Universidad Politécnica de Cartagena

industriales
etsii UPCT

DISEÑO Y CONSTRUCCIÓN DE UN COCHE TELEDIRIGIDO BASADO EN LA PLATAFORMA ARDUINO

Titulación: Electrónica Industrial

Intensificación: Automática

Alumno/a: Jennifer Rodríguez Muñoz

Director/a/s: Juan Suardíaz Muro

Cartagena, 10 de Febrero de 2015

ÍNDICE

CAPÍTULO 1

INTRODUCCIÓN

En este capítulo primero se presentarán los objetivos del proyecto así como las diferentes fases que se han seguido para su desarrollo e implementación.

Haciendo referencia a los objetivos, se describirá con detalle todo aquello que se pretende desarrollar justificando, de igual manera, las razones de su tipo de elaboración así como de aspectos más específicos.

Respecto a las fases, se describirá en cada una de ellas respectivamente las diferentes visiones y aspectos a estudiar y desarrollar en dicho proyecto a presentar. Desde el primer capítulo introductorio hasta el último sobre los posibles trabajos futuros y conclusiones llegadas, se darán a conocer los diferentes objetivos a alcanzar en cada uno de ellos de forma que sirva tanto al autor, en la propia realización del proyecto, como al lector de manera de guía práctica a seguir de todo el proceso de realización.

1.1. OBJETIVOS DEL PROYECTO

El presente proyecto se desarrollará en torno a un coche dirigido por radiocontrol con mando, enfocándonos en la comunicación inalámbrica programada la cual gestionará los movimientos del vehículo.

El objetivo del proyecto será el desarrollo de la interfaz entre vehículo y mando de manera que nos permita la libertad de control y ajuste de precisión y respuesta del movimiento del mismo; dicha interfaz y electrónica de acondicionamiento será la que se aborde a lo largo del presente proyecto. Se hará una propuesta de diseño que implementará un microcontrolador de bajo coste que dote de inteligencia y de posibilidad de control más avanzado al sistema.

1.2. ESTRUCTURA DE LA MEMORIA DEL PROYECTO

La estructuración de los objetivos de este proyecto, partiendo del capítulo introductorio en el que hablamos de los objetivos principales de manera general, ha sido organizada en diferentes secciones;

Capítulo 2. Estado del arte del mundo del coche teledirigido

En este segundo capítulo haremos referencia a los aspectos más significativos que rodean al mundo de los coches teledirigidos de manera que conozcamos su historia y desarrollo y más concretamente en España.

También desarrollaremos los aspectos técnicos tales como el funcionamiento de manera tanto analógica como digital, los distintos tipos de coches teledirigidos así como de las partes y componentes por los que está formado.

Hablaremos también no sólo del coche sino del mando que lo controla, de su comunicación, tipos y funcionamiento del mismo. También tendrán mención y desarrollo el apartado acerca la comercialización sobre todo el mundo del coche teledirigido como también se hará sobre los diferentes fabricantes.

Capítulo 3. Análisis y arquitectura hardware de control

En este capítulo profundizaremos en el diseño de la arquitectura hardware encargada del control del sistema. Empezaremos por una pequeña introducción de conceptos básicos siguiendo con el desarrollo del microcontrolador así como sus diferentes características tales como su tipo de arquitectura interna, procesador, memoria, puertos y diferentes tipos de familias.

Se realizará el estudio detallado sobre cada uno de los bloques presentados partiendo de las generalidades que los conforman. Se hablará también sobre la plataforma interfaz elegida y se hará alusión a las razones de dicha elección frente a otras opciones disponibles.

Abordaremos pues una sección perteneciente a la conexión inalámbrica, esto es, a la comunicación entre las partes a desarrollar a partir del prototipo elegido anteriormente en el que trabajamos este proyecto.

Capítulo 4. Implementación de la arquitectura hardware

En este apartado se presentarán los componentes que intervienen directamente en la construcción del proyecto. Se realizará un estudio sobre las características de dichos módulos, sus principios físicos y demostraciones teóricas de los mismos.

Se presentarán también los métodos empleados para su realización así como el software utilizado a lo largo de todo el diseño del prototipo.

Capítulo 5. Implementación del software

Este capítulo se centrará en la explicación del funcionamiento y análisis del software de control utilizado por el microcontrolador incorporado en la plataforma software elegida.

Se explicará detalladamente la implementación del software de control para la comunicación de ambos módulos, emisor y receptor.

También se detallarán los pasos a seguir para la asignación de características de cada módulo por separado.

Capítulo 6. Presupuesto final, conclusiones y trabajos futuros

En este último capítulo se expondrán las conclusiones que se obtienen tras la elaboración del presente proyecto. Se explicarán los problemas más relevantes que han ido surgiendo en la elaboración del prototipo y la forma de solventarlos.

A continuación se realizará una valoración a título personal sobre lo que supone al autor el desarrollo y elaboración del proyecto.

Por último se trazarán aquellas posibles líneas en las que se puede orientar la elaboración de futuros trabajos y proyectos relacionados con el mundo del radiocontrol y en particular con el proyecto expuesto en esta memoria.

Además se incluirá un presupuesto donde vendrán detallados la totalidad de componentes incluidos en el sistema desarrollado con su correspondiente coste.

CAPÍTULO 2

ESTADO DEL ARTE DEL MUNDO DEL COCHE TELEDIRIGIDO

En este segundo capítulo se tratará sobre el estado del arte del mundo del coche teledirigido, donde se realizará un estudio sobre los aspectos más importantes que engloban a éste.

Para comenzar, se hará un repaso histórico del mismo a nivel internacional, para centrarse en la evolución que ha sufrido en España. También se analizarán aspectos técnicos de funcionamiento, diferenciando entre naturaleza analógica y digital, además de profundizar en los principios físicos de mayor importancia.

Posteriormente, se estudiarán los principales elementos que intervienen, como son los vehículos y sus componentes, los mandos o controladores analógicos y digitales. Son abordados también los diferentes fabricantes y comerciales a nivel internacional e internacional relacionados con este mundo del control teledirigido y vehículos radiocontrolados a distancia.

2.1. INTRODUCCIÓN

El automodelismo o radiocontrol de vehículos es una afición muy extendida a día de hoy pero cuyos comienzos fueron relativamente tardíos.

Básicamente se trata del control de un automóvil a escala en tiempo real, equipado éste con un motor eléctrico que responde a las señales de un mando del que depende el movimiento del coche. Estas señales pueden venir dadas tanto por medio alámbrico como inalámbrico aunque ambas permitan igualmente el control a distancia respecto al emisor.

El primer modelo radiocontrolado data de la década de 1960 y pertenecía a la categoría de pista. El primer todoterreno es de 1977.

Últimamente y debido a los avances en el campo de la electrónica, se están extendiendo bastante los microautomóviles eléctricos.

Existen bastantes federaciones tanto nacionales como internacionales que regulan y reglamentan el uso y concurso de este tipo de sector del automovilismo de manera que haya competiciones de muy diferentes categorías dependiendo del tipo, tamaño y características del vehículo de radiocontrol utilizado.

2.2. HISTORIA. INICIOS

No hay una fecha exacta de comienzo del Automodelismo, sin embargo, poco después de la segunda guerra mundial, aparecen coches cuyo movimiento se asemeja a una versión de los aviones de vuelo circular, pero sobre tierra.

El automodelismo radio-controlado nace aprovechando las radios de control digital proporcional y los motores de metanol, desarrollados en aeromodelismo.

El primer coche que tomó forma tal como lo conocemos hoy día fue el de asfalto en escala 1/8, tracción trasera, sin suspensión, con motor de 3.5 cc, combustible metanol, y radio de dos canales, aunque hubo prototipos anteriores con motores de 6 cc, interiores al coche, que requerían refrigeración.

Los motores de aeromodelismo tuvieron que especializarse para coches: buena respuesta a todos los regímenes (un coche acelera y frena, a diferencia de un avión), refrigeración mejorada, y control de ruido.

Este tipo de vehículo se ha hecho cada vez más popular con el avance de la tecnología y su masificación.

Existen desde juguetes hasta automóviles radiocontrolados profesionales los cuales pueden clasificarse de a partir de diferentes características como:

- Tipo de motor; eléctrico o de combustión.
- Escala; tamaño del vehículo respecto a uno real y son las siguientes: ¼, 1/6, 1/8, 1/10, 1/16, 12/32 y 123/3. Las más populares son las 1:8 y 1:10.
- Tracción; que es el número de ruedas motrices. Sólo existen dos configuraciones:
- Tracción normal. Cuando las ruedas motrices son dos, las traseras o las delanteras, y también se designa por las siglas inglesas 2WD (2 Wheels Driving).
- Tracción integral. Cuando las cuatro ruedas del automodelo son motrices, y también se designa por las siglas inglesas 4WD (4 Wheels Driving).
- Categoría; es según si están diseñados para correr sobre asfalto o tierra, y si pueden soportar saltos sin dañarse. Las denominaciones utilizadas son Off road, On road, Monster, Drifting, Rockcrawling, Short Course, Bashers y Touring.

Las principales categorías, que están reconocidas por todas las federaciones internacionales son:

- Pista; On-Road. En esta categoría existen las escalas 1/4, 1/5, 1/8 y 1/10. En las escalas 1/4 y 1/5 (Gran Escala) sólo hay tracción 2WD; en la escala 1/8 la tracción es 4WD; mientras que en la escala 1/10 la tracción puede ser 2WD o 4WD.
- Todoterreno; Off-Road. En esta categoría está la escala 1/8 con tracción 4WD, la escala 1/10 con 2WD y 4WD.
- Turismo
- Rally; Sólo reconocida oficialmente en países como Italia y España. Oficialmente, sólo existen escalas 1/8 y 1/10.

Varios coches de RC comercialmente viables estaban disponibles a mediados de 1966, producido por la empresa italiana El-Gi de Reggio Emilia. Su primer modelo, un Ferrari 250LM 01:12 estaba disponible en el Reino Unido en diciembre de 1966, a través de importadores Motor Libros y Accesorios, St. Martins, Londres, y principios de 1967 a través de la tienda modelo de Atkinson en Swansea. Este modelo fue seguido por El-Gi 1:10 Ferrari P4, que se muestra por primera vez en la Feria del Juguete de Milán a principios de 1968.

***Fig.2.1**. Modelo Ferrari 250LM 01:12*

A mediados de los años 1960 una compañía británica, Mardave, con sede en Leicester, comenzó a producir coches de RC comercialmente viables. Sus primeros coches fueron nitro-o coches de gas vendidos en la zona a principios de 1970.

A principios de la década de 1970 varios productos comerciales fueron creados por las pequeñas empresas en los EE.UU. La mayoría de ellas empezaron como empresas de coches de ranura y con la decadencia de la popularidad de este género se movió en el campo de R/C.

Entre éstos se encontraban Associated Electrics, Thorp, Dynamic, Taurus, Delta, y Escorpión.

Estos kits de primeros eran de 1/8 escala nitro-powered coches de aluminio cacerola plana con motor de 0.21 o menor. Los cuerpos de estos coches fueron hechos de policarbonato. El motor más popular fue el K y B Veco McCoy.

El organismo regulador principal de las carreras de estos coches es Racers Auto.

En 1973-1974, Jerobee, una empresa con sede en el estado de Washington, creó su 1/12 coche nitro con un motor de 0.049 Cox. Varias compañías del mercado crearon piezas para este coche en particular los órganos claros de Lexan, disipadores de calor y tanques de combustible más grandes.

Esta escala se convirtió en 1/12 de carreras eléctrico escala al Associated Electrics creó el RC12E en 1976-77 - Jerobee convirtió Jomac y creó su propio kit eléctrico llamado el Rayo 2000 que ganó el "rugido" Campeonato Nacional de 1981 y 82 de 6 celdas de modificación y 82 las clases de producción de 6 celdas. El Rayo de 2000 fue diseñado por Don McKay y Jon Congdon.

A finales de 1970, los intereses en escala 1/12 de carreras eléctrico comenzó a crecer como octavo corredores IC escala, la única categoría que compite con en el tiempo, que necesitan para correr durante todo el invierno como una alternativa a los coches IC impracticables comenzó a la raza 1/12 coches, por lo tanto, se ha desarrollado una serie nacional de invierno. Como resultado de ello, la serie creció en popularidad como un gran número de coches scratch.

En 1976, la firma japonesa Tamiya, que fue reconocida por sus kits modelo plásticos decorados con detalle, lanzó una serie mecánicamente sencilla en carretera; los modelos de automóviles que se venden como adecuados para el control de la radio Tamiya pronto comenzaron. Se produjeron automóviles más a propósito del control remoto, y lanzaron buggies todo terreno que ofrecen sistemas de suspensión reales.

Fue esta progresión hacia la clase todoterreno la que provocó gran parte de la popularidad de la afición, ya que significaba coches teledirigidos que ya no se limitaban al asfalto y superficies lisas, pero podría ser conducido prácticamente en cualquier lugar.

La primera línea verdadera Tamiya de vehículos todoterreno fuera el Scorcher Arena y el Rough Rider, ambos lanzados en 1979, y ambos basados en diseños realistas buggy.

Fig.2.2. *Modelo Scorcher Arena*

Tamiya continuó produciendo los vehículos con suspensiones de trabajo, motores más potentes, con textura de goma neumáticos off-road y diversos organismos estilizados "buggy".

También produjeron camiones, tales como la HiLux Pickup Toyota, que contó con 3 cajas de cambio de velocidad y sistemas de suspensión de muelle de lámina.

Todos estos modelos eran realistas, duraderos, sencillos de montar, susceptibles de ser modificados y fáciles de reparar. Eran tan populares que podrían ser acreditados como el apogeo en los coches modelo de radio control en la primera mitad de 1980, y sirvieron de base para el mercado de coches de radio control de hoy.

Los modelos más populares Tamiya fueron los modelos de camiones monstruo Clodbuster saltamontes y los buggies Hornet, así como el Blackfoot y los primeros modelos.

Reconociendo su continua popularidad, varios de los primeros kits incluso han sido re-lanzado por Tamiya durante el período 2005-2007, con algunas modificaciones.

Una empresa británica, Schumacher Racing, fue el primero en desarrollar un diferencial de bolas ajustable en 1980, lo que permitió tuning casi infinito para varias condiciones de la pista. Por el momento la mayoría de los coches de carretera habían tenido un eje sólido, mientras que los coches fuera de la carretera generalmente tenían un diferencial de tipo de engranajes. Team Associated hizo lo mismo con la introducción del gas RC100 escala 1/8 coche en carretera, RC12 1/12 escala de coches eléctricos en la carretera y RC10 escala 1/10 todo terreno eléctrico coche de carreras en 1984. Equipo Losi siguió con la introducción de la JRX2 en 1988.

Fig.2.3. *Losi JRX2*

Como fechas significativas en Automodelismo, podemos indicar:

- 1962: Jerry Pullen y Doug Spreng estalecen las bases del control digital proporcional y del servo realimentado en posición con potenciómetro mientras trabajaban en el Jet Propulsion Laboratory (JPL, Pasadena, California, USA).
- 1969: comienzos.
- 1971: carrera de 1/8 asfalto el lunes de Semana Santa (5 de abril) en Inglaterra.
- 1973: primer coche con diferencial trasero (John Thorp).
- 1974: primer campeonato europeo (1/8 asfalto). En Fórmula gana Franco Sabatini, y en Prototipos Vittorio Merlotti.
- 1977: primer campeonato del mundo (1/8 asfalto), en el circuito permanente de Pomona (California, USA), ganado por el americano Butch Kroels.
- 1978: primera Copa del Mundo en Mónaco.
- 1978: primer circuito de pista en España en Igualada.
- 1979: segundo campeonato del mundo (1/8 asfalto en Ginebra (Suiza), ganado por Phil Boot (Inglaterra).
- 1980: primeros coches 1/8 todo terreno (TT), sin suspensión y tracción trasera, coches que evolucionaron rápidamente a tener suspensiones y tracción total con tres diferenciales.
- 1980: primer campeonato de España en Igualada (1/8 pista).
- 1981: primer coche de asfalto con suspensión (AMPS), pero sin amortiguadores.
- 1981: primer campeonato de Europa de 1/8 TT.
- 1981: tercer campeonato del mundo (1/8 asfalto), en Indianápolis (USA), ganado por Arturo Carbonell (USA) con un coche sin suspensión, pero donde debutaron los coches con suspensión que en poco tiempo demostraron su eficacia. Se vio también algún ensayo de la tracción total.

- 1982: primeros coches eléctricos de competición, tracción trasera, escala 1/12, baterías de níquel-cadmio (NiCd), y control de velocidad mediante un servo que movía un reostato.
- 1982: primeros cambios de marcha automáticos.
- 1983: primer sistema automático (con transpondedor en coche) para contar vueltas (AMB: iniciales de Alfonsus Marie Bervoets). Este sistema inicialmente se basa en un ordenador específico, después en una tarjeta ISA para PC que lee los transpondedores y realiza la temporización, y finalmente es una caja externa que comunica con el PC por puerto serie. Los transpondedores empiezan siendo recargables, a cargo de la organización de carrera (que los reparte antes de cada carrera y los recoge al acabar ésta) y limitados primero a 10 y luego a 20 simultáneos; con el tiempo se digitalizan y se llega hacia 2002 al transpondedor personal (PT), alimentado desde el coche y con un único código, con lo que cada piloto posee uno y puede correr con él en todo el mundo.
- 1984: primer coche comercial para asfalto con tracción total mediante tres diferenciales (SG). La doble tracción en estos coches ha cambiado a ser por correa del tren trasero al delantero, y rodamientos de un solo sentido ("one way") en el tren delantero, con diferencial único (del que se suele prescindir) en el eje trasero.
- 1985: primera moto con motor de explosión de 3.5 cc (DWA) en escala 1/4, difícil modalidad que no cuajó hasta muchos años después.
- 1985: primer coche comercial con motor de gasolina, escala 1/4 (DWA). Bycmo, en España, intentó, sin éxito, que cuajase la escala 1/5, más pequeña y manejable, y que años más tarde haría olvidar la 1/4. Posteriormente, Bycmo populariza la escala 1/7 en la modalidad Rallygame.
- 1986: primer campeonato del mundo de 1/8 TT en Grenoble (Francia), ganado por Veyssere (Francia). Muchos observadores japoneses hubo allí, lo que se tradujo en que en poco tiempo Japón produjo coches de todo terreno de gran éxito.
- 1987: primer coche de todo terreno en escala 1/6 con motor de gasolina. Se popularizan muchos años después.
- 1992: los programas de gestión de carreras posibilitan las carreras de 100 y más corredores, distribuyendo mangas, leyendo transpondedores, realizando instantáneamente las clasificaciones, y componiendo sub-finales.
- 1993: aparecen motores de 2.5 cc y coches escala 1/10 (pista y todo terreno), al objeto de reducir costos. Para simplificar su arranque, se incorpora un tirador en la trasera del bloque motor, que mueve el cigüeñal. Los coches tienen la misma estructura mecánica que los 1/8. Con el tiempo, motores y coches se hacen plenamente competitivos, y se prescinde del tirador.
- 1997: declina la escala 1/4, en favor de la 1/5. Durante algunos años se disputan carreras de F1 en escala 1/4, imitación exacta de los reales, pero asimismo terminarían con los años haciéndose populares estos coches en escala 1/5.
- 1999: en pista se popularizan los motores de 2.1 y 2.5 cc en las escalas de 1/10 con 220 y 200 mm (Touring) de anchura. Estos coches heredan la calidad de materiales de la escala 1/8, pero con una potencia menor se consigue una mucho mayor fiabilidad, con una notable reducción de costes.
- 2000: se popularizan las baterías de níquel-hidruro de metal (NiMH) de 3000 mAH para tracción de coches eléctricos. Con el tiempo, se popularizan estas baterías en tamaños AA (2000 a 2700 mAh) y AAA (750 a 1000 mAh), sustituyendo las usadas de NiCd en el equipo de radio.
- 2001: evolución en motores eléctricos y ESC's, apareciendo los motores sin colector específicos para automodelismo. Son una adaptación de la idea de los motores paso a paso, y su ESC es específico. Se hacen populares en aviones y barcos, pero su autorización en carreras de coches se retrasa.

