Media Remote for Android. Disponible en: https://play.google.com/store/apps/details?id=com.sony.seconddisplay.view&hl= es

[19] Jose A. Gutierrez, Marco Naeve, Eaton Corporation, Ed Callaway, Monique Bourgeois, Motorola Labs, Vinay Milter, Qualcomm Inc. Bob Heile, consultant. IEEE 802. 75.4: (Octubre 2001). A Developing Standard for Low-Power Low-Cost Wireless Personal Area Networks. IEEE Network.

[20] José A. Gutiérrez, Edgar H. Callaway, Raymond L. Barrett. (18 de Junio de 2007). Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4.

[21] Thomas J. Watson Research Center, Yorktown Heights, NY (25 junio 2001). An Overview of the Bluetooth Wireless Technology,

[22] Wi-Fi (802.11b) and Bluetooth: enabling coexistence. Lansford, J. (Sep/Oct 2001). 15 , Issue: 5.

[23] J. Borenstein and L. Feng."UMBmark: (Octubre 22-26, 1995).A benchmarktest for measuring odometryerrors in mobile robots", SPIE Conference on Mobile Robots,Philadelphia.

[24] A. Bucken and S. Thrun, (Abril 1996). "Learning maps for indoor mobile robot navigation", CarnegieMellon University, Pittsburgh.

[25] Lokhande, Omar. ( Febrero 7, 2016). Sensors in Robotics. Disponible en: https://es.slideshare.net/omkara12/sensors-in-robotics

[26] Discusión Tech. (Mayo 16, 2013). Redes y nomenclatura según alance. Disponible en: http://discusiontech.blogspot.com.es/2013/05/local-area-network-lan-la-lan-es-una.html

[27] Taringa. (2014). Arduino Vs Launchpad MP430 Vs Launchpad Tiva. (2014). Disponible en: https://www.taringa.net/posts/hazlo-tu-mismo/17477034/Arduino-Vs-Launchpad-MP430-Vs-Launchpad-Tiva.html

[28] https://www.taringa.net/posts/hazlo-tu-mismo/17477034/Arduino-Vs-Launchpad-MP430-Vs-Launchpad-Tiva.html

[29] Todo lo que debe saber sobre su maquinaria. (2013). Disponible en: https://talleresymaquinaria.wordpress.com/2013/

[30] Foro de carreteres por Asturias. (2015). Disponible en: https://decarreteres.wordpress.com/2015/12/20/condicion-ackermann/

[31] Pablo Luque, Daniel Álvarez y Carlos Vera. (2004). Ingeniería del automóvil. Sistemas y comportamiento dinámico.

[32] Arduino. (Marzo de 2016). Obtenido de www.playground.arduino.cc

[33] Luis Llamas. (2017). Medir distancia con Arduino y sensor de ultrasonidos HC-SR04. Disponible en: https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/

[34] HACKADAY.IO. (2017). Disponible en: https://hackaday.io/project/11936-radar-ultrasonico-arduino

[35] Datasheet del puente H. (2001) Dsiponible en: http://www.alldatasheet.es/datasheet-pdf/pdf/113422/ETC1/SCTX2B.html

[36] Il Bambino, (Diciembre de 2015) “Una introducción a la robótica móvil.” Monografía de la robótica móvil. Disponible en: http://www.aadeca.org/pdf/CP_monografias/monografia_robot_movil.pdf

[37] Rockwell Automation. (2017). Disponible en: https://www.rockwellautomation.com/

# 10. Listado de figuras

# 11. Listado de tablas

# ANEXO 1. Código del microcontrolador

## Bucle Principal

```
#include<SoftwareSerial.h>


int dcha =9; //Pines Analogicos Arduino
int izda = 5;
int front = 10;
int back = 11;
int vel = 255; // Velocidad de los motores (0-255)

//unsigned char prevEstado = 0;
char estado = 0 ;
char on_off;

bool ultimoLadoDcha = false;
bool ultimoLadoIzda = false;

int RXArduino = 2;
int TXArduino = 4;

SoftwareSerial Blue(RXArduino,TXArduino); // RXArduino = 2 TXArduino =
4

const byte Pecho_av=6;    // Pines Digitales Arduino
const byte Pecho_dcha=7;
const byte Pecho_izda=12;
const byte Ptrig_general=8;

void setup()  {
  Serial.begin(9600);          //  inicia  el  puerto  serial  para
comunicacion con el Bluetooth
  Blue.begin(9600);
  Serial.println("Empezamos");
  pinMode(dcha, OUTPUT);
  pinMode(izda, OUTPUT);
  pinMode(front, OUTPUT);
  pinMode(back, OUTPUT);

  pinMode(Pecho_av, INPUT);
  pinMode(Ptrig_general, OUTPUT);
  pinMode(13,OUTPUT);
  pinMode(Pecho_dcha, INPUT);
  pinMode(Pecho_izda, INPUT);


}
```

```
void loop()
{
  if (Blue.available())
  {
    estado =  Blue.read();
    Serial.println(estado, HEX);
   }

   bool aparcar = (estado >> 1) & 0x07 == 0x07;
   if (aparcar){
    //
   }else{
     switch (estado & 0x01)
     {
      case 0:
        Serial.println("Auto");
        Automatico(estado,ultimoLadoDcha);
      break;

      case 1:
        Serial.println("Manual");
        Manual(estado);
      break;
      }
   }

}
```