- 2002: Bycmo introduce coches por fascículos a través de Ediciones Altaya. Son coches de iniciación completos, que facilitan en gran manera la introducción al automodelismo, ya que lo hace llegar a todos los rincones de España a través de los quioscos y una importante campaña de televisión. Con el tiempo esta idea se exporta y se copia.
- 2003: las baterías de NiMH en tamaño AAA y capacidades de 750 a 1000 mAh posibilitan la tracción en coches eléctricos en escalas 1/18 y 1/24, y dan autonomía suficiente para desarrollar carreras de hasta 20 minutos.
- 2004: aunque ya existían diversos juegos para PC de simuladores de carreras de coches, eran menos los orientados a R/C, donde el puesto de conducción está en el pódium. Con el aumento de velocidad en los procesadores y las tarjetas gráficas se consiguen simulaciones muy reales de conducción desde el pódium. El 28/10/04 Virtual RC lanza un juego casi gratuito, y comienzan las competiciones por Internet, iniciándose en modalidad individual contra el cronómetro.
- 2005: el 28/07/05 Virtual RC lanza la segunda versión de su juego, con carreras individuales contra el cronómetro, incluyendo coches generados por el ordenador.
- 2005: aparecen equipos de radio basados en DSS ("Digital Spread Spectrum") que posibilitan controlar el coche sin preocuparse de frecuencias ni cambios de cuarzo, así como un gran número de coches corriendo simultáneamente.
- 2006: las baterías de polímero de litio (LiPo), con las cuales había avanzado grandemente el aeromodelismo eléctrico, alcanzan una gran madurez para uso automodelero en escala 1/18, popularizándose asimismo los circuitos de mini RC al aire libre, tanto de asfalto como de todo terreno. Posteriormente su uso se extiende a otras escalas eléctricas.
- 2006: primer campeonato del mundo de motos de pista en escala 1/5.
- 2009: aparecen servos y receptores que admiten alimentación a 7.4V nominales, lo que permite la utilización de baterías LiPo en coches de explosión, reduciendo peso y bajando el centro de gravedad.
- 2011: el 01/09/11 se lanza el simulador sucesor del Virtual RC: VRC Pro.
- 2011: el 06/09/11 nos deja Ted Longshaw (1926-2011), un verdadero caballero, primer presidente de IFMAR.
- 2012: coches todo terreno en el simulador VRC Pro.

2.2.1. Desarrollo en España

En España, los comienzos son a partir del primer circuito permanente de asfalto en Igualada (1978), tras lo que varios pilotos y organizadores entusiastas catalanes, madrileños y valencianos crearon AECAR (Asociación Española de Coches a Radiocontrol) en 1979, con Francisco Arnaldo como primer presidente, asociación que más tarde publicó el famoso Libro Rojo, primero donde aparecieron los reglamentos europeos traducidos.

En 1976 la afición a los automóviles y en particular al modelismo, impulsó a un grupo de Igualadinos a formar el primer club de automodelismo de España y uno de los pioneros en Europa.

Las primeras pruebas se realizaron en el aparcamiento de una conocida fábrica textil de Igualada y, en el año 1977, se inicia la construcción de una pista de asfalto especialmente concebida para estos coches en un terreno próximo al aeródromo General Vives Igualada-Odena.

En 1978 el Club ICAR pasa a formar parte de la Federación Europea “EFRA”, recién constituida, y entra en lo que podría llamar “El circo de la Formula 1 a escala”

La primera prueba importante se desarrolla ese
mismo año con el primer Campeonato de Cataluña. El evento tiene mucha repercusión dando lugar a la concesión del Gran Premio Efra 1979, siendo esta la primera prueba internacional realizada en España.

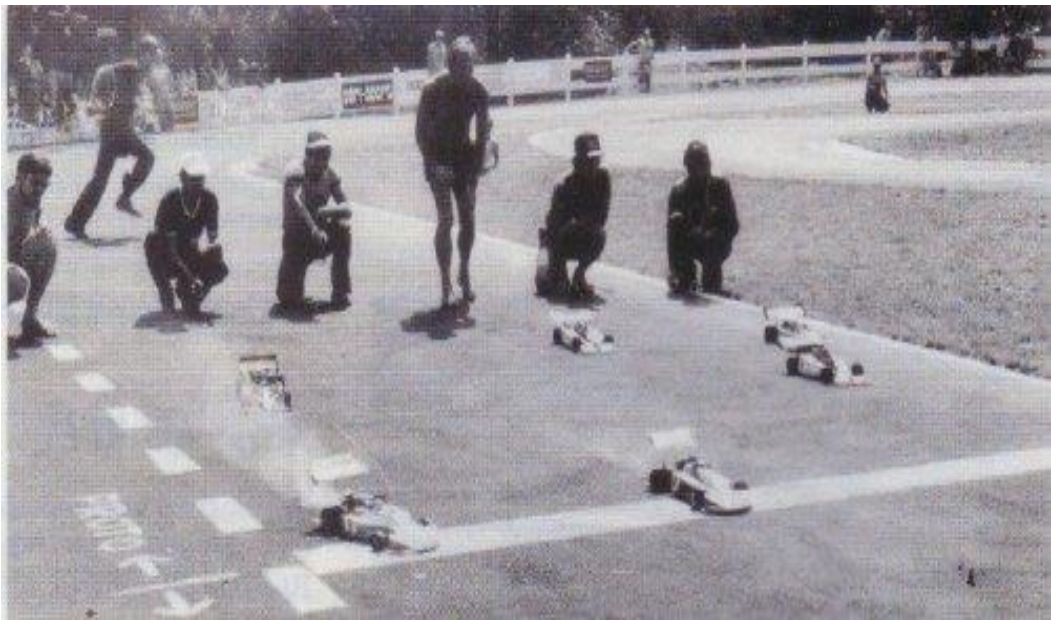


Fig.2.4. *Gran Premio Efra, 1979*

Al circuito de Igualada le siguieron los también permanentes de asfalto de Alcira (Valencia), sede del inolvidable trofeo Carmen Picó en 1980, Miralbueno (Zaragoza) y Paesa (Madrid), pero los circuitos más comunes han sido los de todo terreno, de construcción mucho más fácil, rápida y barata.

Los fabricantes nacionales han estado presentes.

Hubo coches de pista 1/8 de Tecnic, Zamicar, Bycmo y Modelhob; ésta última hizo intentos con un motor de 3.5 cc para coche.

Bycmo hizo asimismo coches de pista y todo terreno 1/8, ganando varias veces el campeonato de España TT, así como camiones 1/7 y coches 1/4 y 1/5 (motor de gasolina), y desde 1991 coches 1/7 (en esta escala, hasta 1998, ha producido más de 30000 unidades); en 2002 hace llegar coches por fascículos a todos los quioscos de España y asimismo exporta esta idea.

Crojet desarrolla desde 1995 una línea completa en 1/4 y 1/5; posteriormente lo hace Contrast.

En el año 1996, el club ICAR emprendió un ambicioso proyecto que culminó en la construcción de un nuevo circuito, al que se bautizó con el nombre de ICAR-II.

Este nuevo circuito ocupaba una superficie de 6500m2., disponía de una pista de 480mt. De longitud y 5m de ancho y fue considerado como el mejor de Europa.

Fig.2.5. *Circuito ICAR II, 1996*

En la actualidad, en el año 2011, el club ICAR emprende un nuevo proyecto, en este caso manteniendo la pista de RC original, y construyendo lo que se ha denominado ICAR INDOOR, en unas instalaciones con 510m2.
En esta nueva etapa el club apuesta por introducir el Slot dentro de sus disciplinas, pasando de este modo a ampliar la oferta existente y poniendo a disposición de todos, además del ya conocido y habitual trazado para las carreras de RC:

- Pista de velocidad 1/24 con 6 carriles y un recorrido de 64 mt.
- Pista de velocidad 1/32 con 6 carriles y un recorrido de 47 mt.
- Pista de RC para coches eléctricos de escala 1/12, 1/18 y Mini Z

Cada sección está equipada individualmente con control y monitorización de tiempos y boxes

Fig.2.6. .*ICAR Indoor, 2011*

2.3. ASPECTOS TÉCNICOS

A continuación se describen los principios de funcionamiento de los vehículos por RC, tanto analógicos como digitales, destacando las características que los diferencian.
Además, también se tratarán los aspectos físicos que intervienen en el mecanismo de los automóviles.

2.3.1. Funcionamiento modo analógico.

En el mundo del automodelismo y, sobretodo, a comienzos, el funcionamiento del coche rc venía dado por un sistema interno analógico.

La pieza fundamental de este sistema es el servo controlador, en este caso, analógico, cuya metodología explicaremos a continuación.

- ***El servo analógico o estándar***

El componente principal de un servo es un motor de corriente continua, que realiza la función de actuador en el dispositivo: al aplicarse un voltaje entre sus dos terminales, el motor gira en un sentido a alta velocidad, pero produciendo un bajo par. Para aumentar el par del dispositivo, se utiliza una caja reductora, que transforma gran parte de la velocidad de giro en torsión.

Tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y de mantenerse estable en dicha posición.
El dispositivo utiliza un circuito de control para realizar la ubicación del motor en un punto, consistente en un controlador proporcional de manera que, en un servo analógico, la rampa de subida de la señal PWM dispara un monoestable cuyo tiempo en alto viene dado por la resistencia del potenciómetro del servo; y ambas señales (la recibida y la generada) son comparadas por una puerta XOR.

Si ambos pulsos son de la misma duración, la salida es "0" y no se actúa sobre el motor porque el servo está donde tiene que estar; si son de distinta duración, la salida es un pulso cuya posición (respecto a la "duración central" de 1,5ms) y anchura dependen de la diferencia de duración de ambas señales. Es este pulso el que se amplifica en corriente y se entrega al motor para moverlo.

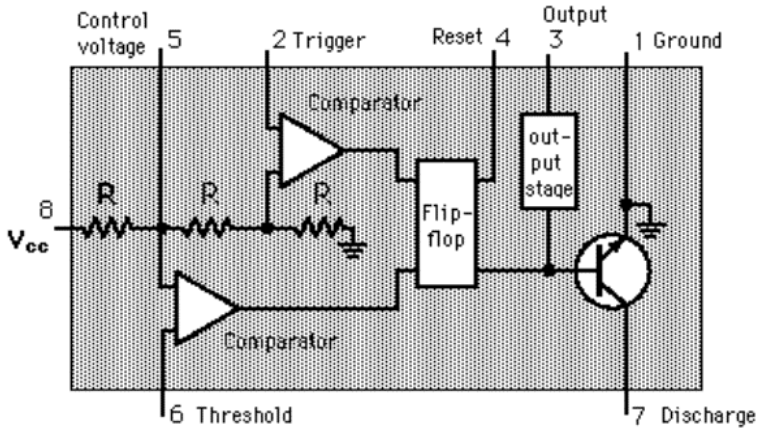


***Fig.2.7**. Esquema básico servo analógico*

El servo, en radiocontrol, se encarga de transformar las órdenes enviadas por el receptor en movimientos de rotación que, a su vez, por medio de transmisiones de mando unidos a éstos se convierten en movimientos de desplazamiento o traslación.

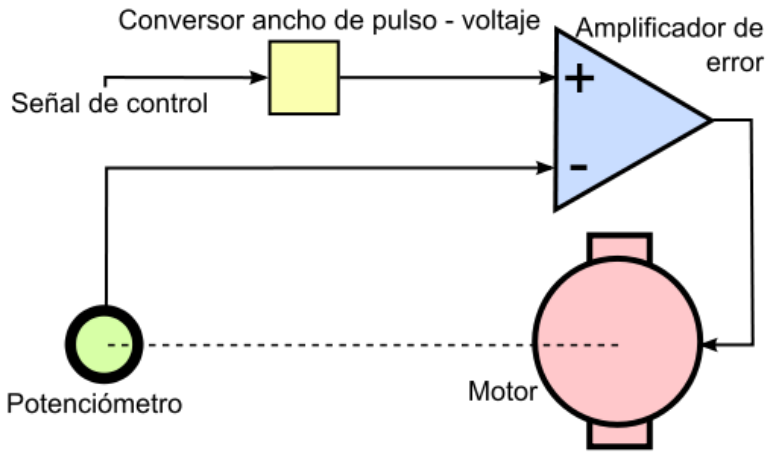


Fig.2.8. *Esquema general de funcionamiento de un servomotor*

Un servo se constituye principalmente de un motor y una reductora. El movimiento de salida de éste es una rotación. Una vez que el **motor** gira, el eje del servo cambia de posición, lo que modifica la resistencia del potenciómetro.

La función de la placa de control es la de controlar el motor para que la posición del eje de salida esté de acuerdo a la orden recibida desde el receptor.

La orden consiste en un pequeño impulso eléctrico cuya duración está comprendida entre 0,5 ms y 2,5 ms. El valor de 1,5 ms da al servo su posición central. Las órdenes de posiciones son transmitidas bajo la forma de una señal codificada en amplitud de impulso, repetida periódicamente, en general cada 50 ms, lo que permite a la electrónica de control de corregir continuamente la posición angular del eje de salida.

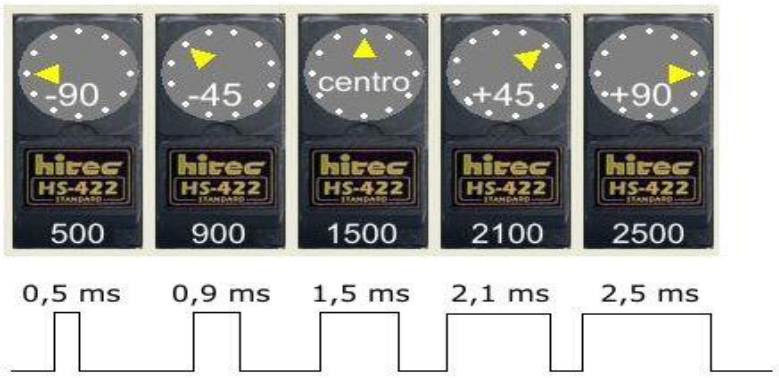


Fig.2.9.*Control del servo basado en el ancho de pulso*

El aumento de la amplitud de los impulsos actúa sobre el control de la velocidad, hasta que el brazo del servo se encuentra en la posición deseada. Durante la rotación, el potenciómetro indica al circuito electrónico el momento donde la posición deseada es alcanzada.

Los impulsos disminuyen entonces en amplitud hasta que ninguna tensión es aplicada al motor del servo, conservando el brazo en su nueva posición.

Dependiendo del modelo del servo, la tensión de alimentación puede estar comprendida entre los 4 y 8 voltios. El control de un servo se reduce a indicar su posición mediante una señal cuadrada de voltaje: el ángulo de ubicación del motor depende de la duración del nivel alto de la señal.

Cada servo, dependiendo de la marca y modelo utilizado, tiene sus propios márgenes de operación. Por ejemplo, para algunos servos los valores de tiempo de la señal en alto están entre 1 y 2 ms, que posicionan al motor en ambos extremos de giro (0° y 180°, respectivamente). Los valores de tiempo de alto para ubicar el motor en otras posiciones se hallan mediante una relación completamente lineal: el valor 1,5 ms indica la posición central, y otros valores de duración del pulso dejarían al motor en la posición proporcional a dicha duración.

Los servomotores tienen tres terminales de conexión: dos para la alimentación eléctrica del circuito y uno para la entrada de la señal de control. El voltaje de alimentación generalmente es de alrededor de 6 voltios, pues aunque el motor soporta mayores voltajes de trabajo, el circuito de control no lo hace.

Para poder entender de manera fácil y sencilla el funcionamiento del servo estandarizado, nos pondremos en situación del mando de control; nos guiaremos de las pulsaciones enviadas al servo para conocer su modo de actuación, así, en función de la posición del botón o interruptor pulsado, se generará un pulso de ancho constante y proporcional a dicha posición, de forma que a igualdad de posición corresponda igualdad de ancho de pulso.