# Manual

```
void Manual(char estado_2)
{

const char veldi = 255 ; // valor cte para girar derecha o izda
char var5;               // Velocidad

var5 = estado_2 & 0xF0; // Velocidad

char dir = (estado_2 >> 1) & 0x07;


  switch (dir){
    case 1:                        // Avanzar
      analogWrite(front,var5);
      analogWrite(back, 0);
      //analogWrite(dcha, 0);
      //analogWrite(izda, 0);
      break;

     case 2:                      // Girar dcha
      //analogWrite(front,0);
      //analogWrite(back, 0);
      analogWrite(dcha, veldi);
      analogWrite(izda, 0);
      break;
```

```
      case 3:                      // Girar izda
      // analogWrite(front,0);
       //analogWrite(back, 0);
       analogWrite(dcha, 0);
       analogWrite(izda, veldi);
       break;

      case 4:                     // Retroceder
       analogWrite(front,0);
       analogWrite(back, var5);
       //analogWrite(dcha, 0);
       //analogWrite(izda, 0);
       break;

      case 0:                     // STOP
       analogWrite(front,0);
       analogWrite(back, 0);
       analogWrite(dcha, 0);
       analogWrite(izda, 0);
       break;

      case 5:                    // Alinear Ruedas
       analogWrite(dcha, 0);
       analogWrite(izda, 0);
      break;
  }


}
```

# Automático

```
long distancia_front,distancia_izda,distancia_dcha;

void Automatico(char estado_3, bool& ultimaParedDcha)
{
 const char vel_aux = 255 ; // valor cte para girar derecha o izda

 char velocidad;
 char aux_on_off = (estado_3 >> 1) & 0x01;  //0x03

 velocidad = estado_3 & 0xF0; // Velocidad

 if (aux_on_off == 1) on_off = 1;
 else on_off = 2;

 switch (on_off)
 {
    case 1:        // Boton ON,  se mueve sensando distancia

   lee_distancias();
   ultimoLadoDcha    =    (distancia_dcha    <    distancia_izda)    and
(distancia_izda > 30);
   ultimoLadoIzda    =    (distancia_dcha    >    distancia_izda)    and
(distancia_dcha > 30);

    if (distancia_front <= 60)
    {
```

```
      if (ultimoLadoDcha)
        {
         izquierda(255);//Girar izda
         analogWrite(front,100);//Avanzar recto
        }
      else if (ultimoLadoIzda)
        {
         derecha(255);//Girar dcha
         analogWrite(front,100);//Avanzar recto
        }
       else          // Callejón
        {
          para_motores();
          analogWrite(back,50); // Retroceder
          delay(2000);
        }
     }
     else
     {
       recto();
       analogWrite(front,100); // Avanzar recto
      }

   break;

   case 2:          // Boton OFF, detiene los motores no hace nada
        para_motores();
   break;

  }

}
```

# Funciones auxiliares

```
void derecha(byte vel)
{
  analogWrite(izda,0);
  analogWrite(dcha,vel);
}


void izquierda(byte vel)
{
  analogWrite(dcha,0);
  analogWrite(izda,vel);
}


void recto()
{
  analogWrite(dcha,0);
  analogWrite(izda,0);
}
```

```
void para_motores()
{
     analogWrite(dcha, 0);
     analogWrite(izda, 0);
     analogWrite(front, 0);
     analogWrite(back, 0);
}

void lee_distancias()
{
   distancia_front = LeerDistancia(Pecho_av,Ptrig_general);
    Serial.print("Front: ");
    Serial.print(distancia_front);
    Serial.print(" cm     " );


    distancia_izda = LeerDistancia(Pecho_izda,Ptrig_general);
    Serial.print("Left: ");
    Serial.print(distancia_izda);
    Serial.print(" cm     " );


    distancia_dcha = LeerDistancia(Pecho_dcha,Ptrig_general);
    Serial.print("Right: ");
    Serial.print(distancia_dcha);
    Serial.println(" cm" );
}



int LeerDistancia(byte Pecho, byte Ptrig)
{
  unsigned long duracion;
  long distancia;

  digitalWrite(Ptrig, HIGH);   // genera el pulso de trigger por 10us
  delay(0.01);
  digitalWrite(Ptrig, LOW);

  duracion = pulseIn(Pecho, HIGH);              // Lee el tiempo del
Echo
  distancia = (duracion/2) / 29;            // calcula la distancia en
centimetros
  delay(10);

  return distancia;
}
```

# ANEXO 2. Código de la aplicación

```
import ketai.ui.*;
import controlP5.*;
import ketai.net.*;
import android.os.Bundle;
import ketai.net.bluetooth.*;
import android.content.Intent;


int             uX,uY;
byte                                    VEL_AUT,    ESTADO_AUT,    VEL_AUT2,
ESTADO_AUT2;//ESTADO_AUT3;
byte[]          Dato;
Tabs2           Pestanas;
String          Seleccion;
boolean         Conectado;
Println         Consola;
Textarea        AreaDeTexto;
KetaiList       ListaKetai;
ControlP5       Cp5;
BotonRedondo    Boton1,Boton2,Boton3,Boton4,Boton5,Boton6,Boton7,Boton8
,Boton9,Boton10;//Boton11,Boton12;
KetaiBluetooth BlueTooth; //Objeto  de  comunicacion  por  bluetooth  y
variables asociadas

void setup()
{
 orientation(PORTRAIT);
 size(1080,1920);
 uX = width/18;
 uY = height/32;
 Dato = new byte[1];
 Seleccion = "";
 Conectado = false;
 Cp5 = new ControlP5(this);                    //Inicializa la clase de
controlp5
 BlueTooth.start();                            //Inicializa la clase de
bluetooth

 AreaDeTexto = Cp5.addTextarea("Txt").setPosition(uX,4*uY)
                                     .setSize(16*uX,20*uY)
                                     .setFont(createFont("",25))
                                     .setLineHeight(2*uY)
                                     .setColor(color(255,255,255))
                                     .setColorBackground(color(100,100
,100))
                                     .setColorForeground(color(0,255,0
));
 Consola = Cp5.addConsole(AreaDeTexto);

 Boton1 = new
BotonRedondo(3*uX,28*uY,5*uX,color(0,0,255),color(255,0,0),
```

```
                            "Buscar",uY,color(255,255,255));

 Boton2 = new
BotonRedondo(15*uX,28*uY,5*uX,color(0,0,255),color(255,0,0),
                            "Conectar",uY,color(255,255,255));

 Boton3 = new
BotonRedondo(6*uX,10*uY,5*uX,color(0,0,255),color(255,0,0),
                            "ON",uY,color(255,255,255));

 Boton4 = new
BotonRedondo(12*uX,10*uY,5*uX,color(0,0,255),color(255,0,0),
                            "OFF",uY,color(255,255,255));

 Boton5 = new
BotonRedondo(9*uX,5*uY,5*uX,color(0,0,255),color(255,0,0),
                            "Avanzar",uY,color(255,255,255));

 Boton6 = new
BotonRedondo(9*uX,16*uY,5*uX,color(0,0,255),color(255,0,0),
                            "Retroceder",uY,color(255,255,255));

 Boton7 = new
BotonRedondo(3*uX,10*uY,5*uX,color(0,0,255),color(255,0,0),
                            "Izquierda",uY,color(255,255,255));

 Boton8 = new
BotonRedondo(14*uX,10*uY,5*uX,color(0,0,255),color(255,0,0),
                            "Derecha",uY,color(255,255,255));

 Boton9 = new
BotonRedondo(9*uX,28*uY,5*uX,color(0,0,255),color(255,0,0),
                            "STOP",uY,color(255,255,255));

 Boton10 = new
BotonRedondo(9*uX,10*uY,5*uX,color(0,0,255),color(255,0,0),
                            "Recto",uY,color(255,255,255));
   /*
 Boton11 = new BotonRedondo(__*uX,__*uY,5*uX,color(0,0,255),color(255,
0,0),
                            "Bateria",uY,color(255,255,255));

 Boton12 = new BotonRedondo(__*uX,__*uY,5*uX,color(0,0,255),color(255,
0,0),
                            "Cordon",uY,color(255,255,255));
 */

 Cp5.addSlider("Slider1").setPosition(2*uX,20*uY)
                          .setSize(15*uX,4*uY)
                          .setRange(0,255)
                          .setValue(0)
                          .setColorCaptionLabel(0);

// Cp5.getController("Slider1").getCaptionLabel()
                            //   .setFont(createFont("",50));

 Cp5.getController("Slider1").getValueLabel()
                              .hide();

 Cp5.addSlider("Slider2").setPosition(2*uX,20*uY)
                          .setSize(15*uX,4*uY)
```

```
                          .setRange(0,255)
                          .setValue(0)