Estos pulsos serán “encadenados” consecutivamente de forma que tendremos un tren de pulsos y luego un espacio sin señal para realizar la sincronía y así conseguir que el receptor “entienda” que ha finalizado un envío y se prepare para recibir el siguiente, en la siguiente figura vemos tres señales de control, y el espacio de sincronismo entre ellas;

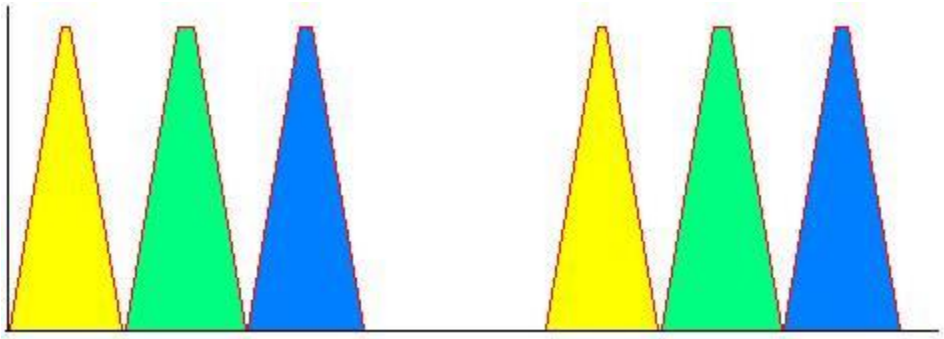


Fig.2.10. *Las tres señales de control y el espacio de sincronismo entre ellas en el servo*

Esa señal, al ser recibida en el receptor, se decodificaría a su vez en tres trenes de pulsos independientes para controlar por separado cada uno de los servos conectados. El ancho de pulso oscilará dependiendo del fabricante entre 500 y 2500 msg de forma que aproximadamente un pulso de 1500 msg se le denomina pulso neutro ya que lleva el brazo a posición 0º en caso no estar ya en dicha posición, un pulso de 500 msg representa un posicionamiento a –90º y uno de 2500msg provoca un posicionamiento a +90º, vemos por tanto que el ancho de pulso determina el ángulo de giro del servo.

Estos pulsos son recibidos e interpretados en el receptor y transferidos a la electrónica del servo la cual se encarga de realizar dos funciones indispensables, el control de posiciona-miento (para saber si hay diferencia entre la posición actual y la requerida usando el potenciómetro de posicionamiento) y la gestión de la potencia, que es enviada al motor, así pues, diferenciaremos entre la electrónica de control y la de potencia.

En un servo convencional cuando este se encuentra en espera (definiéndose posición de espera como aquella en la cual el servomotor permanece detenido porque ha alcanzado y mantiene la posición solicitada por la emisora, que no tiene por qué coincidir obligatoriamente con la posición 0 del servomotor) no se envía tensión al motor de posicionamiento, cuando varía el ancho de pulso enviado por la emisora o se ejerce una fuerza sobre el brazo del servo que provoca la variación de posición del mismo, la electrónica de control responde ordenando a la electrónica de potencia que alimente al servomotor para mantener o alcanzar una nueva posición.

***Fig.2.11**. Servo estándar*

El control de la potencia enviada consiste en “chopear” la tensión nominal de alimentación del servo a una frecuencia de 50 ciclos por segundo (Herzios), esto es lo que se conoce como control de potencia por ancho de pulso, ósea un control “PWM”, este tratamiento conocido como modulación del ancho de pulso consiste en suministrar esa tensión pero no constante en el tiempo sino que lo haremos a intervalos regulares de tiempo constantes con lo cual lo que le está llegando al motor son pulsos de tensión continua , como hemos dicho que este proceso se realiza a 50 ciclos por segundo, los pulsos de alimentación le estarán llegando al motor cada 1/50 segundos, es decir, cada 20 milisegundos.

En cuanto a la ***velocidad de posicionamiento***, si variamos el ancho de los pulsos generados en la electrónica de potencia lo que estamos haciendo es aumentar el ciclo de trabajo teniendo en cuenta que el ciclo de trabajo es el resultado de dividir el tiempo durante el cual aplicamos tensión al motor entre el tiempo total del ciclo, es decir si el pulso tiene un ancho de 10 msg y el ancho total del ciclo es de 20msg, tendremos un ciclo de trabajo del 50% ya que estamos aplicando tensión durante más tiempo, luego cuanto más alto sea el ciclo de trabajo, más velocidad desarrollará el servomotor y por tanto el posicionamiento será más rápido.

El segundo parámetro característico de un servo es el ***posicionamiento*****;** por el potenciómetro de realimentación, la electrónica de control recibe pulsos del receptor de un ancho correspondiente a la posición deseada, a través del potenciómetro y de otros elementos obtenemos los pulsos con ancho correspondiente a la posición actual, comparándolos se obtiene el error de posicionamiento, si existe error se activa la electrónica de potencia para corregirlo, a mayor error en la posición mayor ciclo de trabajo y por tanto mayor velocidad, a medida que

disminuye el error la electrónica de potencia disminuye también el ciclo de trabajo hasta alcanzar ciclo de trabajo 0.

Nos centraremos ahora en el tercer parámetro de funcionamiento de un servo, la ***banda muerta*** o "Deadband"; sabemos ya que un pulso muy estrecho, es decir, un ciclo de trabajo muy bajo, no proporciona prácticamente ningún desplazamiento, ya que la tensión aplicada durante un lapso tan breve de tiempo no será capaz de vencer la fuerza contraelectromotriz, pues bien, definimos banda muerta como el recorrido mínimo de palote, volante o gatillo de emisora necesario para que observemos desplazamiento en el brazo del servo.

El último parámetro es la ***resolución,*** la cual se define como la mínima variación de posición alcanzable por el servo, aquí intervienen varios factores como son la precisión del potenciómetro de realimentación de posición y sobretodo la frecuencia de trabajo, ya que la posición no se variará con periodos inferiores a 20 milisegundos, es decir, cada 20 msg se generará un pulso de ancho x para llevar el brazo hasta la posición deseada.

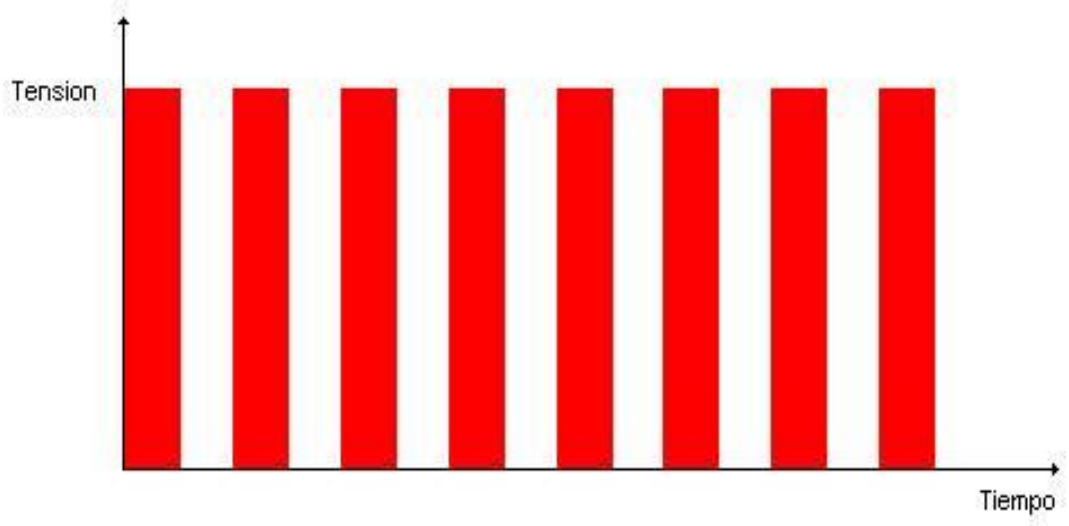


Fig.2.12*.Ciclo de trabajo del 50% en un servo estándar a 50 Hz*

2.3.2. Funcionamiento modo digital

De igual manera que se utiliza el servo estándar o analógico para el control del vehículo, en la última década se comenzó a incorporar el servo digital de manera que, junto a las prestaciones del mismo, se consigue optimizar el mando de dirección y velocidad del coche radiocontrolado.

A continuación, procedemos a definir e introducir de manera general los mecanismos que caracterizan al servo digital.

- ***El servo digital***

Los servos digitales son similares a los servos analógicos, pero cuentan con ciertas ventajas como lo son un mayor par, una mayor precisión, un tiempo de respuesta menor, y la posibilidad de modificar parámetros básicos de funcionamiento como ángulos máximo y mínimo de trabajo, velocidad de respuesta, sentido de giro y posición central, entre otros.

Un servo digital es lo mismo que un servo estándar con la diferencia de que incorpora un cristal de cuarzo y un microprocesador el cual analiza la señal enviada por el receptor a la vez que se encarga de controlar el funcionamiento del servomotor.

Es incorrecto pensar que un servo digital es completamente diferente a un estándar en cuanto a arquitectura hardware; un servo digital incorpora el mismo motor, piñones y caja que los estándar, e incluso y lo más importante es que también disponen de un potenciómetro para la realimentación de posición.

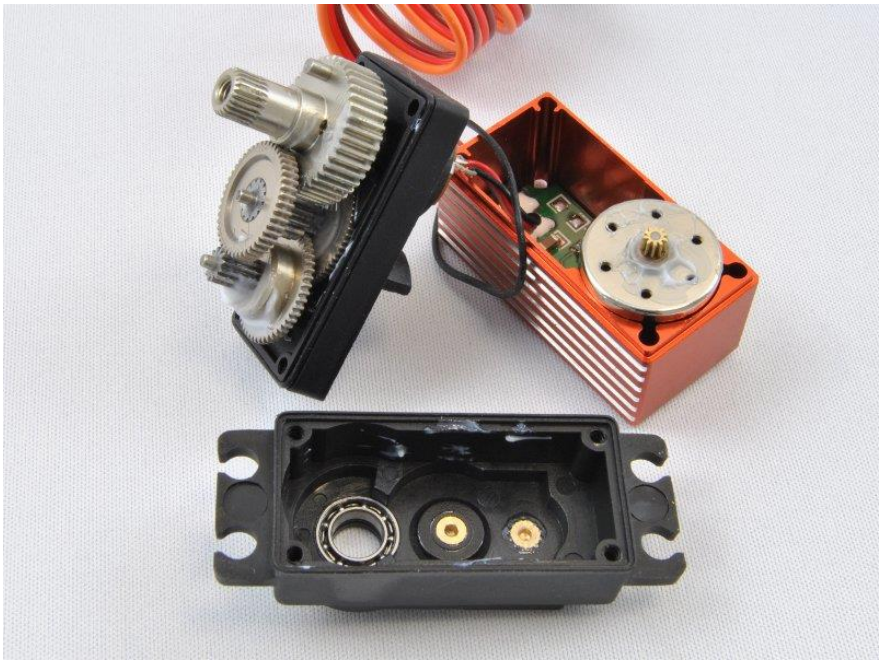


Fig.2.13. *Servo digital*

Además de un mayor costo, tienen la desventaja de que requieren más energía para su funcionamiento, lo cual es crítico cuando se utilizan en aplicaciones que requieren el máximo ahorro de energía posible, tales como robots robustos o aviones radiocontrolados.

Los servos digitales poseen también un motor eléctrico, engranajes reductores y potenciómetro, pero se diferencian de los analógicos por el hecho de que incorporan un microprocesador que analiza la señal recibida del receptor y controla el motor del servo.

La diferencia principal radica en la manera de tratar la información recibida del receptor y el control ejercido sobre la tensión eléctrica aplicada al motor, lo que permite de reducir la zona neutra, aumentando así la amplitud del movimiento y generando una gran estabilidad del posicionamiento estático.

Como hemos dicho, al llevar integrado un microprocesador, es capaz de variar la forma en la que se envía potencia al servomotor, lo que significa que modifica el ancho de los pulsos y por tanto el ciclo de trabajo en función de unos parámetros de funcionamiento internos y ya no solo en función de la señal enviada por el receptor, de forma que optimice el rendimiento del servomotor, también es posible modificar el funcionamiento en función de nuestras necesidades, por ejemplo, invertir el sentido de giro, la velocidad de desplazamiento, ancho de pulso neutro, etc.

Es capaz de aumentar la frecuencia de trabajo, si con un servo estándar teníamos 50 ciclos por segundo ahora podremos tener hasta 300 ciclos por segundo con lo cual la duración del periodo baja hasta los 1/300 = 3,33 ms, lógicamente al disminuir el periodo proporcionalmente también disminuirá el ancho de pulso manejable, pero el ciclo de trabajo permanecerá constante, con lo cual conseguimos enviar pulsos mucho más estrechos pero con más frecuencia, debido a las características constructivas y de funcionamiento de cualquier motor eléctrico se da la circunstancia de que es precisamente esta situación en la que se obtiene un mayor rendimiento del mismo**,** ya que con frecuencias muy altas no se descarga la bobina equivalente creada por el inducido del motor y los picos de corriente son menores, es por tanto más efectivo, en general en un motor el rendimiento es proporcional a la frecuencia de trabajo.

Con este aumento de potencia no solo se consigue aumentar la velocidad de respuesta ante una variación del comando de posicionamiento si no que la variación del aumento o disminución de la potencia suministrada al aumentar la frecuencia proporciona una disminución de la banda muerta, una aceleración / deceleración mucho más rápida y suave**,** mayor resolución en el posicionamiento y un mayor par.

Dicho aumento de par se ve reflejado tanto en funcionamiento estático como dinámico, es decir, cuando el servo está detenido en una posición, la fuerza que hay que ejercer sobre el brazo del mismo para conseguir que gire es muy superior a la de un servo estándar, asimismo el par de giro suministrado cuando está realizando un desplazamiento es tres veces superior al de un servo estándar.

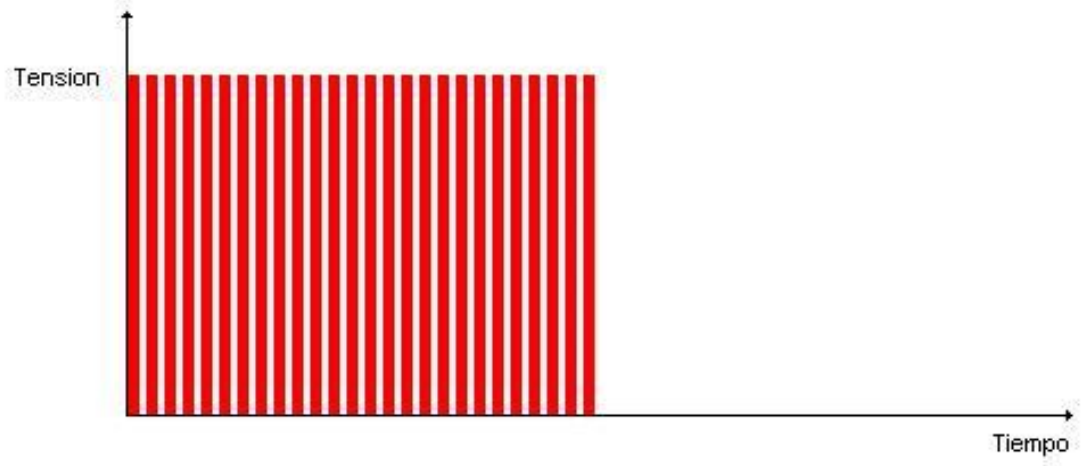


Fig.2.14*. Ciclo de trabajo del 50% en un servo digital a 300 Hz*

2.3.3. Diferencias modo analógico-modo digital.

La diferencia principal entre ambos tipos reside en la manera en la que se procesa la señal recibida desde el receptor, y en como controla el envío de potencia al servomotor de posicionamiento.

En el servo analógico convencional se utiliza gestión analógica para interpretar el tren de pulsos y traducirlo a una posición de potenciómetro, que es donde nuestra cabeza de servo irá.

En el digital se utilizan métodos digitales para convertir el tren de pulsos en una posición. En el caso de Sävox, se divide el potenciómetro en 4096 partes o posiciones que pueden tomar la cabeza del servo.

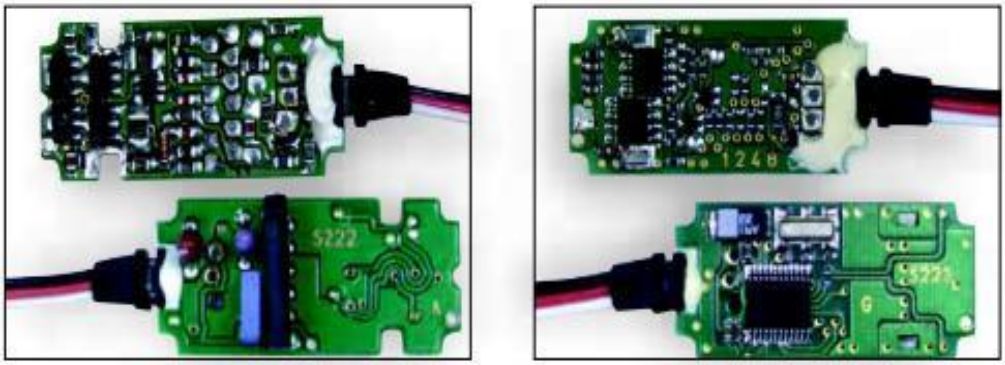


Fig.2.15. *Servo estándar (izda.) y servo digital (dcha.)*

2.3.4. Ventajas y desventajas modo analógico frente modo digital

- **MODO ANALÓGICO**

  - VENTAJAS

  - Circuito sencillo basado en comparadores y amplificadores operacionales que siempre funcionan de la misma forma.
  - Bajo costo debido a su sencilla circuitería.
  - En muchas ocasiones, mayor durabilidad ya que no incorporan el cristal de cuarzo que corresponde al servo digital.