                          .setColorCaptionLabel(0);

// Cp5.getController("Slider2").getCaptionLabel()
//                             .setFont(createFont("",50));

 Cp5.getController("Slider2").getValueLabel()
                            .hide();



 Pestanas = new
Tabs2("Configuración",uY,color(255,255,255),color(0,0,0),
                      color(0,100,200),color(0,200,255),color(0,255,25
5),

                      "Manual",uY,color(255),color(0),
                      color(0,100,200),color(0,200,255),color(0,255,25
5),

                      "Automatico",uY,color(255),color(0),
                      color(0,100,200),color(0,200,255),color(0,255,25
5),

                      "Aparcar",uY,color(255),color(0),
                      color(0,100,200),color(0,200,255),color(0,255,25
5));

 }
//************************************************
//Acciones Botones
void AccionesBoton1()    // Boton Buscar el bluetooth
{
 AreaDeTexto.clear();
 BlueTooth.discoverDevices();  //Obtiene  la  lista  de  dispositivos
bluetooth emparejados
}

void AccionesBoton2()        // Conectar con Bluetooth
{
 if (BlueTooth.getDiscoveredDeviceNames().size() > 0)
  ListaKetai = new                                    KetaiList(this,
BlueTooth.getDiscoveredDeviceNames());       //Crea una nueva lista de
seleccion  en  pantalla  para  que  el  usuario  elija  un  dispositivo
bluetooth
 else
  if (BlueTooth.getPairedDeviceNames().size() > 0)
   ListaKetai = new KetaiList(this, BlueTooth.getPairedDeviceNames());
}

void AccionesBoton3()              //Boton ON , enciende automatico
{

if(Conectado)
  {
    ESTADO_AUT2 = 1 << 1;
    Dato[0] = byte(VEL_AUT2 | ESTADO_AUT2);
    BlueTooth.writeToDeviceName(ListaKetai.getSelection(),Dato);

   }
```

```
}

void AccionesBoton4()                       // Boton OFF
{
 if(Conectado)
  {
    ESTADO_AUT2 = 2 << 1;
    Dato[0] = byte(VEL_AUT2 | ESTADO_AUT2);
    BlueTooth.writeToDeviceName(ListaKetai.getSelection(),Dato);

   }

}

void AccionesBoton5()                       // Boton AVANZAR
{
 if(Conectado)
  {
    ESTADO_AUT = 1 << 1;
    Dato[0] = byte(VEL_AUT | ESTADO_AUT | 0x1);
    BlueTooth.writeToDeviceName(ListaKetai.getSelection(),Dato);

   }

}

void AccionesBoton6()                       // Boton RETROCEDER
{
 if(Conectado)
  {
    ESTADO_AUT = 4 << 1;
    Dato[0] = byte(VEL_AUT | ESTADO_AUT | 0x1);
    BlueTooth.writeToDeviceName(ListaKetai.getSelection(),Dato);

   }

}

void AccionesBoton7()                       // Boton IZQUIERDA
{
 if(Conectado)
  {
    ESTADO_AUT = 3 << 1;
    Dato[0] = byte(VEL_AUT | ESTADO_AUT | 0x1);
    BlueTooth.writeToDeviceName(ListaKetai.getSelection(),Dato);

   }

}

void AccionesBoton8()                       // Boton DERECHA
{
 if(Conectado)
  {
    ESTADO_AUT = 2 << 1;
    Dato[0] = byte(VEL_AUT | ESTADO_AUT | 0x1);
    BlueTooth.writeToDeviceName(ListaKetai.getSelection(),Dato);

   }
```

```
}

void AccionesBoton9()                       // Boton STOP
{
 if(Conectado)
  {
    ESTADO_AUT = 0 << 1;
    Dato[0] = byte(VEL_AUT | ESTADO_AUT | 0x1);
    BlueTooth.writeToDeviceName(ListaKetai.getSelection(),Dato);

   }

}

void AccionesBoton10()                       // Boton Alinear Ruedas
{
 if(Conectado)
  {
    ESTADO_AUT = 5 << 1;
    Dato[0] = byte(VEL_AUT | ESTADO_AUT | 0x1);
    BlueTooth.writeToDeviceName(ListaKetai.getSelection(),Dato);

   }

}
/*

void AccionesBoton11()                       // Boton APARCAR BATERÍA DC
HA
{
 if(Conectado)
  {
    ESTADO_AUT3 = 0 << 4;
    Dato[0] = byte(0x0 | ESTADO_AUT3 | 0xE);
    BlueTooth.writeToDeviceName(ListaKetai.getSelection(),Dato);

   }

}

void AccionesBoton12()                       // Boton APARCAR CORDÓN IZD
A
{
 if(Conectado)
  {
    ESTADO_AUT3 = 1 << 4;
    Dato[0] = byte(0x0 | ESTADO_AUT3 | 0xE);
    BlueTooth.writeToDeviceName(ListaKetai.getSelection(),Dato);

   }

}
*/
//**********************************************
//Objetos a Mostrar u Ocultar en las Pestañas
void ShowObjsTab1()
{
 Boton1.Show();
 Boton2.Show();
 AreaDeTexto.show();
```

```
}

void HideObjsTab1()
{
 Boton1.Hide();
 Boton2.Hide();
 AreaDeTexto.hide();
}

void ShowObjsTab2()
{
 Cp5.getController("Slider1").show();
 Boton5.Show();
 Boton6.Show();
 Boton7.Show();
 Boton8.Show();
 Boton9.Show();
 Boton10.Show();

}

void HideObjsTab2()
{
 Cp5.getController("Slider1").hide();
 Boton5.Hide();
 Boton6.Hide();
 Boton7.Hide();
 Boton8.Hide();
 Boton9.Hide();
 Boton10.Hide();

}

void ShowObjsTab3()
{
 Cp5.getController("Slider2").show();
 Boton3.Show();
 Boton4.Show();
}

void HideObjsTab3()
{
 Cp5.getController("Slider2").hide();
 Boton3.Hide();
 Boton4.Hide();
}

void ShowObjsTab4()
{
// Boton11.Show();
 //Boton12.Show();
}

void HideObjsTab4()
{
 //Boton10.Hide();
 //Boton11.Hide();
}
//************************************************
void Slider1(float ValorRango)              //ValorRango es un real
{
```

```
 if(Conectado)
 {
   int valorRangoInt = (int) ValorRango;
   VEL_AUT = (byte)( valorRangoInt & 0xF0);
 }
}

void Slider2(float ValorRango)              //ValorRango es un real
{
 if(Conectado)
 {
   int valorRangoInt = (int) (ValorRango);
   VEL_AUT2 = (byte)( valorRangoInt & 0xF0);

 }
}
//*********************************************

void onCreate(Bundle savedInstanceState)
{
 super.onCreate(savedInstanceState);
 BlueTooth = new KetaiBluetooth(this);
}

void onActivityResult(int requestCode, int resultCode, Intent data)
{
 BlueTooth.onActivityResult(requestCode, resultCode, data);
}

void  onKetaiListSelection(KetaiList  Klist)       //Obtiene  la  cadena
recien seleccionada
{
 Seleccion = Klist.getSelection();
 BlueTooth.connectToDeviceByName(Seleccion);  //Efectua   la   conexion
bluetooth al dispositivo seleccionado
 if(Seleccion!="")
  Conectado = true;
 Klist = null;                                  //Desaloja la lista, ya
no es necesaria
}

void onBluetoothDataEvent(String who, byte[] data)
{

}

public class BotonRedondo
{
 int     TamanoTextoBoton;
 int     CentroX,CentroY,Diametro;
 color   ColorTextoBoton;
 color   ColorBoton,ColorBotonPressed;
 String  TextoBoton;
 boolean vDragged,vReleased,Activo;

 BotonRedondo(int   X,int   Y,int   D,color   Cb,color   Cp,String   T,int
Tt,color Ct)
 {
  CentroX = X;
  CentroY = Y;
  Diametro = D;
```

```
  ColorBoton = Cb;
  ColorBotonPressed = Cp;
  TextoBoton = T;
  TamanoTextoBoton = Tt;
  ColorTextoBoton = Ct;
  vDragged = false;
  vReleased = false;
  Activo = true;
  smooth();
  noStroke();
  fill(ColorBoton);
  ellipse(CentroX,CentroY,Diametro,Diametro);
  textSize(TamanoTextoBoton);
  fill(ColorTextoBoton);
  textAlign(CENTER,CENTER);
  text(TextoBoton,CentroX,CentroY);
 }

 void Hide()
 {
  Activo = false;
  fill(g.backgroundColor);
  stroke(g.backgroundColor);
  rect(CentroX-(Diametro/2)-1,CentroY-(Diametro/2)-
1,Diametro,Diametro);
  noStroke();
 }

 void Show()
 {
  Activo = true;
  smooth();
  noStroke();
  fill(ColorBoton);
  ellipse(CentroX,CentroY,Diametro,Diametro);
  textSize(TamanoTextoBoton);
  fill(ColorTextoBoton);
  textAlign(CENTER,CENTER);
  text(TextoBoton,CentroX,CentroY);

 }

 void Pressed()
 {
  vReleased = false;
  if(dist(CentroX,CentroY,mouseX,mouseY) < Diametro/2)
  {
   smooth();
   noStroke();
   fill(ColorBotonPressed);
   ellipse(CentroX,CentroY,Diametro,Diametro);
   textSize(TamanoTextoBoton);
   fill(ColorTextoBoton);
   textAlign(CENTER,CENTER);
   text(TextoBoton,CentroX,CentroY);
  }
 }

 void Dragged()
 {
  if(dist(CentroX,CentroY,mouseX,mouseY) < Diametro/2)
```

```
    vDragged = true;
  }

  void Released()
  {
   if(dist(CentroX,CentroY,mouseX,mouseY) < Diametro/2)
   {
    smooth();
    noStroke();
    fill(ColorBoton);