  - DESVENTAJAS

  - Poco flexibles en cuanto a la señal debido a la carencia de microprocesador interno
  - Sensibilidad al ruido
  - Menor precisión que el servo digital
  - Señal decodificada en ancho de pulso
  - Lentitud en el desplazamiento hacia la posición correcta ya que el pulso de la señal será más corto

- **MODO DIGITAL**

  - VENTAJAS

  - Reducción de la banda muerta por lo que genera el aumento de la resolución generando valores del par estático y dinámico mucho más elevados.
  - Repuesta más rápida ante órdenes de control
  - Mayor eficacia en posicionamiento; mayor repetitividad.
  - Optimización del rendimiento.
  - Mayor inmunidad al ruido
  - Mayor precisión

- DESVENTAJAS

- Aumento del consumo de potencia de aproximadamente un 60%
- Inutilización del uso del sistema “BEC” debido a la alta demanda de tensión
- Coste elevado
- Mayor peso
- Mayor fragilidad dada su implementación interna

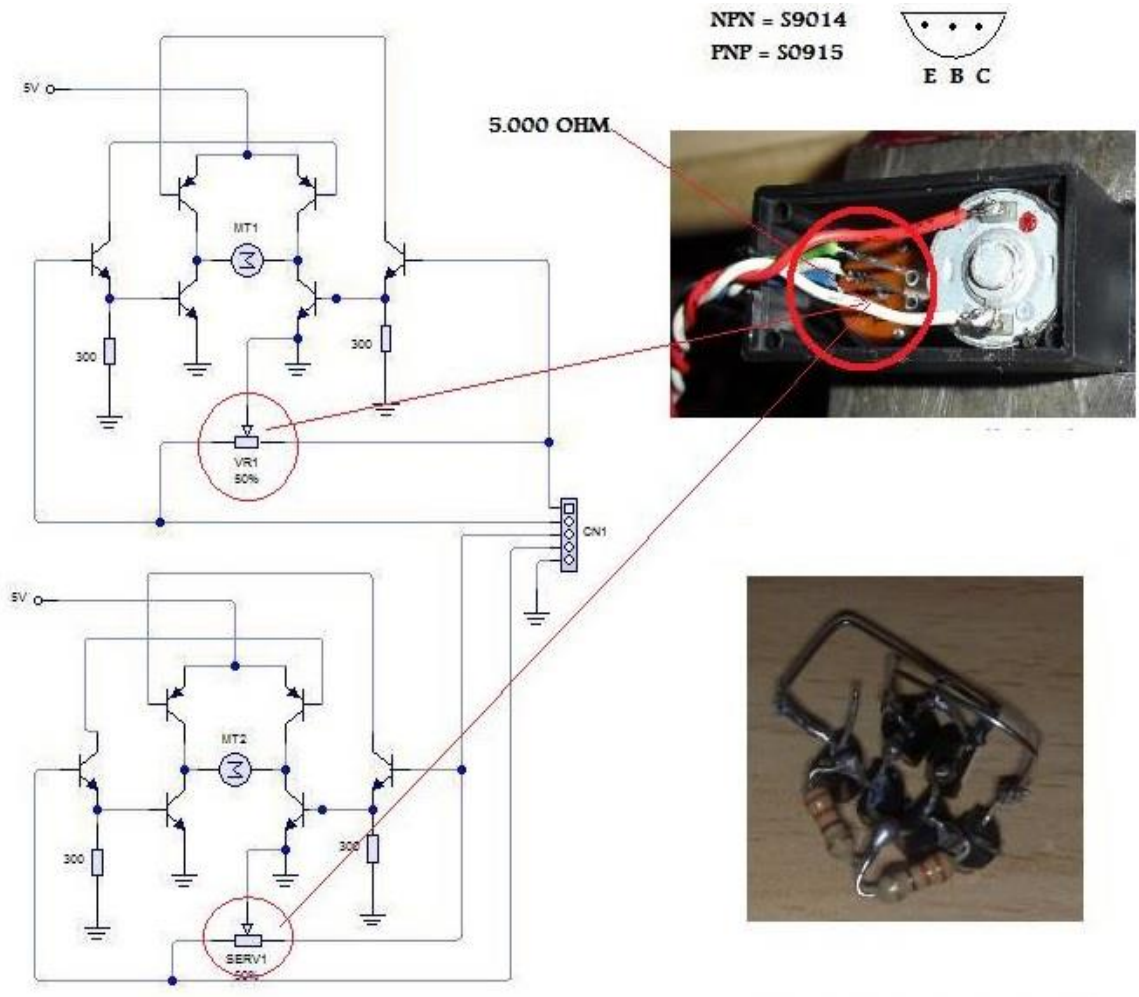


***Fig.2.16**. Esquemas de servo analógico (arriba) y digital (abajo)*

2.3.5. Componentes del servo de control

En este apartado veremos los diferentes elementos que componen al servo controlador, principal accionador del movimiento del coche teledirigido.
Dependiendo del servo, la composición interna variará aunque no de manera demasiado notable ya que el funcionamiento de los diferentes servos es básicamente idéntico.

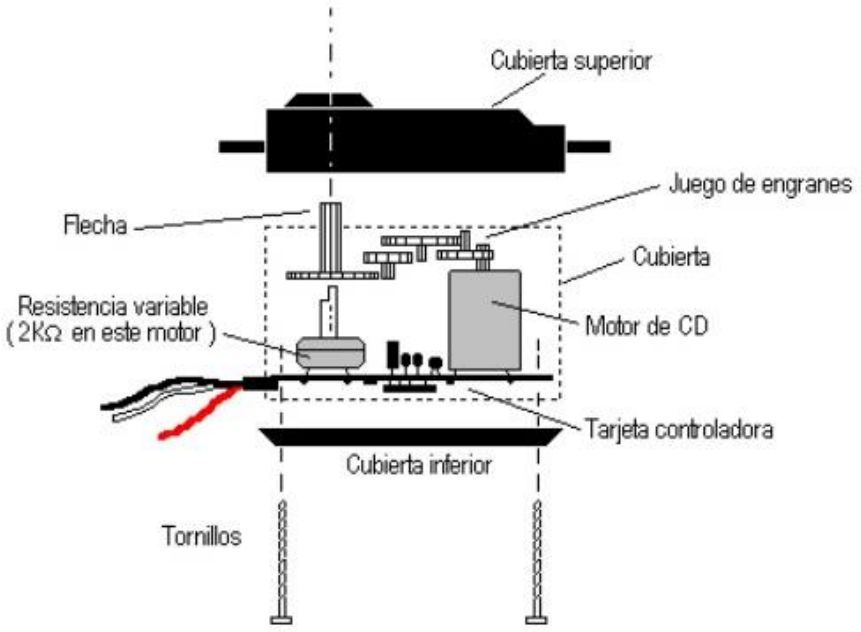


***Fig.2.17**. Despiece básico del servo*

Empezaremos hablando y describiendo brevemente el funcionamiento de los diferentes elementos, desde el propio motor de continua hasta el potenciómetro del que depende;

- **Motor de continua**

  Es una máquina que convierte la energía eléctrica en mecánica, provocando un movimiento de rotor, debido a la acción del campo magnético el cual brinda movilidad al servo. Cuando se aplica un potencial a sus dos terminales, este motor gira en un sentido a su velocidad máxima. Si el voltaje aplicado sus dos terminales es inverso, el sentido de giro también se invierte.
  Basa su funcionamiento en el rechazo que se produce entre el campo magnético que rodea al electroimán del rotor, campo inductor, y el campo magnético de un imán permanente colocado de forma fija en el cuerpo del motor, campo inducido.
  Se trata, la mayoría de las veces, de motores eléctricos sin escobillas o brushless de manera que se optimizaría el rendimiento del mismo ya que no existiría el rozamiento que se produciría por la presencia de las escobillas. La energía a la salida se deriva de los principios de la fuerza electromagnética cuya característica es el par de la fuerza electromotriz interna, Mi;

$$M_i = \frac{1}{2\pi}\,\frac{p}{a}\,\frac{N}{60}\,\Phi\, I_i$$

  Donde:

  Mi = par interno
  p = número de pares de polos de la máquina. Las máquinas eléctricas contienen un número par de polos que se designan por 2p.
  a = número de ramas en paralelo del inducido.
  N = número total de conductores.
  Φ= flujo magnético
  Ii = intensidad que recorre los conductores del inducido.

Cuando la máquina está ya construida, la mayoría de las magnitudes son constantes, por lo que la expresión que toma el par interno es la siguiente;

$$M_i = K \Phi I_i.$$

En cuanto a la fuerza electromotriz; E, tenemos que;

$$E_b = \frac{p}{a} \frac{N}{60} \Phi n$$

Y, como ya hemos puntualizado antes, cuando la máquina está ya contruida, se mayoría de magnitudes que entran en juego se vuelven constantes quedando la anterior expresión como;

$$E_b = K_b n \Phi$$

De manera que la fuerza contraelectromotriz es directamente proporcional a la velocidad de giro y al flujo inductor.
La tensión aplicada U y la Eb alcanzan su equilibrio, con lo que aparece la intensidad Ii

$$I_i = \frac{U - E.}{r_i}; U = E. + r_i I_i$$

- **Engranajes reductores**

Se encargan de convertir gran parte de la velocidad de giro del motor de corriente continua en torque.

Fig.2.18.*Tren de engranajes reductores de un servo*

- **Circuito de control**

Este tipo de circuitos se caracterizan porque se encargan de generar las señales necesarias para controlar los servos; Recibe los pulsos de entrada y ubica al motor en su nueva posición dependiendo de los pulsos recibidos.

Se lleva a cabo mediante una serie de pulsos tal que la duración del pulso indica el ángulo de giro del motor. Cada servo tiene sus márgenes de operación, que se corresponden con el ancho del pulso máximo y mínimo que el servo entiende.
Si el eje está en el ángulo correcto, entonces el motor está apagado pero si, por el contrario, el circuito comprueba que el ángulo no es correcto, el motor volverá a la dirección correcta, hasta llegar al ángulo deseado.
El eje del servo es capaz de llegar alrededor de los 180 grados. Normalmente, en algunos llega a los 210 grados pero varía según el fabricante.

Fig.2.19. *Circuito de control y realimentación*

- **Resistencia variable**

  Este potenciómetro se encuentra conectado al eje centrar ser servomotor; permite a la circuitería de control, supervisar el ángulo actual del servo motor.

- **Cableado**

  Terminal positivo: recibe la alimentación del motor (4 a 8 voltios)
  Terminal negativo: referencia tierra o masa del motor (0 voltios)
  Entrada de señal: recibe la señal de control del motor; ancho de pulso a decodificar

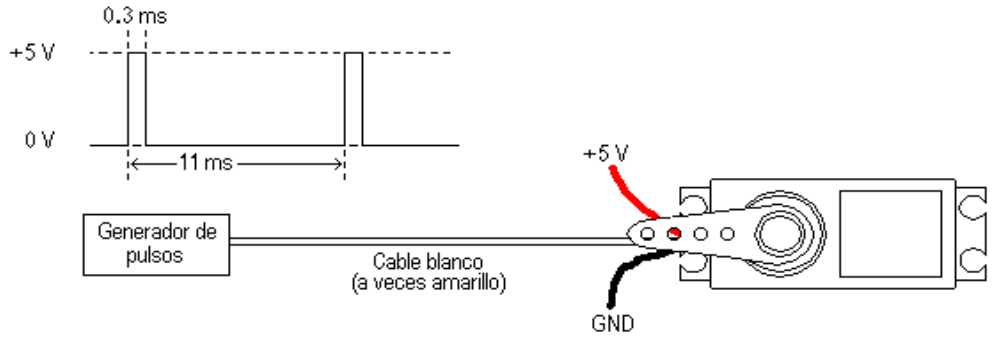


Fig.2.20. *Cableado servo*

2.3.6. Principios físicos

En este apartado, nos centraremos en los principios físicos básicos de la dinámica y cinemática que rigen al coche de radiocontrol, es decir, las expresiones y definiciones que ilustran las fuerzas y movimientos presentes en el mismo una vez puesto en marcha. Nos centraremos en el vehículo de radiocontrol ya que, por su parte, el mando que lo comanda carece de movimiento alguno, permaneciendo estático en sí mismo exceptuando, claro está, el joystick o botonera de control. El movimiento del vehículo, será MRU una vez estabilizado o MRUA en los procesos de arranque y frenado.

Para el análisis de los movimientos, haremos una introducción a las definiciones básicas de los conceptos mecánicos que intervienen en el proceso físico.

Consideremos una partícula o punto material moviéndose sobre una línea recta representada por la coordenada *x*. Supongamos que en el instante *ti* se encuentra en la posición *xi* y en el *tf* en la posición *xf*
Se define la velocidad media de la partícula en ese intervalo de tiempo como:

$$\bar{v} = \frac{x_f - x_i}{t_f - t_i} \equiv \frac{\Delta x}{\Delta t}$$

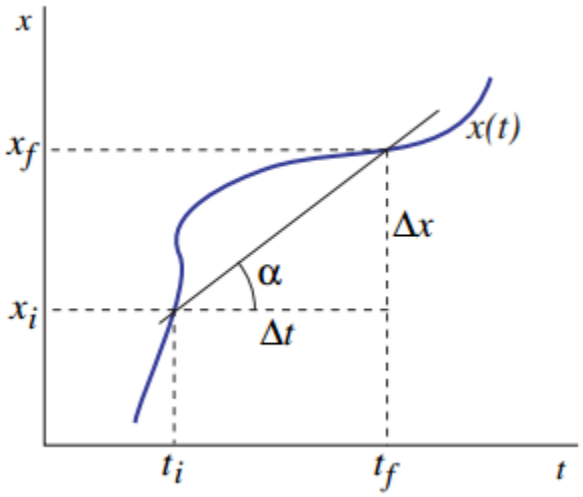


***Fig.2.21**. Gráfica velocidad en ejes espacio-tiempo*

La velocidad media es independiente de la trayectoria seguida por la partícula, sólo depende del espacio recorrido y el tiempo transcurrido. Si una partícula parte de un determinado punto y vuelve a él después de un tiempo, su velocidad media en ese intervalo es cero.

Geométricamente, la velocidad media representa la pendiente de la recta que une los puntos inicial y final.

$$\bar{v} = \frac{\Delta x}{\Delta t} = \tan \alpha.$$

La velocidad de la partícula en un instante de tiempo cualquiera se denomina velocidad instantánea, concepto importante especialmente cuando la velocidad media en diferentes intervalos de tiempo no es constante.

Para determinarla debemos hacer el intervalo temporal tan pequeño como sea posible de modo que esencialmente no tengan lugar cambios en el estado de movimiento durante el mismo;

$$v = \lim_{\Delta t \to 0} \bar{v} = \lim_{\Delta t \to 0} \frac{\Delta x}{\Delta t} = \frac{dx}{dt} \quad \Longrightarrow \quad v(t) = \frac{dx(t)}{dt}$$

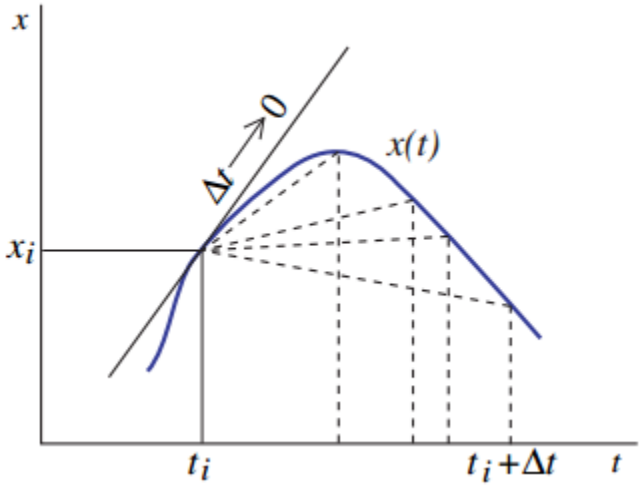


Fig.2.22. *Gráfica límite-pendiente de velocidad en ejes espacio-tiempo*

Cuando $\Delta t \to 0$, el cociente, $\Delta x/\Delta t$, representa la pendiente de la recta tangente a la curva, *x(t),* en el instante *ti* . Una vez conocida la velocidad como función del tiempo, *v = v(t),* es posible determinar la posición de la partícula en cualquier instante;

$$v = \frac{dx}{dt} \longrightarrow v\,dt = dx \longrightarrow \int_{x_0}^{x} dx = \int_{v_0}^{v} v(t)\,dt$$

$$\Longrightarrow \quad x = x_0 + \int_{t_0}^{t} v(t)\,dt$$

Ell desplazamiento, *x − x0*, se puede interpretar geométricamente como el área bajo la curva *v = v(t).*

Cuando la velocidad de una partícula permanece constante se dice que realiza un movimiento uniforme, pero en general la velocidad puede variar con el tiempo. Supongamos una partícula que en el instante *ti* tiene velocidad *vi* y en el *tf* velocidad *vf* .

Se define la aceleración media en ese intervalo como:

$$\bar{a} = \frac{v_f - v_i}{t_f - t_i} = \frac{\Delta v}{\Delta t}$$

Al igual que con la velocidad, conviene definir una aceleración instantánea como límite de la aceleración media en un intervalo temporal muy pequeño;

$$a = \lim_{\Delta t \to 0} \bar{a} = \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t} = \frac{dv}{dt} \quad \Longrightarrow \quad a(t) = \frac{dv(t)}{dt}$$

Si conocemos la aceleración instantánea, *a = a(t),* podemos calcular la velocidad instantánea, *v = v(t);*

$$a(t) = \frac{dv}{dt} \longrightarrow dv = a\,dt \longrightarrow \int_{v_0}^{v} dv = \int_{t_0}^{t} a\,dt \quad \Longrightarrow \quad v(t) = v_0 + \int_{t_0}^{t} a(t)\,dt$$

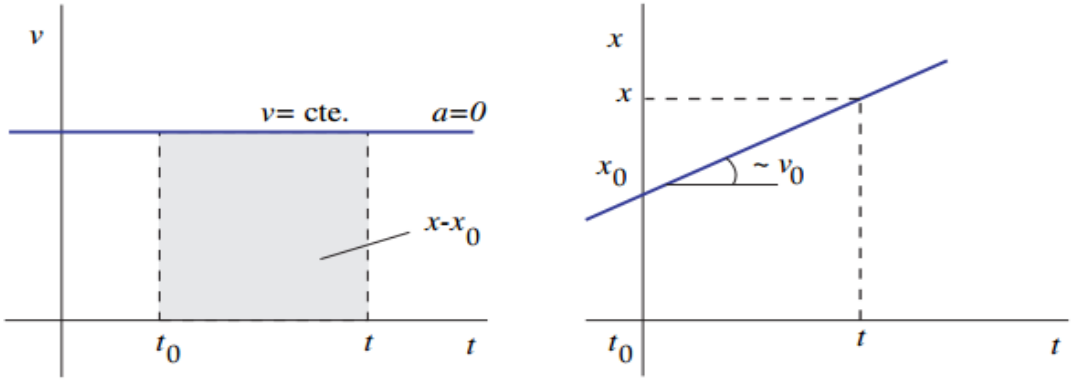
Visto lo anteriormente descrito respecto a la velocidad y aceleración relacionadas con la posición y el tiempo de la partícula, en este caso, de nuestro vehículo, procederemos a describir los movimientos en el proceso de manipulación; arranque, frenado y mantenimiento de la velocidad, es decir, el MRU y el MRUA; el primero se produce cuando *v ≡ v0 =cte* y el segundo cuando *a ≡ a0 =cte;*

$$x = x_0 + v_0 \int_{t_0}^{t} dt = x_0 + v_0(t - t_0)$$

$$v = v_0 + a_0 \int_{t_0}^{t} dt = v_0 + a_0(t - t_0) \quad \Longrightarrow \quad v(t) = v_0 + a_0(t - t_0)$$

La primera ecuación es la relación que liga posición con tiempo en un movimiento unidimensional uniforme, MRU y MRUA en la segunda.