    ellipse(CentroX,CentroY,Diametro,Diametro);
    textSize(TamanoTextoBoton);
    fill(ColorTextoBoton);
    textAlign(CENTER,CENTER);
    text(TextoBoton,CentroX,CentroY);
    vReleased = true;
   }

   if(vDragged)
   {
    vDragged = false;
    smooth();
    noStroke();
    fill(ColorBoton);
    ellipse(CentroX,CentroY,Diametro,Diametro);
    textSize(TamanoTextoBoton);
    fill(ColorTextoBoton);
    textAlign(CENTER,CENTER);
    text(TextoBoton,CentroX,CentroY);
    vReleased = true;
   }

  }

}
void mouseDragged()
{
 Pestanas.Dragged();
//************************************************
//Botones
 if(Boton1.Activo)
  Boton1.Dragged();

 if(Boton2.Activo)
  Boton2.Dragged();

  if(Boton3.Activo)
  Boton3.Dragged();

  if(Boton4.Activo)
  Boton4.Dragged();

  if(Boton5.Activo)
  Boton5.Dragged();

  if(Boton6.Activo)
  Boton6.Dragged();

  if(Boton7.Activo)
  Boton7.Dragged();
```

```
  if(Boton8.Activo)
  Boton8.Dragged();

 if(Boton9.Activo)
  Boton8.Dragged();

  if(Boton10.Activo)
  Boton8.Dragged();

  /* if(Boton11.Activo)
  Boton8.Dragged();

  if(Boton12.Activo)
  Boton8.Dragged();*/

}
void mousePressed()
{
 Pestanas.Pressed();
//************************************************
//Botones
  if(Boton1.Activo)
  Boton1.Pressed();

  if(Boton2.Activo)
  Boton2.Pressed();

  if(Boton3.Activo)
  Boton3.Pressed();

  if(Boton4.Activo)
  Boton4.Pressed();

  if(Boton5.Activo)
  Boton5.Pressed();

  if(Boton6.Activo)
  Boton6.Pressed();

  if(Boton7.Activo)
  Boton7.Pressed();

  if(Boton8.Activo)
  Boton8.Pressed();

  if(Boton9.Activo)
  Boton8.Pressed();

  if(Boton10.Activo)
  Boton8.Pressed();

  /*if(Boton9.Activo)
  Boton8.Pressed();

  if(Boton10.Activo)
  Boton8.Pressed();*/

}
void mouseReleased()
{
 Pestanas.Released();
```

```
//***********************************************
//Botones
 if(Boton1.Activo)
 {
  Boton1.Released();
  if(Boton1.vReleased)
   AccionesBoton1();
 }

 if(Boton2.Activo)
 {
  Boton2.Released();
  if(Boton2.vReleased)
   AccionesBoton2();
 }

 if(Boton3.Activo)
 {
  Boton3.Released();
  if(Boton3.vReleased)
   AccionesBoton3();
  }

  if(Boton4.Activo)
 {
  Boton4.Released();
  if(Boton4.vReleased)
   AccionesBoton4();
  }

  if(Boton5.Activo)
 {
  Boton5.Released();
  if(Boton5.vReleased)
   AccionesBoton5();
  }

  if(Boton6.Activo)
 {
  Boton6.Released();
  if(Boton6.vReleased)
   AccionesBoton6();
  }

  if(Boton7.Activo)
 {
  Boton7.Released();
  if(Boton7.vReleased)
   AccionesBoton7();
  }

  if(Boton8.Activo)
 {
  Boton8.Released();
  if(Boton8.vReleased)
   AccionesBoton8();
  }

  if(Boton9.Activo)
 {
  Boton9.Released();
```

```
  if(Boton9.vReleased)
   AccionesBoton9();
  }

  if(Boton10.Activo)
 {
  Boton10.Released();
  if(Boton10.vReleased)
   AccionesBoton10();
 }

/*  if(Boton11.Activo)
 {
  Boton9.Released();
  if(Boton9.vReleased)
   AccionesBoton9();
  }

  if(Boton12.Activo)
 {
  Boton10.Released();
  if(Boton10.vReleased)
   AccionesBoton10();
  }*/

}
public class Tabs2
{
 int
TamanoTextoTab1,TamanoTextoTab2,TamanoTextoTab3,TamanoTextoTab4;
 color  ColorTab1Activa,ColorTab1Inactiva,ColorTab1Pressed;
 color  ColorTab2Activa,ColorTab2Inactiva,ColorTab2Pressed;
 color  ColorTab3Activa,ColorTab3Inactiva,ColorTab3Pressed;
 color  ColorTab4Activa,ColorTab4Inactiva,ColorTab4Pressed;
 color  ColorTextoActivoTab1,ColorTextoInactivoTab1;
 color  ColorTextoActivoTab2,ColorTextoInactivoTab2;
 color  ColorTextoActivoTab3,ColorTextoInactivoTab3;
 color  ColorTextoActivoTab4,ColorTextoInactivoTab4;
 String  TextoTab1,TextoTab2,TextoTab3,TextoTab4;
 boolean
Tab1,Tab2,Tab1Dragged,Tab2Dragged,Tab3,Tab3Dragged,Tab4,Tab4Dragged;
//*********************************************
 Tabs2(String  Tt1,int  Ttt1,color  Ctat1,color  Ctit1,color  Cta1,color
Cti1,color Ctp1,
       String  Tt2,int  Ttt2,color  Ctat2,color  Ctit2,color  Cta2,color