MRU



MRUA

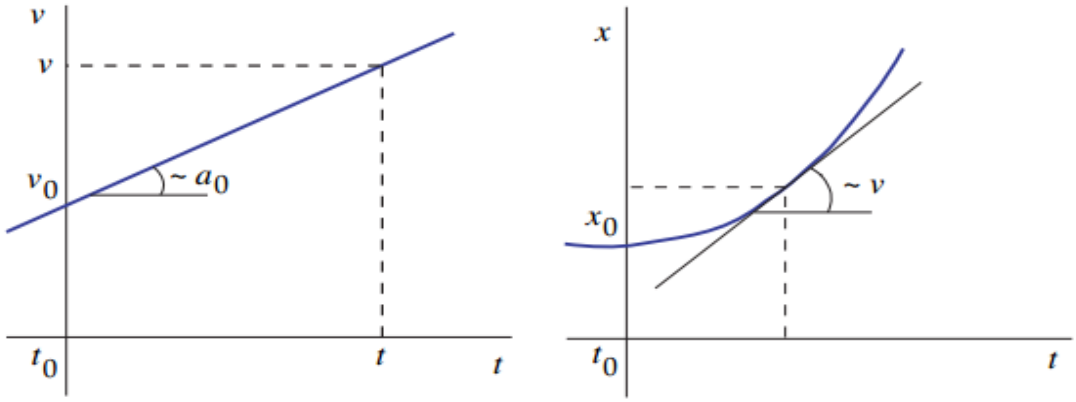


Fig.2.23*. Gráficas MRU (arriba) y MRUA (abajo)*

Esta partícula, que se ha estado moviendo en línea recta, se mueve ahora en el espacio. Denotamos su posición en cada instante de tiempo por medio de un vector posición $\vec{r} = \vec{r}(t)$ En coordenadas cartesianas, la ecuación de la trayectoria vendrá dada por*: x = x(t), y = y(t)* y *z = z(t).*

Definimos pues la velocidad media de nuestro vehículo en el intervalo temporal *(tf-ti)* como;

$$\vec{v} = \frac{\vec{r}_f - \vec{r}_i}{t_f - t_i} = \frac{\Delta\vec{r}}{\Delta t}$$

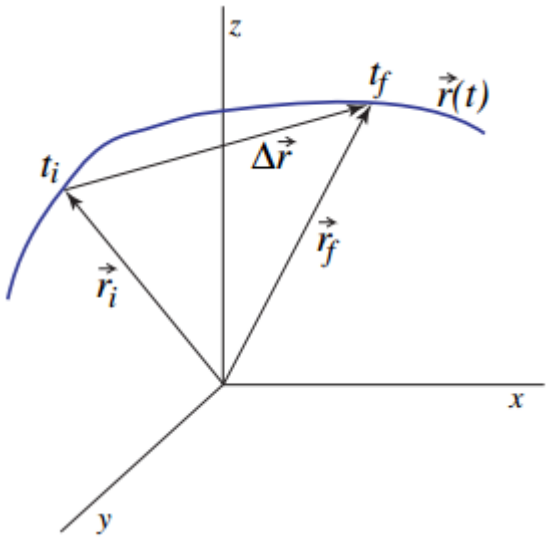


Fig.2.24*.Representación en ejes x, y, z del movimiento de una partícula en un tiempo tf-ti*

Para la velocidad instantánea tomaremos el límite cuando el intervalo temporal tiende a cero;

$$\vec{v} = \lim_{\Delta t \to 0} \frac{\Delta\vec{r}}{\Delta t} = \frac{d\vec{r}}{dt}$$

$$\vec{v} = \frac{dx}{dt}\vec{i} + \frac{dy}{dt}\vec{j} + \frac{dz}{dt}\vec{k} = v_x\vec{i} + v_y\vec{j} + v_x\vec{k}.$$

En un movimiento curvilíneo, la velocidad puede variar tanto en módulo como en dirección o sentido, por lo que definiremos la aceleración media como el cambio de velocidad en un intervalo temporal determinado:

$$\vec{\bar{a}} = \frac{\Delta\vec{v}}{\Delta t}$$

Y para la instantánea;

$$\vec{a} = \lim_{\Delta t \to 0} \vec{\bar{a}} = \lim_{\Delta t \to 0} \frac{\Delta\vec{v}}{\Delta t} = \frac{d\vec{v}}{dt} = \frac{dv_x}{dt}\vec{i} + \frac{dv_y}{dt}\vec{j} + \frac{dv_z}{dt}\vec{k}$$

Destacar que dicho vector de aceleración instantánea, tiene la misma dirección que el vector velocidad pero sin ser tangente ni perpendicular a la trayectoria, de manera que siempre está dirigido hacia la concavidad de la curva, es decir, hacia la región que contiene el centro de curvatura, el cual representa la trayectoria de la partícula ya que esa es la dirección en que cambia la velocidad.

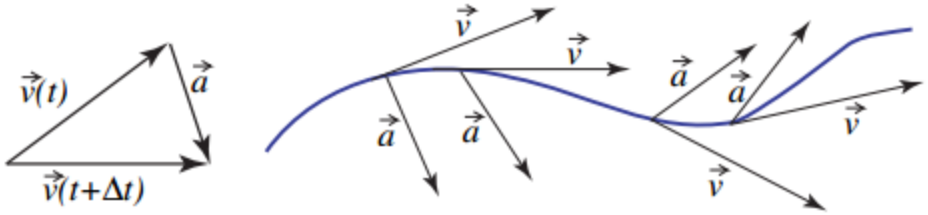


Fig.2.25 *.Representación gráfica del vector aceleración a lo largo del movimiento de una partícula*

Lo primero que hay que tener en cuenta es que un coche se mueve porque nosotros se lo ordenamos. Cuando un objeto se pone en movimiento influye sobre él una fuerza llamada la inercia.

Dicho de forma llana, la inercia es la resistencia que opone el objeto a detenerse. Poniendo un paralelismo, sería lo que “tira de nosotros” cuando nos lanzamos por una pendiente, intentamos parar en seco y nos resulta imposible sin precipitarnos hacia adelante.

Esa inercia puede afectar a su movimiento en un plano longitudinal (en la dirección de la marcha), transversal (perpendicular a la dirección de la marcha) o vertical (lo cual no siempre significa perpendicular al suelo).
En cualquier caso, hay que tener en cuenta que, como ocurre con las fuerzas, la inercia sólo sabe moverse en línea recta, por lo que no entiende de curvas. Dicho de otra forma, una inercia longitudinal excesiva al inicio de una curva es una mala compañera de viaje y si es transversal, también.
La inercia será mayor cuanto mayor sea la energía cinética que acumule el vehículo en movimiento. Y esta energía depende de la masa del vehículo y la velocidad a la que se desplace.

La fórmula define que cuanto más pesa un vehículo (cuanto mayor es su masa), más energía cinética acumula y cuanto mayor es su velocidad, mucho mayor es esa energía cinética.
El hecho de que la velocidad se multiplique por sí misma (en la fórmula aparece elevada al cuadrado) indica que cuando este factor aumenta se disparará la cantidad de energía cinética que acumule el vehículo.

$$\mathrm{E_c} = \frac{1}{2}.\,\mathrm{m.v^2}$$

El momento de inercia y la resistencia a la rodadura son dos de los aspectos de la dinámica que más se prestan a los "mitos" y a las ideas preconcebidas.

Refiriéndonos al momento de inercia, cuando estudiamos movimientos circulares, como el de la rueda tenemos que "traducir" todas las ecuaciones lineares en angulares, Cuanto mayor sea la velocidad o menor el radio de la rueda, más rápido girará ésta.

$$v = \omega r \rightarrow \omega = \frac{v}{r}$$
$$I = kmr^2$$
$$Ec_t = \tfrac{1}{2}mv^2 \rightarrow Ec_r = \tfrac{1}{2}I\omega^2$$

La Ec de las ruedas tiene dos componentes, una lineal debido a su traslación que funciona igual que la masa del chasis más otra rotacional debido al giro alrededor de su eje.

$$Ec_t = \tfrac{1}{2}mv^2$$

$$Ec_r = \frac{1}{2} I\omega^2$$

En cuanto a la resistencia a la rodadura, una de las tres resistencias fundamentales que se oponen al avance de un vehículo, la podemos considerar como una fuerza en sentido contrario al avance y que se opone al mismo, incrementando su magnitud a medida que aumentamos la velocidad.

Hablando de la resistencia a la rodadura, cabe nombrar la fuerza que se opone al avance físico, la fuerza de rozamiento; la fuerza de rozamiento estática determina la fuerza mínima necesaria para poner en movimiento un cuerpo de manera que existe un valor mínimo de fuerza a aplicar para que esto ocurra. Eso se debe a que existe una fuerza de rozamiento que se opone al inicio del movimiento.

Conviene tener en cuenta que la fuerza de rozamiento no se relaciona con variaciones de ningún tipo de energía potencial, es decir, no es una fuerza conservativa.

$$fr = \mu N$$

$$fr = \rho N$$

Vemos que, la diferencia entre rozamiento y deslizamiento es muy sutil, así, en lo único que se diferencian las ecuaciones anteriores es en los coeficientes; la primera ecuación hace referencia a la fuerza de rozamiento con el coeficiente de rozamiento (μ) y la segunda, a la fuerza de deslizamiento o rodadura utilizando el coeficiente de deslizamiento (*ρ)*.

En definitiva la fuerza que se opone al avance del vehículo la se llamará (Fr), y la relación entre esta fuerza y la carga normal a la superficie de rodadura (P) aplicada a la rueda se denomina conoce como coeficiente de resistencia a la rodadura.

$$fr = \frac{F_r}{P}$$

Sabemos que el mecanismo de compresión y expansión del neumático genera una fuerza que se opone al giro de la rueda y que se localiza ligeramente por delante del centro de la huella. Es importante comprender que no es un rozamiento sino una fuerza física.

Para disminuirla deberemos disminuir la cantidad de deformación de la huella, la energía disipada en la compresión-expansión o la distancia entre el punto de aplicación de la fuerza y el centro de la huella, ahí es donde influye la longitud del diámetro de la rueda a la hora de mejorar las prestaciones, de manera que, un diámetro mayor nos otorga una resistencia a la rodadura menor, sin embargo suele implicar un mayor momento de inercia.

Pero esto no influye en la energía cinética de rotación (Ecr) ya que, aunque una rueda de mayor diámetro tiene un mayor momento de inercia, su velocidad angular es menor (gira menos veces por minuto).

Concluimos que la Ecr es independiente del radio de la rueda, por ello, parece razonable acercarse al diámetro de rueda máximo permitido por el reglamento (siempre que la resistencia aerodinámica no aumente demasiado) pero cuidando la distribución de la masa dentro de la misma.

$$Ec_r = \frac{1}{2} I\omega^2 = \frac{1}{2}\left[\frac{1}{2} kmr^2\right]\left(\frac{v}{r}\right)^2 = \frac{1}{4} kmr^2 \frac{v^2}{r^2} = \frac{1}{4} kmv^2$$

Como hemos estado demostrando anteriormente respecto a todo el sistema interno del vehículo y, más concretamente, en referencia a las ruedas motrices y motor propio.

Es evidente que cada uno de los anteriores órganos que componen el sistema de transmisión de un vehículo se produce pérdidas debidas al rozamiento entre piezas y a otras causas, que hacen que la potencia final transmitida a las ruedas motrices sea menor que la potencia generada en el propio motor.

En este sentido se define el rendimiento de la transmisión (ηt) a la relación entre la potencia finalmente transmitida por las ruedas motrices (Pr) y la potencia que transmite el motor (Pm), o lo que es lo mismo;

$$\eta = \frac{Pr}{Pm}$$

Considerar un rendimiento medio para una transmisión del 85% suele ser una buena aproximación para la mayoría de los casos.

Refiriéndonos al frenado o parada del motor, hay que tener en cuenta que la energía ni se crea ni se destruye, sino que se transforma, por eso, para que un vehículo se detenga, habrá que transformar toda la energía cinética que haya acumulado al moverse.

Normalmente ésta energía se transforma en calor por efecto de la fricción de los elementos de frenado, por el rozamiento de las ruedas contra el asfalto y por el rozamiento de toda la carrocería contra el aire que la rodea, tanto seco como viscoso, desarrollando una potencia;

$$P_r = \vec{F}_r \cdot \vec{v}$$

Dado que la fuerza de rozamiento se opone a la velocidad relativa, esta potencia es negativa, esto es, disipa energía que, en el caso del rozamiento seco dinámico, es proporcional a la rapidez;

$$P_r = -\mu_d |\vec{F}_n| |\vec{v}|$$

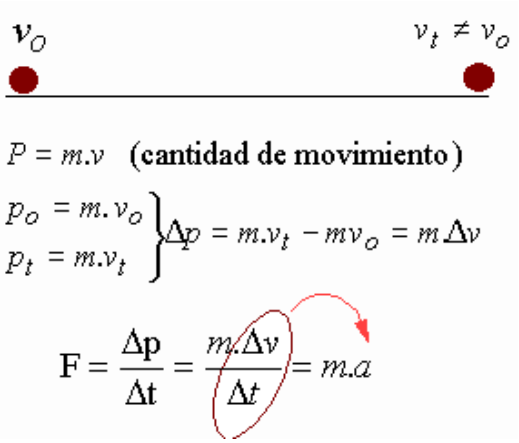
Esta energía disipada se transmite como calor, aumentando la temperatura de las superficies de contacto y llegando en ocasiones a su fusión lo que se conoce como “gripado” de los motores, por ello es importante reducir la fricción en máquinas y mecanismos.

Al hablar de una partícula en movimiento, ya sea uniforme o uniformemente acelerado, no podemos olvidar que esto conlleva a la generación de un momento lineal, ya que, esta partícula, el vehículo, ha adquirido la velocidad debido a su masa, de esta manera si la fuerza resultante de todas las que actúan sobre un cuerpo es nula el momento lineal del mismo permanece constante (otra forma de enunciar el principio de la inercia).
Es una magnitud vectorial proporcional a la masa y a la velocidad del objeto. Partiendo de esta definición y aplicando la ley fundamental de la mecánica de Newton, las variaciones de cantidad de movimiento se expresan en función de la fuerza resultante y el intervalo de tiempo durante el cual se ejerce ésta.

$$\vec{p} = m\vec{v}$$

Relacionada estrechamente con el momento lineal o cantidad de movimiento, se denomina impulso lineal a la cantidad de variación en el momento lineal que experimenta un objeto físico en un sistema cerrado y viene a representar una magnitud física que interviene en las acciones violentas o impactos, tales como choques. En este tipo de acciones conviene considerar la duración del impacto y la fuerza ejercida durante el mismo.

$$I = F\,\Delta t$$



En cuanto al propio sistema motriz del vehículo, hablaremos de la fuerza que éste genera, es decir, el torque, también conocido como par motor, que proporciona un empuje poderoso aún a bajas revoluciones.
El par motor o torque (T) es el producto vectorial de la fuerza aplicada (F) de empuje a los cilindros por la distancia (d) al eje geométrico de giro;

$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F}$$

$$\tau = rF \sin\theta,$$

Donde r es la distancia desde el eje de rotación de la partícula, F es la magnitud de la fuerza aplicada, y θ es el ángulo entre los vectores de posición y de fuerza.
El par de equilibrio en un cuerpo a lo largo del eje de rotación determina la tasa de cambio del cuerpo de momento angular;

$$\boldsymbol{\tau} = \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t}$$

Que para la rotación alrededor de un eje fijo será;

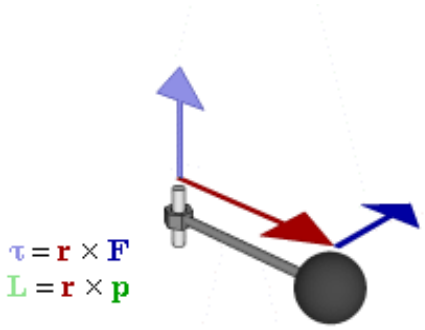
$$\mathbf{L} = I\boldsymbol{\omega},$$

Donde I es el momento de inercia y ω es la velocidad angular, además resulta que,

$$\boldsymbol{\tau}_{\mathrm{net}} = \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t} = \frac{\mathrm{d}(I\boldsymbol{\omega})}{\mathrm{d}t} = I\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t} = I\boldsymbol{\alpha},$$

Donde α es la aceleración angular del cuerpo. Además, un par de 1 N · m, aplicado a través de una revolución completa requerirá una energía de exactamente 2π julios, lo que es lo mismo, hablamos de la energía por vuelta completa del motor;

$$E = \tau\theta$$



Cuanto más alto sea el torque máximo y más bajo el número de revoluciones del motor al que se alcanza, más fuerza de empuje tendrá el vehículo; el motor se comporta más “elástico”, pudiéndose concluir que el torque es más importante para el desplazamiento del vehículo que la potencia misma.

La potencia desarrollada por el par motor es proporcional a la velocidad angular del eje de transmisión, viniendo dada por;

$$P = M\,\omega$$

Al desarrollar una potencia en un sistema mecánico, se desplaza al sistema a un punto de equilibrio entre el par motor y el par resistente, que es la oposición que el propio sistema ejerce al movimiento de sí mismo, es decir, la resistencia que debe ser vencida por el par motor. Esto vale para cualquier tipo de motor, sea asincrónico o no, sea eléctrico o de combustión interna.

Sabemos que, cuanta más energía acumula un vehículo, más espacio necesitará para transformar su energía cinética hasta detenerse. Y si sufre una colisión, los daños que experimente el vehículo serán mayores, puesto que la energía cinética se transformará de forma violenta mientras el vehículo reduce su velocidad de forma precipitada.
Cuando una partícula está bajo la acción de una fuerza conservativa, el trabajo de dicha fuerza es igual a la diferencia entre el valor inicial y final de la energía potencial.

$$\int_A^B \vec{F} \cdot d\vec{r} = E_{pA} - E_{pB}$$

Dicho de otra forma: el hecho de que un coche pese más que otro no garantiza una mayor seguridad, puesto que la masa del vehículo es uno de los factores determinantes en la acumulación de energía cinética. El otro, evidentemente, es la velocidad, y lo es en mayor medida. Esa garantía de seguridad vendrá dada en realidad por el dominio de la velocidad y por el diseño del vehículo, que influirá especialmente en la capacidad del automóvil para adherirse al suelo.

En el terreno de la adherencia hay un concepto útil para comprender las reacciones de un vehículo: el centro de gravedad. Se entiende como centro de gravedad el punto de aplicación de las fuerzas que actúan sobre un cuerpo. Cuanto más bajo esté localizado, mayor adherencia tendrá el vehículo sobre el terreno. Pero este centro de gravedad sólo es estable cuando el vehículo circula a velocidad constante y en línea recta. El centro de gravedad de un cuerpo viene dado por el único vector, por lo que al acelerar, al desacelerar y al girar el centro de gravedad se desplaza. Es lo que se denomina transferencia de masas.

$$M\mathbf{g}(\mathbf{r}_{\text{c.g.}}) = \int_V \mathbf{g}(\mathbf{r})\rho(\mathbf{r})dV$$

En un campo gravitatorio uniforme, es decir, uno en que el vector de campo gravitatorio es el mismo en todos los puntos, la definición anterior se reduce a la definición del centro de masas:

$$\mathbf{r}_{\mathrm{c.m.}} = \frac{1}{M}\int_V \mathbf{r}\,\rho(\mathbf{r})dV$$

Cuando aceleramos, el centro de gravedad se transfiere a la parte posterior del vehículo. La parte anterior se eleva y la posterior baja: es lo que se llama encabritado. Por contra, al frenar el coche experimenta un hundimiento por la parte frontal mientras que la parte posterior tiende a levantarse. Al girar, se aprecia un movimiento de balanceo: el vehículo se agacha por un lado y se eleva por el opuesto.
Si el coche ha acumulado mucha energía cinética, la transferencia de masas será brusca con el consiguiente riesgo de pérdida de adherencia.

Aun si el objeto está en rotación, el centro de masa se mueve como si fuera partícula.
Algunas veces el centro de masa se describe como si estuviera en el punto de equilibrio de un objeto sólido.

La segunda ley de Newton se aplica a un sistema cuando se usa el centro de masa.

$$\mathbf{F}_{\mathrm{net}} = \frac{\mathrm{d}(m\mathbf{v})}{\mathrm{d}t}$$

Si consideramos que la velocidad se mantiene constante, al igual que la masa ya es constante;

$$\mathbf{F} = m\mathbf{a}$$

Pero, ¿qué es la adherencia? Es la capacidad que tiene el vehículo de mantenerse en contacto con el suelo; es la relación entre el esfuerzo máximo que puede ser aplicado a la llanta sin patinaje y el peso P de la rueda.
De la adherencia dependerá que el vehículo disponga de una capacidad de tracción y direccionalidad sobre un terreno concreto. Y que el vehículo mantenga su adherencia vendrá condicionado por la masa y velocidad del vehículo, la calidad de los neumáticos y el estado del suelo. Cuando no se cumple esta inecuación la rueda desliza sobre el carril.

$$E \leq P\varphi$$

El coeficiente de adherencia se halla con la siguiente fórmula:

$$\varphi = \frac{0.24}{1 + 0.01v}$$

Hay que tener en cuenta que la adherencia se manifiesta en dos sentidos: longitudinal y transversal. La adherencia longitudinal funciona siempre a costa de la adherencia transversal, y viceversa. Cuando aceleramos o frenamos echamos mano de la adherencia longitudinal.
Cuando giramos, utilizamos la adherencia transversal.

Si empleamos toda la adherencia longitudinal, por ejemplo porque frenamos de forma brusca, nos quedaremos sin adherencia transversal y el vehículo no podrá girar aunque haya una curva. Si por contra utilizamos toda la adherencia transversal, el vehículo no podrá avanzar longitudinalmente siguiendo la carretera.

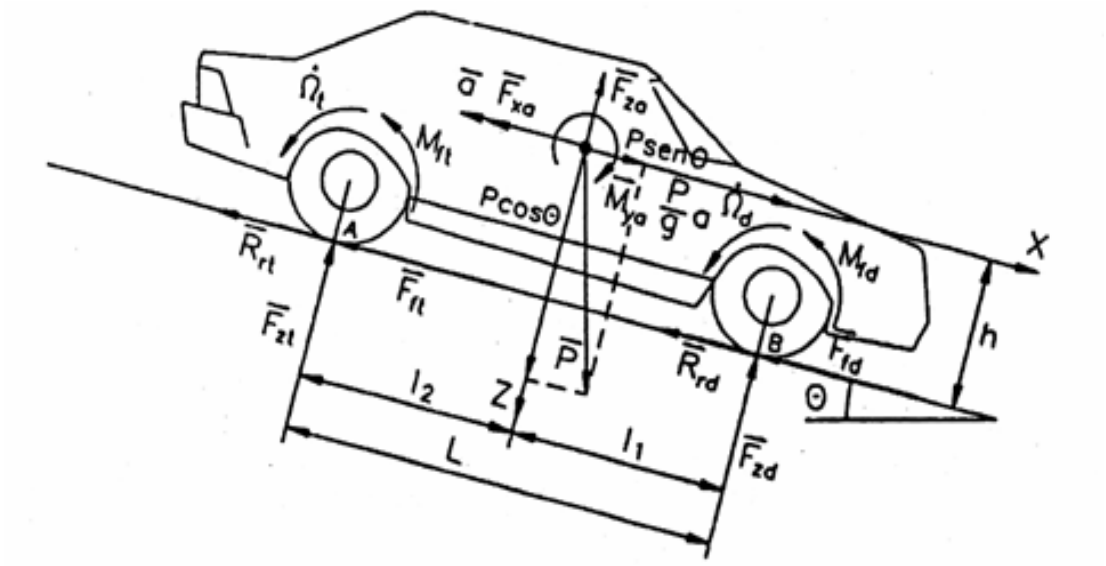


Fig.2.26.*Reparto de las fuerzas distribuidas en el vehículo en el proceso de frenado*

2.4. EL VEHÍCULO TELEDIRIGIDO

Con vehículo teledirigido o coche de radiocontrol hacemos referencia al objeto controlado a distancia.
Dependiendo de diferentes parámetros como la tipología de motor, tipo de escala, tipo de superficie e incluso según sus ruedas motrices, podremos clasificar al vehículo.

2.4.1. Tipos

- MOTOR
    - Explosión; coches de combustión. Según la escala del vehículo, la cilindrada variará desde 2,11cc a 23cc siendo alimentado el motor por gasolina, metanol o nitrometano.
    - Eléctrico; funcionan mediante corriente continua con voltajes que van de 7,2 V a 16V aproximandamente. El motor de continua puede ser con o sin escobillas siendo, esto últimos, mucho más eficientes.

- ESCALA
    - ¼, 1/5, 1/8, 1/10, 1/12 y 1/16 siendo 1/1 el tamaño real
- SUPERFICIE
    - Track u on-road; el circuito se hace en pista
    - Off-road; el circuito se encuentra sobre zona no asfaltada
    - Turismo
    - Rally

***Fig.2.27**.Jeep radiocontrolado de tipo off-road*

- RUEDAS MOTRICES

    - 2WD; tracción normal, es decir, tiene sólo dos ruedas motrices, las delanteras o las traseras

    - 4WD; tracción integral, esto es, las cuatro ruedas del vehículo son motrices

2.4.2. Componentes del vehículo

A nivel general un vehículo teledirigido tanto por combustión como eléctrico posee casi los mismos componentes a nivel estructural.

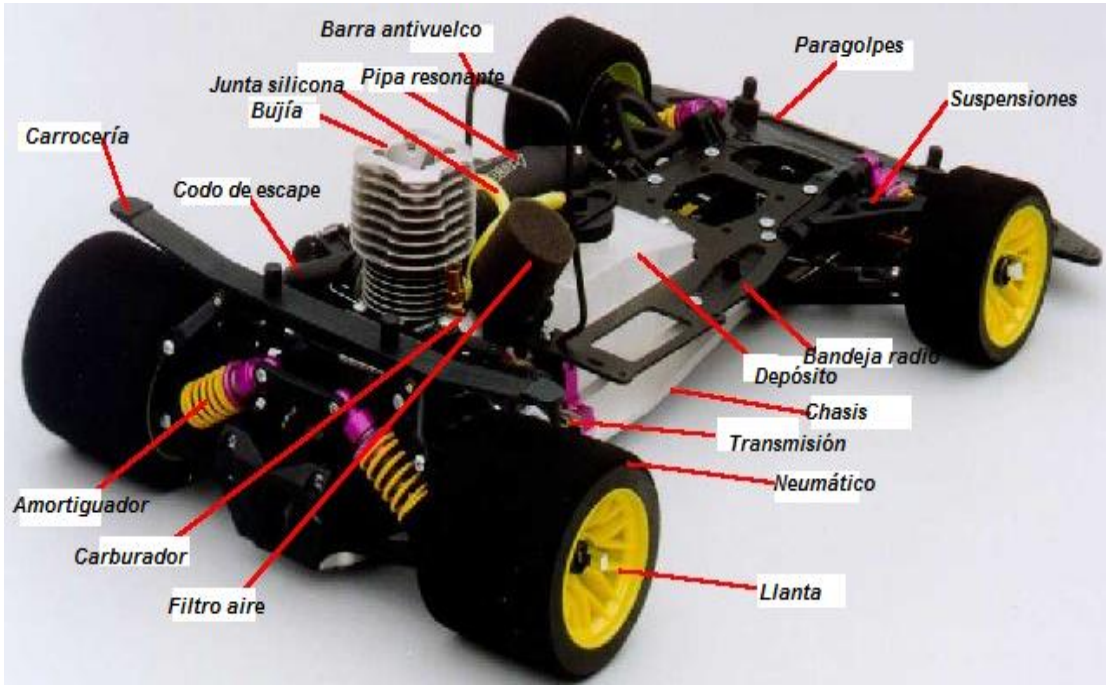


***Fig.2.28**. Diferentes componentes de vehículo radiocontrolado, en este caso, de combustión*

En el ejemplo de la fotografía a estudiar hemos tomado como referencia un vehículo por combustión; como vemos, tiene muchos componentes específicos relacionados con el motor de combustión como son la misma bujía, el depósito y el carburador entre otros.

Nos centraremos en los componentes que existen en común para cualquier vehículo de radiocontrol independientemente del tipo de motor por el que es regentado.

- **Chasis**

  Estructura interna que sostiene y aporta rigidez y forma a un vehículo u objeto en su construcción y uso; sostiene varias partes mecánicas como el motor o la suspensión.

Fig.2.29. *Chasis coche radiocontrol*

- **Transmisión**

  Encontramos transmisión de movimiento fuera del motor eléctrico en varias partes como;

  - Piñón-corona a la salida del motor:
    - Directo en el caso de motores eléctricos.
    - A través de embrague en el caso de motores de explosión.
    - Con doble piñón y corona si hay cambio de marchas.
  - A los ejes de rueda con tracción:
    - A través de correas en coches de dos diferenciales.
    - A través de palieres en coches de tres diferenciales.
    - A través de cadena en motos (eslabones de paso 6 mm o menos).
  - En los diferenciales (de piñones cónicos o rectos, de bolas, tipo Thorsen, etc).
  - A las ruedas de los coches con suspensiones por palieres o juntas "cardan".

- **Rodamientos**

  Elemento que reduce la fricción entre un eje y las piezas conectadas a éste por medio de rodadura, que le sirve de apoyo y facilita su desplazamiento.

Son de uso universal en todas las partes del coche y existen de diversas modalidades como las que veremos a continuación;

- Rodamientos de bolas. Son los más comunes. Además de por sus dimensiones (diámetros exterior e interior y anchura) se caracterizan según:
  - Blindaje: ninguno, simple o doble. Las tapas pueden ser metálicas o de goma.
  - Bolas: generalmente metálicas, aunque existen bolas cerámicas.
  - Jaula: generalmente metálica. El rodamiento principal en los motores de metanol suele ser de jaula fenólica.
- Rodamientos de jaula de agujas, empleado a veces en embragues.
- Rodamientos axiales, empleados en embragues tipo centax o en diferenciales de bolas.
- Rodamientos de un solo sentido ("one-way"). Están constituidos por una jaula de agujas; la jaula tiene un tallado especial que bloquea las agujas contra el eje sobre el que se apoyan si este gira en un sentido, y las deja libres en el otro. Se usan en cambios de marcha y en algunas transmisiones delanteras.

- **Suspensiones**

  Casi siempre independientes y en las cuatro ruedas, por trapecios en paralelogramo de brazos desiguales. También existen los coches carentes de suspensión.

- **Amortiguadores y muelles**

  Deben tener algún dispositivo para la compensación de volumen desplazado debido a la introducción del vástago, que casi siempre es un diafragma (el aire entre este diafragma y el extremo del amortiguador se comprime y compensa el volumen del vástago introducido). El aceite debe ser de silicona, graduada en cps, según su viscosidad, o en todo caso aceite que mantenga su viscosidad al subir su temperatura. Para amortiguador, el aceite de silicona varía entre 150 cps (fluido) a 600 cps (viscoso).

- **Barras estabilizadoras**

  Parte de la suspensión que permite solidarizar el movimiento vertical de las ruedas opuestas, minimizando la inclinación lateral que sufre la carrocería de un vehículo cuando se somete a la fuerza centrífuga, típicamente en las curvas.

- **Carrocería**

  En policarbonato ("Lexan") transparente, pintada por el interior con pintura especial para policarbonato.

- **Llantas**

  La llanta es la pieza, normalmente metálica, sobre la cual se asienta un neumático y que forma parte de la rueda (compuesta esta última por llanta y disco).

- **Neumáticos**

  De goma o espuma, según el tipo de coche.

- **Paragolpes**

  Delantero y, según el coche, trasero.

- **Anti vuelco**

  Como protección, que además sirve para agarrar el coche.

- **Alerón**

  Pieza que facilita el aerodinamismo del vehículo y su adherencia al terreno.

2.5. MANDO DE CONTROL

2.5.1. Tipos

En un sistema de control mando-vehículo intervienen, obviamente, el emisor y receptor.
En este caso, el receptor será el vehículo a controlar y el emisor, el mando controlador.

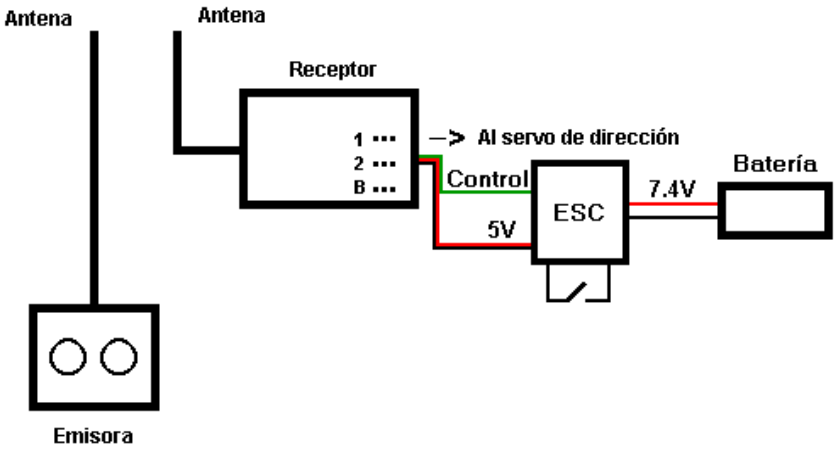


Fig.2.30. *Partes del equipo de radio en un coche con motor eléctrico*

Existen diferentes tipos de mando controlador, al igual que hay múltiples variedades de vehículos.
En este caso, existen más concretamente dos tipos de mandos según;

- EMISIÓN

- **AM**

  Actualmente en retirada en cuanto a vehículos radiocontrolados, pero perfecta para alcanzar largas distancias en aeromodelismo ya que a mayor frecuencia es mayor la interferencia y menor el radio de cobertura.

La frecuencia no es muy elevada; del orden de GHz. Tiene la desventaja de que puede ser que dos coches funcionen a la misma frecuencia y que choquen al perder el control del vehículo al recibir 2 o más señales válidas procedentes del mando de control del vehículo.

Para evitarlo las personas se ponen de acuerdo en que frecuencia utilizar, cambiando unos cristales de cuarzo del circuito resonante.

- **Spread Spectrum.**

  Este modo funciona emitiendo y recibiendo señales del orden del GHz.

  A pesar de que a estas frecuencias las interferencias ambientales son importantes, esta tecnología evita que nos preocupemos por el control individual del vehículo, de manera que no se solapen varios terminales.

  Se elimina también la necesidad de tener que cambiar los cristales, simplemente se enciende y el usuario se despreocupa ya que la frecuencia queda reservada.

  La desventaja teórica es que se consigue una distancia máxima de medio km.

Fig.2.31. *Mando con tecnología de emisión Spread Spectrum*

- ESTRUCTURA

- **Stick**

  A la izquierda el palito de acelerador-freno de manera que hacia adelante acelera, y hacia atrás frena, y a la derecha el de dirección, con movimiento derecha-izquierda.

- **De volante y gatillo**

  También denominado como emisora de pistola, donde se encuentra en el volante la dirección, y en el gatillo, el acelerador-freno.

***Fig.2.32**. Mando de stick (izda.) y pistola(dcha.)*

2.5.2. Componentes del mando de control

Los componentes del mando de control variarán en función del tipo de mando utilizado, es decir, dependerán de las características físicas diferentes; un mando de stick plano no tendrá los mismos componentes o disposición que un mando de pistola.

Aquí, se centrará la información en el mando de control tipo stick, ya que, como se explicará en capítulos posteriores, en principio, el mando, que será el módulo emisor, constará del Arduino Uno, de un joystick conectado al mismo, y del módulo de radio nRF24L01 para la posterior comunicación con el módulo receptor, el vehículo.

Así, hablaremos de forma general de las partes q componen un mando de stick, nuestro mando específico como módulo emisor será descrito minuciosamente en capítulos posteriores.

***Fig.2.33**. Partes mando stick*

1.**- Cubierta**: protege los circuitos internos el dispositivo y da estética al producto.
2**.- Palanca**: permiten el control de movimiento de los gráficos del juego en pantalla con varias direcciones.
3.- **Gatillo**: se usa para realizar el movimiento más común que es el disparo.
4.- **Botones superiores**: tienen funciones secundarias.
5.- **Botones inferiores**: tiene funciones primarias.
6.- **Cable**: transmite las coordenadas de movimientos y alimenta al dispositivo.

Inicialmente, estos dispositivos utilizaban el puerto llamado Gameport, el cual también podía ser utilizado para conectar dispositivos MIDI (lenguaje de comunicación de ciertos dispositivos musicales). El Gameport en el ámbito de la electrónica comercial, se le denomina conector DB15 ("D-subminiature type B, 15 pin"), esto es D-subminiatura tipo B, para 15 pines.