Cti2,color Ctp2,
       String  Tt3,int  Ttt3,color  Ctat3,color  Ctit3,color  Cta3,color
Cti3,color Ctp3,
       String  Tt4,int  Ttt4,color  Ctat4,color  Ctit4,color  Cta4,color
Cti4,color Ctp4)
 {
  HideObjsTab4();
  HideObjsTab3();
  HideObjsTab2();
  ShowObjsTab1();
  smooth();
  noStroke();

  TextoTab1 = Tt1;
  TamanoTextoTab1 = Ttt1;
  ColorTextoActivoTab1 = Ctat1;
```

```
 ColorTextoInactivoTab1 = Ctit1;
 ColorTab1Activa = Cta1;
 ColorTab1Inactiva = Cti1;
 ColorTab1Pressed = Ctp1;

 TextoTab2 = Tt2;
 TamanoTextoTab2 = Ttt2;
 ColorTextoActivoTab2 = Ctat2;
 ColorTextoInactivoTab2 = Ctit2;
 ColorTab2Activa = Cta2;
 ColorTab2Inactiva = Cti2;
 ColorTab2Pressed = Ctp2;

 TextoTab3 = Tt3;
 TamanoTextoTab3 = Ttt3;
 ColorTextoActivoTab3 = Ctat3;
 ColorTextoInactivoTab3 = Ctit3;
 ColorTab3Activa = Cta3;
 ColorTab3Inactiva = Cti3;
 ColorTab3Pressed = Ctp3;

 TextoTab4 = Tt4;
 TamanoTextoTab4 = Ttt4;
 ColorTextoActivoTab4 = Ctat4;
 ColorTextoInactivoTab4 = Ctit4;
 ColorTab4Activa = Cta4;
 ColorTab4Inactiva = Cti4;
 ColorTab4Pressed = Ctp4;

 Tab1 = true;
 Tab2 = false;
 Tab3 = false;
 Tab4 = false;
 Tab1Dragged = false;
 Tab2Dragged = false;
 Tab3Dragged = false;
 Tab4Dragged = false;
 DibujarTabs();
}
//************************************************
 void Pressed()
 {
  if(mouseX>0 && mouseX<(width/4) &&
     mouseY>0 && mouseY<2*uY)
  {
   Tab1 = true;
   Tab2 = false;
   Tab3 = false;
   Tab4 = false;
   fill(ColorTab1Pressed);
   rect(0,0,(width/4),2*uY);
   textSize(uY);
   fill(ColorTextoActivoTab1);
   textAlign(CENTER,CENTER);
   text(TextoTab1,int((18*uX/8)),uY);
  }
 if(mouseX>(width/4) && mouseX<(width/2) &&
    mouseY>0 && mouseY<2*uY)
 {
  Tab1 = false;
  Tab2 = true;
```

```
  Tab3 = false;
  Tab4 = false;
  fill(ColorTab2Pressed);
  rect((width/4),0,(width/4),2*uY);
//Posible concentracion de fallos
  textSize(uY);
  fill(ColorTextoActivoTab2);
  textAlign(CENTER,CENTER);
  text(TextoTab2,int((27*uX/4)),uY);
  }
 if(mouseX>(width/2) && mouseX<(3*width/4) &&
    mouseY>0 && mouseY<2*uY)
 {
  Tab1 = false;
  Tab2 = false;
  Tab3 = true;
  Tab4 = false;
  fill(ColorTab3Pressed);
  rect((width/2),0,(width/4),2*uY);
  textSize(uY);
  fill(ColorTextoActivoTab3);
  textAlign(CENTER,CENTER);
  text(TextoTab3,int((45*uX/4)),uY);
  }
 if(mouseX >(3*width/4) && mouseX < width &&
    mouseY>0 && mouseY<2*uY)
 {
  Tab1 = false;
  Tab2 = false;
  Tab3 = false;
  Tab4 = true;
  fill(ColorTab4Pressed);
  rect((3*width/4),0,(width/4),2*uY);
  textSize(uY);
  fill(ColorTextoActivoTab4);
  textAlign(CENTER,CENTER);
  text(TextoTab4,int((63*uX/4)),uY);
  }
 }

//************************************************
 void Dragged()
 {
  if(mouseX>0 && mouseX<(width/4) &&
     mouseY>0 && mouseY<2*uY)
   Tab1Dragged = true;

  if(mouseX>(width/4) && mouseX<(width/2) &&
    mouseY>0 && mouseY<2*uY)
   Tab2Dragged = true;

  if(mouseX>(width/2) && mouseX<(3*width/4) &&
    mouseY>0 && mouseY<2*uY)
   Tab3Dragged = true;

  if(mouseX >(3*width/4) && mouseX < width &&
    mouseY>0 && mouseY<2*uY)
   Tab4Dragged = true;
 }
//************************************************
 void Released()
```

```
{
 if(mouseX>0 && mouseX<(width/4) &&
    mouseY>0 && mouseY<2*uY)
 {
  HideObjsTab2();
  HideObjsTab3();
  HideObjsTab4();
  fill(g.backgroundColor);
  rect(0,2*uY+1,width,30*uY);
  ShowObjsTab1();
  DibujarTabs();
 }
 if(mouseX>(width/4) && mouseX<(width/2) &&
   mouseY>0 && mouseY<2*uY)
 {
  HideObjsTab1();
  HideObjsTab3();
  HideObjsTab4();
  fill(g.backgroundColor);
  rect(0,2*uY+1,width,30*uY);
  ShowObjsTab2();
  DibujarTabs();
 }
 if(mouseX>(width/2) && mouseX<(3*width/4) &&
   mouseY>0 && mouseY<2*uY)
 {
  HideObjsTab1();
  HideObjsTab2();
  HideObjsTab4();
  fill(g.backgroundColor);
  rect(0,2*uY+1,width,30*uY);
  ShowObjsTab3();
  DibujarTabs();
 }