Posteriormente con el lanzamiento al mercado del puerto USB, el Gameport es reemplazado y el Joystick se adapta a la nueva tecnología USB, siendo este conector el que domina actualmente el mercado.

Fig. 2.34. *Conector DB-15 macho del "Gamepad" (izda.) conector DB-15 hembra de la tarjeta de sonido (dcha.)*

Fig. 2.35. *Conector USB macho del "Gamepad" y puertos USB hembra*

Siguiendo la tendencia de las consolas de videojuego, los Joystick Wireless son dispositivos de juego, que permiten ser utilizados sin necesidad de cables, utilizando una frecuencia de los 2.4 Ghz, con lo que se tiene libertad de utilizar el dispositivo sin la limitación del largo del cable hacia la computadora; sin embargo, tiene la limitación de la distancia, ya que se puede perder la recepción si se sobrepasa cierta distancia, mientras que otra desventaja es que requiere de alimentación basada en baterías, las cuáles hay que reemplazar al terminarse la vida útil de estas.
El modo en que se interconecta con la computadora, es por medio de un receptor USB, que se conecta al puerto de la computadora y detecta la señal emitida por el dispositivo de juego inalámbrico.

2.6. COMERCIALIZACIÓN Y FABRICANTES

2.6.1. Marcas internacionales

Existen gran variedad de marcas dentro del mundo del radiocontrol así como diferentes categorías y subcategorías con modificaciones y adaptaciones especializadas según la necesidad del consumidor.
Aquí haremos alusión a las principales comerciales a nivel internacional y explicaremos brevemente las funciones que las caracterizan.

- SERPENT

Es uno de los principales fabricantes de RC coches destinados a las competiciones ya que ha estado presente desde el inicio de las carreras desde hace unos 25 años.
Ha sido responsable de carreras con muchas victorias nacionales e internacionales, incluyendo títulos mundiales.
La empresa fue fundada oficialmente en 1980 por Pieter Bervoets y compañero Ron Ton.
Es conocida por su alta participación en el mercado de las carreras de RC, cooperando con IFMAR y otras federaciones internacionales y nacionales, patrocinando muchos de los eventos con más peso a nivel internacional.

La innovación de la marca Serpent radica en los coches con doble suspensión, amortiguadores de bienes, cajas de cambios de 2 velocidades, sistemas de 4 ruedas motrices, el ahora famoso embrague Centax (centrífuga-axial), el INS-Box, etc.
Inventó la clase 235mm y más tarde en las carreras iniciado 200mm 1/10 coches de escala.

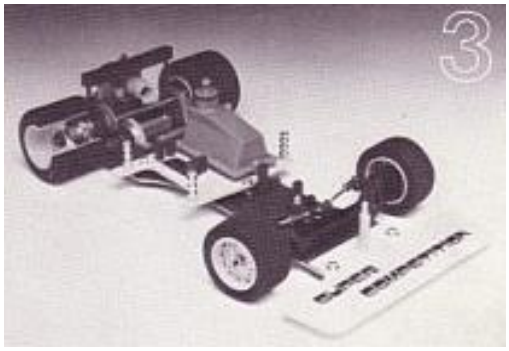


***Fig.2.36**. Serpent MK, 1981*

- KYOSHO

La compañía es uno de los fabricantes de automóviles modelo rc más antiguas de Japón, y produce una amplia variedad de productos, incluyendo coches teledirigidos, aviones, helicópteros y barcos.
Kyosho también produce coches modelo con características muy detalladas venidas de su proceso de fundición.
Su principal competidor en el mercado del automóvil RC es Tamiya.
Ahora tiene productos líderes en su categoría en motores eléctricos (cepillado y sin escobillas), Ni-MH y las baterías de Li-Po y cargadores de baterías.

***Fig.2.37.** Scorpion Kyosho*

- XRAY

Aunque se estableció en 2000 por Hudy Juraj, su fundador ha tenido experiencias largas y exitosas en el diseño de los coches modelo de RC de carreras. Entre las diferentes innovaciones, se destacan el resorte de acero, así como partes de duraluminio (chasis), y materiales compuestos de grafito.

Fig.2.38. *XRay, T4*

- MUGEN SEIKI

Es un fabricante japonés de coches controlados por radio con sede en Funabashi, Chiba, Japón. En 1990, Mugen Seiki lanzó su primer coche campo a través, el Supersport. La compañía fue establecida en 1998 e incorporada en California.
Entre las muchas modificaciones y mejoras de sus coches radiocontrolados se distinguen las piezas de grafito utilizados en la placa superior de dirección, torres de choque, y las aletas de dirección delanteras así como el montaje de un nuevo motor de una sola pieza y más viajes de dirección.

Fig.2.39. *Mugen Seiki MBX7 Eco*

- TEAM LOSI

Es una compañía originaria de EE.UU especializada en el coche radiocontrolado. Fue fundada por Gil Losi a finales de la década de 1980. Por supuesto, esta empresa fue pionera en muchos de los avances tecnológicos que han llegado hasta hoy en día; a partir de sus únicos 5-link brazos de suspensión trasera, chasis de carbono y grafito, y los neumáticos de goma natural, el primer vehículo que lanzaron, el buggy de carreras JRX2, sentará las bases para el avance tecnológico y la innovación.

Fig.2.40. *Losi Mini Desert Buggy 1/18 RTR*

- VBC

VBC Racing es una empresa joven, fundada en 2011 y con sede en Vancouver; se inspira en los kits de conversión sobre el chasis chasis Tamiya y XRay aunque recientemente introdujo su primer kit completo de clase de turismos Wildfire 1 / eléctrico.
Aunque tiene lleva poco tiempo en el mundo del radiocontrol, ya se ha abierto paso en el ámbito de la competición profesional colocándose como una de las mejores marcas de coche gracias a la optimización de sus prestaciones de cara a la carrera de competición.

Fig.2.41*. VBC Firebolt DM 1/10 2WD*

2.6.2. Fabricantes en España

A continuación, algunos de los fabricantes de marca España impulsores del mundo del automovilismo y pioneros en los vehículos de radiocontrol.

- **BYCMO**

Bycmo, empresa dedicada desde 1980 a la creación, fabricación, importación, exportación, distribución y venta de modelismo radiocontrolado. Bycmo nació en la trastienda de un comercio dedicado a la venta de modelismo en Esplugues de Llobregat en octubre de 1980. Desde aquellos años ha ido creciendo, evolucionando y marcando tendencias sin perder nunca de vista los objetivos. En 1986, Bycmo se proclamamó Campeoón de España de coches RC de explosión (gasolina) en la escala 1/8 Buggy.

A partir de esta victoria, los bólidos Bycmo estuvieron a la cabeza de la alta competición española durante cinco años consecutivos, en pugna con las principales marcas internacionales.

Desde ese momento, Bycmo gozó del reconocimiento que le otorgaron las victorias en los Campeonatos de España de 1986-87-88-89-90. A este palmarés se ha de añadir, que la firma ha ganado en dos ocasiones el Campeonato Nacional de Suiza, el Campeonato Nacional de Alemania, el Subcampeonato de Dinamarca y en otras dos ocasiones se ha alzado con la victoria en el Campeonato Nacional de Argentina.

También en 1986, Bycmo lanzó, por primera vez en el mundo, un coche radiodirigido (teledirigido) a escala 1/5 con motor de gasolina de 24c.c. Con este novedoso vehículo se consiguió durante cuatro ediciones el Campeonato de España, lograron ser los primeros en la Copa Europea y ganaron tres Campeonatos de Argentina consecutivos.

Desde sus inicios Bycmo ha tenido como uno de sus principios ser una de las marcas de referencia en el ámbito del automodelismo.

Con este objetivo ha participado desde 1981 en las ferias europeas más importantes del sector como la Spielwarenmesse International Toy Fair Nürnberg (Nuremberg), la Mondial de la Maquette et du Modele Reduit (París) y el Salón del Hobby Ciudad de Barcelona (Barcelona).

- **GUILOY**

Ubicada en uno de los centros productivos del juguete más importantes de Europa, concretamente en la localidad de Ibi (Alicante), Guiloy es una de aquellas marcas que se ha convertido en una marca imprescindible a la hora de citar empresas dedicadas a las miniaturas.

Esto se debe no solamente a ser una de las pocas marcas de miniaturas nacionales que existen en la actualidad, sino que además por méritos propios y su gran calidad a la hora de fabricar productos, siendo una auténtica especialista en las escalas "grandes" y sobre todo a la hora de reproducir maquetas de motocicletas.

Aunque nació bajo otra denominación, la empresa que en un futuro se conocería por Guiloy hace su aparición en el año 1969 por iniciativa de cuatro socios fundadores, centrando sus actividades como empresa auxiliar de varios sectores, entre ellos el del juguete. Inicialmente esta primitiva empresa se dedicaría única y exclusivamente a crear matricería, fabricando más tarde sus propias piezas inyectadas en metal y plástico a partir de estos moldes.

Observando los socios fundadores que la producción de juguetes podía ser una excelente salida para la marca, estas cuatro personas decidieron fabricar su propia línea de juguetes, primero a tiempo parcial para posteriormente dedicarse exclusivamente a esta actividad y creándose ya sí la marca Guiloy, S.A. en el año 1973.

Ya desde sus inicios Guiloy apostó por la innovación y la calidad en sus productos, siendo esta una de las premisas fundamentales de la marca, la cual afortunadamente sería reconocida por sus clientes.

Tanto es esta calidad que incluso durante tres años consecutivos (1995, 1996 y 1997) la empresa consiguió el premio "Miniatura del año" que anualmente otorga la Feria Internacional del Juguete de Nuremberg, uno de los eventos más importantes en el mundo del juguete, por no decir el más importante.

Estas miniaturas, se convertirían más tardes en carrocerías de coche radiocontroladores importadas por otras grandes empresas del mundo del vehículo teledirigido, incluso las beneficiarias del mundo de las competiciones a nivel nacional e internacional.

- **NINCO**

Es una empresa dedicada al hobby. Fabrica y distribuye productos destinados al ocio de niños y adultos de todo el mundo. Es el primer fabricante de coches y pistas de slot de España y comercializa sus productos bajo la marca NINCO en más de 30 países.

También comercializa sus propias marcas de radio control: NINCO4RC (coches), NINCOAIR (helicópteros y aviones) y NINCOCEAN (barcos), todas ellas con un perfil muy definido de usuario, el hobbysta: desde la iniciación hasta el perfeccionmiento.
Además, con la enseña NINCO HOBBY distribuye en España diversas marcas internacionales de juguete y hobby.
NINCO DESARROLLOS fue fundada en 1993 por Eduard Nin y Eladio Cosculluela (NIN+CO).

El primer proyecto de la empresa fue la réplica a escala del Renault Clio 16V. Este coche rompía todos los estándares de la época, fijando su objetivo en el aficionado hobbista, adulto y experto. El éxito de la gama de coches que siguieron a este primer modelo culminó en 1997 con la aparición de las pistas NINCO. Por su diseño, prestaciones y facilidad de uso están consideradas las mejores pistas de slot del mundo.

En el año 2006, incorpora a sus actividades como empresa la distribución en España de marcas internacionales de hobby, que están presentes en más de 400 puntos de venta del país. La distribuidora adopta el nombre de NINCO HOBBY. Desde 2009 desarrolla y comercializa sus propias marcas de radio control.

- **FASOL**

FASOL. S.A. es una empresa creada en 1982 en un pequeño almacén en Manresa, provincia de Barcelona, España, y desde 2006 ubicada en una nave industrial en Santpedor con nuevas, amplias y modernas instalaciones.

Desde los inicios fuimos importadores de productos de R/C para distribuir en España y Portugal las mejores marcas del mercado y así satisfacer al público de R/C con productos de calidad a precios competitivos.

Entre estos artículos disponemos de varias gamas para así cubrir diversas necesidades: emisoras, baterías, servos, motores, receptores, simuladores, coches, aviones, barcos, helicópteros y montajes en madera.

Actualmente la empresa cuenta con más de 25 años en el sector, un equipo profesional con experiencia y una clientela formada por las mejores tiendas del sector suministrando nuestro producto a través de un canal de distribución ágil y eficaz.

CAPÍTULO 3

ANÁLISIS Y ARQUITECTURA HARDWARE DE CONTROL

En este capítulo trataremos ,entre otros muchos aspectos, temáticas pertenecientes al hardware general y particular usado en el diseño del proyecto, sus funciones y características así como los diferentes tipos y las razones prácticas de la elección del mismo. Posteriormente, se definirán, con mayor profundidad, cada uno de sus elementos y se realizará un estudio sobre cada uno de ellos señalando las ventajas sobre la amplia variedad disponible en el mercado.

3.1. SISTEMA DE PROCESAMIENTO

Mediante el sistema de procesamiento generamos información logrando sobre los datos algún tipo de transformación de manera que, esa transformación, convierte al dato en información.
La información encriptada en dato, pasará por cuatro subprocesos constantes e invariables; ingreso (dato), memorización, proceso y salida (información).

3.1.1 Microcontrolador

Para el gobierno de uno o varios procesos se emplea el denominado microprocesador.
Es un circuito integrado que en su interior contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos. Estas partes están interconectadas dentro del microcontrolador.

Todo microcontrolador requiere de un programa para que realice una función específica. Este se almacena normalmente en la memoria ROM, sin un programa, los microcontroladores carecen de utilidad.

El propósito fundamental de los microcontroladores es el de leer y ejecutar los programas que el usuario le escribe, es por esto que la programación es una actividad básica e indispensable cuando se diseñan circuitos y sistemas que los incluyan. El carácter programable de los microcontroladores simplifica el diseño de circuitos electrónicos permitiendo modularidad y flexibilidad, ya que un mismo circuito se puede utilizar para que realice diferentes funciones con solo cambiar el programa del microcontrolador.

Los microcontroladores están diseñados para interpretar y procesar datos e instrucciones en forma binaria; patrones de 1's y 0's conforman el lenguaje máquina de los microcontroladores, y es lo único que son capaces de entender. Estos 1's y 0's representan la unidad mínima de información, conocida como bit, ya que solo puede adoptar uno de dos valores posibles: 0 ó 1.

La representación de datos, instrucciones y señales en forma de bits resulta dificultosa y tediosa para aquellas personas que no estén familiarizadas con el sistema de numeración binario o de bajo nivel debido a que las instrucciones no son propias del lenguaje humano.

Es por esto que la programación comúnmente se lleva a cabo en un lenguaje de alto nivel, es decir, un lenguaje que utilice frases o palabras semejantes o propias del lenguaje humano.

Las sentencias de los lenguajes de alto nivel facilitan enormemente la programación ya que son familiares a nuestra manera de comunicarnos. Lenguajes como el C o BASIC son comúnmente utilizados en la programación de microcontroladores.

Otro tipo de lenguaje más especializado es el lenguaje ensamblador. El lenguaje ensamblador es una lista con un limitado número instrucciones a los cuales puede responder un microcontrolador. Estas instrucciones son palabras o abreviaciones que representan las instrucciones en lenguaje máquina del microcontrolador.

Las instrucciones en lenguaje ensamblador, también conocidas como nemotécnicos, son fáciles de entender y permiten operar directamente con los registros de memoria así como con las instrucciones intrínsecas del microcontrolador. Es por esto que el lenguaje ensamblador es el lenguaje por excelencia en la programación de microcontroladores, ya que permite hacer un uso eficiente de la memoria y minimizar el tiempo de ejecución de un programa.

Cualquiera que sea el lenguaje que se utilice en la programación de microcontroladores, es de lo más recomendable profundizar en su arquitectura interna, ya que con este conocimiento se pueden aprovechar más y mejor las capacidades de un microcontrolador.

De fábrica, la memoria ROM del microcontrolador no posee datos. Para que pueda controlar algún proceso es necesario generar o crear y luego grabar en la memoria EEPROM algún programa, el cual puede ser escrito en lenguaje ensamblador u otro lenguaje para microcontroladores.

La principal diferencia frente a una unidad central de procesamiento normal, es que un microcontrolador es más fácil convertirlo en una computadora en funcionamiento, con un mínimo de circuitos integrados externos de apoyo.

Existen varios fabricantes de microcontroladores tales como Texas Instruments, Motorola, Atmel, Intel, Microchip, Toshiba, Nacional, etc. Todos ellos ofrecen microcontroladores con características más o menos similares, sin embargo, en términos generales, se puede decir que todos sirven para lo mismo: leer y ejecutar los programas del usuario.

El microcontrolador utilizado en este proyecto es el ATMega328P de Atmel

Fig.3.1. *ATMega 328P*

Al estar todos los microcontroladores integrados en un chip, su estructura fundamental y sus características básicas son similares. Todos deben disponer de los bloques esenciales: procesador (CPU), memoria de datos (RAM) y de instrucciones (ROM), líneas de entrada/salida, oscilador de reloj y diversos módulos para de periféricos.

3.1.1.1. ARQUITECTURA BÁSICA

Aunque inicialmente todos los microcontroladores adoptaron la arquitectura clásica de Von Neumann, en la actualidad es más utilizada la arquitectura Harvard.

- VON NEUMANN

  Tradicionalmente los sistemas con microprocesadores se basan en esta arquitectura, en la cual la unidad central de proceso (CPU), está conectada a una memoria principal única (casi siempre sólo RAM) donde se guardan las instrucciones del programa y los datos. A dicha memoria se accede a través de un sistema de buses único (control, direcciones y datos).

  La arquitectura de Von Neumann podría definirse en términos generales como cualquier computador de programa almacenado en el cual no pueden ocurrir una extracción de instrucción y una operación de datos al mismo tiempo, ya que comparten un bus en común.
  Tiene un conjunto dedicado de direcciones y buses de datos para leer datos desde memoria y escribir datos en la misma, y otro conjunto de direcciones y buses de datos para ir a buscar instrucciones.

  El tamaño de la unidad de datos o instrucciones está fijado por el ancho del bus que comunica la memoria con la CPU pero el tener un único bus hace que el microprocesador sea más lento en su respuesta, ya que no puede buscar en memoria una nueva instrucción mientras no finalicen las transferencias de datos de la instrucción anterior.

- HARVARD

  Las partes principales de las computadoras con arquitectura Harvard son la memoria y la CPU, la primera guarda los datos y la CPU los procesa.
  Es una arquitectura de computadora con pistas de almacenamiento y de señal físicamente separadas para las instrucciones y para los datos.