 if(mouseX >(3*width/4) && mouseX < width &&
   mouseY>0 && mouseY<2*uY)
 {
  HideObjsTab1();
  HideObjsTab2();
  HideObjsTab3();
  fill(g.backgroundColor);
  rect(0,2*uY+1,width,30*uY);
  ShowObjsTab4();
  DibujarTabs();
 }
if(Tab1Dragged)
{
 Tab1Dragged = false;
 HideObjsTab2();
 HideObjsTab3();
 HideObjsTab4();
 fill(g.backgroundColor);
 rect(0,2*uY+1,width,30*uY);
 ShowObjsTab1();
 DibujarTabs();
 }
 if(Tab2Dragged)
 {
  Tab2Dragged = false;
  HideObjsTab1();
  HideObjsTab3();
```

```
    HideObjsTab4();
    fill(g.backgroundColor);
    rect(0,2*uY+1,width,30*uY);
    ShowObjsTab2();
    DibujarTabs();
   }
   if(Tab3Dragged)
   {
    Tab3Dragged = false;
    HideObjsTab1();
    HideObjsTab2();
    HideObjsTab4();
    fill(g.backgroundColor);
    rect(0,2*uY+1,width,30*uY);
    ShowObjsTab3();
    DibujarTabs();
   }
   if(Tab4Dragged)
   {
    Tab4Dragged = false;
    HideObjsTab1();
    HideObjsTab2();
    HideObjsTab3();
    fill(g.backgroundColor);
    rect(0,2*uY+1,width,30*uY);
    ShowObjsTab4();
    DibujarTabs();
   }
  }
 //*********************************************
  void DibujarTabs()
  {
   if(Tab1)
   {
    noStroke();
    fill(ColorTab1Activa);
    rect(0,0,width/4,2*uY);
    fill(ColorTab2Inactiva);
    rect(width/4,0,width/4,2*uY);
    fill(ColorTab3Inactiva);
    rect((width/2),0,width/4,2*uY);
    fill(ColorTab4Inactiva);
    rect((3*width/4),0,width/4,2*uY);
    textSize(uY);
    fill(ColorTextoActivoTab1);
    textAlign(CENTER,CENTER);
    text(TextoTab1,int((18*uX/8)),uY);
    fill(ColorTextoInactivoTab2);
    textAlign(CENTER,CENTER);
    text(TextoTab2,int((27*uX/4)),uY);
    fill(ColorTextoInactivoTab3);
    textAlign(CENTER,CENTER);
    text(TextoTab3,int((45*uX/4)),uY);
    fill(ColorTextoInactivoTab4);
    textAlign(CENTER,CENTER);
    text(TextoTab4,int((63*uX/4)),uY);
   }

   if(Tab2)
   {
    noStroke();
```

```
 fill(ColorTab1Inactiva);
 rect(0,0,width/4,2*uY);
 fill(ColorTab2Activa);
 rect(width/4,0,width/4,2*uY);
 fill(ColorTab3Inactiva);
 rect((width/2),0,width/4,2*uY);
 fill(ColorTab4Inactiva);
 rect((3*width/4),0,width/4,2*uY);
 textSize(uY);
 fill(ColorTextoInactivoTab1);
 textAlign(CENTER,CENTER);
 text(TextoTab1,int((18*uX/8)),uY);
 fill(ColorTextoActivoTab2);
 textAlign(CENTER,CENTER);
 text(TextoTab2,int((27*uX/4)),uY);
 fill(ColorTextoInactivoTab3);
 textAlign(CENTER,CENTER);
 text(TextoTab3,int((45*uX/4)),uY);
 fill(ColorTextoInactivoTab4);
 textAlign(CENTER,CENTER);
 text(TextoTab4,int((63*uX/4)),uY);
}
if(Tab3)
{
 noStroke();
 fill(ColorTab1Inactiva);
 rect(0,0,width/4,2*uY);
 fill(ColorTab2Inactiva);
 rect(width/4,0,width/4,2*uY);
 fill(ColorTab3Activa);
 rect((width/2),0,width/4,2*uY);
 fill(ColorTab4Inactiva);
 rect((3*width/4),0,width/4,2*uY);
 textSize(uY);
 fill(ColorTextoInactivoTab1);
 textAlign(CENTER,CENTER);
 text(TextoTab1,int((18*uX/8)),uY);
 fill(ColorTextoInactivoTab2);
 textAlign(CENTER,CENTER);
 text(TextoTab2,int((27*uX/4)),uY);
 fill(ColorTextoActivoTab3);
 textAlign(CENTER,CENTER);
 text(TextoTab3,int((45*uX/4)),uY);
 fill(ColorTextoInactivoTab4);
 textAlign(CENTER,CENTER);
 text(TextoTab4,int((63*uX/4)),uY);
}
  if(Tab4)
{
 noStroke();
 fill(ColorTab1Inactiva);
 rect(0,0,width/4,2*uY);
 fill(ColorTab2Inactiva);
 rect(width/4,0,width/4,2*uY);
 fill(ColorTab3Inactiva);
 rect((width/2),0,width/4,2*uY);
 fill(ColorTab4Activa);
 rect((3*width/4),0,width/4,2*uY);
 textSize(uY);
 fill(ColorTextoInactivoTab1);
 textAlign(CENTER,CENTER);
```

```
    text(TextoTab1,int((18*uX/8)),uY);
    fill(ColorTextoInactivoTab2);
    textAlign(CENTER,CENTER);
    text(TextoTab2,int((27*uX/4)),uY);
    fill(ColorTextoInactivoTab3);
    textAlign(CENTER,CENTER);
    text(TextoTab3,int((45*uX/4)),uY);
    fill(ColorTextoActivoTab4);
    textAlign(CENTER,CENTER);
    text(TextoTab4,int((63*uX/4)),uY);
   }
  }
}
void draw()
{
}
```