  A través de la memoria no solo se pueden manejar los datos sino también el lugar donde se encuentran almacenados, estos dos parámetros son de mucha importancia para la CPU.

  La CPU trabaja con mucha mayor velocidad que las memorias con las que trabaja. Para que la memoria vaya más rápida se aconseja suministrar una pequeña memoria llamada caché que es de acceso rápido aunque también se pueden conseguir memorias con más velocidad pero estas poseen un alto precio.
  Si los datos están en la caché rendirán mucho más tiempo, pero si la caché tiene que obtener los datos a través de la memoria principal estos no perduraran mucho.

  Una de las memorias contiene solamente las instrucciones del programa (Memoria de Programa), y la otra sólo almacena datos (Memoria de Datos).

También la longitud de los datos y las instrucciones puede ser distinta, lo que optimiza el uso de la memoria en general. Para un procesador de Set de Instrucciones Reducido, o RISC (Reduced Instruction Set Computer), el set de instrucciones y el bus de memoria de programa pueden diseñarse de tal manera que todas las instrucciones tengan una sola posición de memoria de programa de longitud.

La arquitectura Harvard permite que los datos y las instrucciones se almacenen en chaches separados para obtener mejor rendimiento. Se utiliza en procesadores de señal digital y en DSPs, que son utilizados en productos para procedimiento de video y audio.

La ventaja fundamental de la arquitectura Harvard es que permite adecuar el tamaño de los buses a las características de cada tipo de memoria y el procesador puede acceder a cada una de ellas de forma simultánea, lo que se traduce en un aumento significativo de la velocidad de procesamiento.

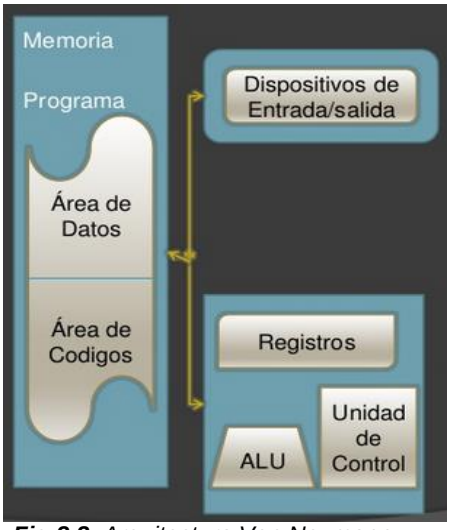


Fig.3.2. *Arquitectura Von Neumann*

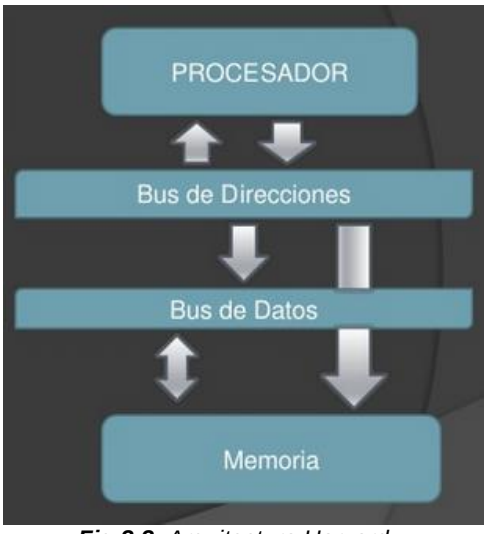


Fig.3.3. *Arquitectura Harvard*

3.1.1.2. PROCESADOR

Es el elemento más importante del microcontrolador y determina sus principales características, tanto a nivel hardware como software.

Es el encargado de realizar diversas funciones, como direccionar la memoria de instrucciones, recibir el código OP de la instrucción en curso, su decodificación y la ejecución de la operación que implica la instrucción, así como la búsqueda de los operandos y el almacenamiento del resultado.

Existen tres tipos básicos de repertorios de instrucciones que determinan la arquitectura del procesador:

- ***CISC***

Denominado “Complex instruction set computing” o Computadores de Juego de Instrucciones Complejo, disponen de un conjunto de instrucciones que se caracteriza por ser muy amplio y permiten realizar operaciones complejas entre operandos situados en la memoria o en los registros internos.
La principal ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúancomo macros.

- ***RISC***

Del acrónimo “Reduced Instruction Set Computer” o Computadores de Juego de Instrucciones Reducido donde el repertorio de instrucciones máquina es muy reducido y las instrucciones son simples y generalmente se ejecutan en un ciclo. Esto permite la optimización del hardware y el software del procesador.

- ***SISC***

Nombrado “Specific Instruction Set Computer” o Computadores de Juego de Instrucciones Específico en el que los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido, es específico, es decir, las instrucciones se adaptan a las necesidades de la aplicación prevista.

3.1.1.3. MEMORIA INTERNA

En los microcontroladores la memoria de instrucciones y datos está integrada en el propio chip. Una parte debe ser no volátil, tipo ROM, y se destina a contener el programa de instrucciones que gobierna la aplicación. Otra parte de memoria será tipo RAM, volátil, y se destina a guardar las variables y los datos.

Hay dos peculiaridades que diferencian a los microcontroladores de los PC's:

- No existen sistemas de almacenamiento masivo como disco duro o disquetes.
- Como el microcontrolador sólo se destina a una tarea en la memoria de programa, sólo hay que almacenar un único programa de trabajo.

La memoria de datos (RAM) en estos dispositivos es de poca capacidad pues sólo debe contener las variables y los cambios de información que se produzcan en el transcurso del programa. Por otra parte, como sólo existe un programa activo, no se requiere guardar una copia del mismo en la RAM pues se ejecuta directamente desde la memoria de programa (ROM).

El usuario de PC está habituados a manejar Megabytes de memoria, pero los diseñadores con microcontroladores trabajan con capacidades de memoria de programa de 512 bytes, 1K, 2K (hasta unos 64K) y de RAM de 20 bytes, 68 bytes, 512 bytes (hasta unos 4K).

Según el tipo de memoria de programa que dispongan los microcontroladores, la aplicación y utilización de los mismos es diferente. Se describen las cinco versiones de memoria no volátil que se pueden encontrar en los microcontroladores del mercado:

- **ROM con máscara**

Es una memoria no volátil de sólo lectura cuyo contenido se graba durante la fabricación del chip.

Máscara viene de la forma cómo se fabrican los circuitos integrados. Estos se fabrican en obleas que contienen varias decenas de chips.

Estas obleas se obtienen a partir de procesos fotoquímicos, donde se impregnan capas de silicio y oxido de silicio, y según convenga, se erosionan al exponerlos a la luz. Como no todos los puntos han de ser erosionados, se sitúa entre la luz y la oblea una máscara con agujeros, de manera que donde deba incidir la luz, esta pasará.

Con varios procesos similares pero más complicados se consigue fabricar los transistores y diodos que componen un circuito integrado.

El elevado coste del diseño de la máscara sólo hace aconsejable el empleo de los microcontroladores con este tipo de memoria cuando se precisan cantidades superiores a varios miles de unidades.

- **OTP**

El microcontrolador contiene una memoria no volátil de sólo lectura "programable una sola vez" por el usuario. OTP (One Time Programmable). Es el usuario quien puede escribir el programa en el chip mediante un sencillo grabador controlado por un programa desde un PC. La versión OTP es recomendable cuando es muy corto el ciclo de diseño del producto, o bien, en la construcción de prototipos y series muy pequeñas.

Tanto en este tipo de memoria como en la EPROM, se suele usar la encriptación mediante fusibles para proteger el código contenido.

- **EPROM**

Los microcontroladores que disponen de memoria EPROM (Erasable Programmable Read Only Memory) pueden borrarse y grabarse muchas veces. La grabación se realiza, como en el caso de los OTP, con un grabador gobernado desde un PC. Si, posteriormente, se desea borrar el contenido, disponen de una ventana de cristal en su superficie por la que se somete a la EPROM a rayos ultravioleta durante varios minutos. Las cápsulas son de material cerámico y son más caros que los microcontroladores con memoria OTP que están hechos con material plástico. Hoy día se utilizan poco, siendo sustituidas por memorias EEPROM o Flash.

- **EEPROM**

Se trata de memorias de sólo lectura, programables y borrables eléctricamente EEPROM (Electrical Erasable Programmable Read Only Memory). Tanto la programación como el borrado, se realizan eléctricamente desde el propio grabador y bajo el control programado de un PC. Es muy cómoda y rápida la operación de grabado y la de borrado. No disponen de ventana de cristal en la superficie.

Los microcontroladores dotados de memoria EEPROM una vez instalados en el circuito, pueden grabarse y borrarse cuantas veces se quiera sin ser retirados de dicho circuito. Para ello se usan "grabadores en circuito" que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo.

El número de veces que puede grabarse y borrarse una memoria EEPROM es finito, por lo que no es recomendable una reprogramación continua. Hoy día están siendo sustituidas por memorias de tipo Flash. Se va extendiendo en los fabricantes la tendencia de incluir una pequeña zona de memoria EEPROM en los circuitos programables para guardar y modificar cómodamente una serie de parámetros que adecuan el dispositivo a las condiciones del entorno. Este tipo de memoria es relativamente lenta.

- **FLASH**

Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una ROM y una RAM pero consume menos y es más pequeña. A diferencia de la ROM, la memoria FLASH es programable en el circuito. Es más rápida y de mayor densidad que la EEPROM.

La alternativa FLASH está recomendada frente a la EEPROM cuando se precisa gran cantidad de memoria de programa no volátil. Es más veloz y tolera más ciclos de escritura/borrado. Son idóneas para la enseñanza y la Ingeniería de diseño.

Las memorias EEPROM y FLASH son muy útiles al permitir que los microcontroladores que las incorporan puedan ser reprogramados "en circuito", es decir, sin tener que sacar el circuito integrado de la tarjeta. Así, un dispositivo con este tipo de memoria incorporado al control del motor de un automóvil permite que pueda modificarse el programa durante la rutina de mantenimiento periódico, compensando los desgastes y otros factores tales como la compresión, la instalación de nuevas piezas, etc.

La reprogramación del microcontrolador puede convertirse en una labor rutinaria dentro de la puesta a punto.

3.1.1.4. PUERTOS E/S

La principal utilidad de las patillas que posee la cápsula que contiene un microcontrolador es soportar las líneas de E/S que comunican al computador interno con los periféricos exteriores y según los controladores de periféricos que posea cada modelo de microcontrolador, se destinan a proporcionar el soporte a las señales de entrada, salida y control.

Todos los microcontroladores destinan algunas de sus patillas a soportar líneas de E/S de tipo digital, esto es, todo o nada. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertos. Las líneas digitales de los Puertos pueden configurarse como Entrada o como Salida cargando un 1 ó un 0 en el bit correspondiente de un registro destinado a su configuración.

Los puertos de entrada y salida o puertos E/S, se agrupan generalmente en conjuntos de 8 bits de longitud, que permiten leer datos del exterior o escribir en ellos desde el interior del microcontrolador, el destino habitual es el trabajo con dispositivos simples como relés, leds o motores.

Algunos puertos de también E/S tienen características especiales que le permiten manejar salidas con determinados requerimientos de corriente o incorporar mecanismos especiales de interrupción para el procesador. Normalmente, cualquier pin de E/S puede ser considerado como E/S de propósito general, compartiendo los pines con otros periféricos.

3.1.1.5. INTERRUPCIONES

Las interrupciones son esencialmente llamadas a subrutina generadas por los dispositivos físicos, al contrario de las subrutinas normales de un programa en ejecución.

Es como un subprograma, pero puede ser llamado vía una interrupción por hardware y detiene al programa principal en cualquier ejecución permitiéndole luego retornar a la labor que se estaba ejecutando. Como el salto de subrutina no es parte del hilo o secuencia de ejecución programada, el controlador guarda el estado del procesador en la pila de memoria y entra a ejecutar un código especial denominado controlador de interrupciones que atiende al periférico específico que generó la interrupción.

Al terminar la rutina, una instrucción especial le indica al procesador el fin de la atención de la interrupción. En ese momento el controlador restablece el estado anterior, y el programa que se estaba ejecutando antes de la interrupción se reanuda. Las rutinas de atención de interrupciones deben ser lo más breves posibles para que el rendimiento del sistema sea satisfactorio, porque normalmente cuando una interrupción es atendida, todas las demás interrupciones están en espera.

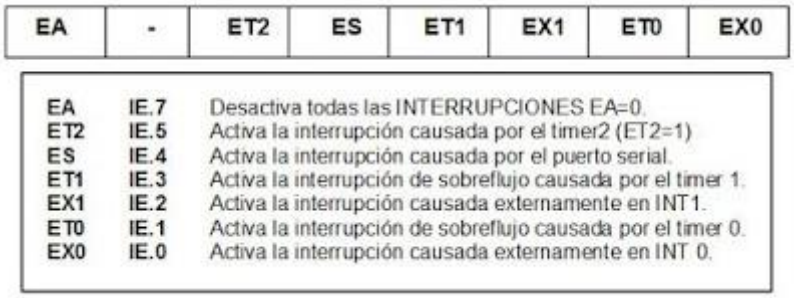
Interrupcion	Direccion
Reset	0000
Interrupcion Externa 0	0003
Timer 0	000B
Interrupcion Externa 1	0013
Timer 1	001B
Puerto Serial Tx y Rx	0023
Timer 2 (Solo At89c52 y superiores)	002B

***Fig.3.4**. Direcciones de memoria de los vectores de interrupción*

Cada una de estas interrupciones posee una dirección de memoria de programa a la cual se direcciona para ejecutar el programa que atenderá dicha interrupción. Las direcciones de interrupción se denominan vectores de interrupción.

Al alimentar el microcontrolador las interrupciones están desactivadas asique deben habilitarse manualmente vía software ya que existe un registro global de interrupciones que habilita la función de interrupción en general; este registro se denomina "IE" Interrup Enable.

Otro registro importante es el IP Interrup Priority de prioridad de interrupción, encargado, en caso de que varias interrupciones se ocasionen al mismo tiempo,de dar a cada una de ellas determina prioridad para ser atendido primero, esto configurado por el software e individualmente cada una de las interrupciones puede ser habilitada o deshabilitada.



EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	Desactiva todas las INTERRUPCIONES EA=0.
ET2	IE.5	Activa la interrupción causada por el timer2 (ET2=1)
ES	IE.4	Activa la interrupción causada por el puerto serial.
ET1	IE.3	Activa la interrupción de sobreflujo causada por el timer 1.
EX1	IE.2	Activa la interrupción causada externamente en INT1.
ET0	IE.1	Activa la interrupción de sobreflujo causada por el timer 0.
EX0	IE.0	Activa la interrupción causada externamente en INT 0.

***Fig.3.5**.Registro "IE" Interrup Enable*

3.1.1.6. OSCILADOR

Todos los microcontroladores disponen de un circuito oscilador que se encarga de generar una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema.

Generalmente, el circuito de reloj está incorporado en el microcontrolador y sólo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Estos componentes suelen consistir en un cristal de cuarzo junto a elementos pasivos. También se puede utilizar un resonador cerámico o una red RC.

Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva aparejado un incremento del consumo de energía.

Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva aparejado un incremento del consumo de energía y de calor generado.

3.1.1.7. RECURSOS ESPECIALES

Cada fabricante oferta numerosas versiones de una arquitectura básica de microcontrolador.

En algunas amplía las capacidades de las memorias, en otras incorpora nuevos recursos, en otras reduce las prestaciones al mínimo para aplicaciones muy simples, etc. La labor del diseñador es encontrar el modelo mínimo que satisfaga todos los requerimientos de su aplicación de esta forma, minimizará el coste, el hardware y el software.

Los principales recursos específicos que incorporan los microcontroladores son:

- **Temporizadores:** también denominados “timers”, son circuitos síncronos para el conteo de los pulsos y se empleados para tareas como medición de frecuencia e implementación de relojes.
- **Perro guardián:** se denomina también “watchdog”, es un mecanismo de seguridad que provoca un reset del sistema en el caso de que éste se bloquee.
- **Brownout:** protección ante fallo de alimentación.
- **Estado de reposo:** o modo “sleep”, que optimiza el rendimiento a bajo consumo.
- **Conversor analógico** – **digital:** realiza la conversión de una señal de naturaleza analógica a digital. Las resoluciones más frecuentes son 8 y 10 bits.
- **Comparador analógico:** es un circuito analógico basado en amplificadores operacionales que tiene la característica de comparar dos señales analógicas y dar como salida los niveles lógicos ‘0’ o ‘1’ según el resultado de la comparación.
- **Modulador de ancho de pulsos o PWM:** realiza una técnica de gran utilidad en diferentes periféricos, como en el control de motores.
- **Puertos de comunicación:** existen numerosos tipos de comunicaciones, entre los que destacan el puerto serie, SPI, USB, I2C, Ethernet y Can.

3.1.1.8. FAMILIAS DE MICROCONTROLADORES

Existen numerosas familias de microcontroladores, cada una de las cuales posee un gran número de variantes. En la siguiente tabla, se indicará los más populares y con mayor uso del mercado:

FABRICANTE	8 bits	16 bits	32 bits
Atmel	89SXXXX, ATmega serie 8XX y 4XX		SAM7, SAM3, SAM9
Freescale (Motorola)	68HC05, 68HC08, 68HC11, HCS08	68HC12, 68HCS12, 68HCSX12, 68HC16	ColdFire, PowerPC, 683XX
Intel	Familias 8048 y 8051	MCS96, MXS296	
Microchip	Familias 10f2XX, 12CXX, 12FXX, 16CXX, 16FXX, 18CXX, y 18FXX	PIC24F, PIC24H, PIC30FXX, dsPIC33F	PIC32
NXP Semiconductors (Philips)	80C51	XA	Cortex-M3, Cortex-M0, ARM7, ARM9
Renesas (Hitachi, Mitsibishi y NEC)	78K, H8	H8S, 78K0R, R8C, R32C/M32C/M16C	RX, V850, SuperH, SH-Mobile, H8SX
STMicroelectronics	ST 62, ST 7		
Texas Instruments	TMS370, MSP430		C2000, TMS570

***Fig.3.6**. Familias de microcontroladores más utilizadas en el mercado*

A nivel individual, destacan los siguientes microcontroladores:

- **8048 (Intel):** es el padre de los microcontroladores actuales, el primero de todos. Su precio, disponibilidad y herramientas de desarrollo hacen que todavía sea muy popular.
- **ATmega328 (Atmel):** usado para la plataforma de hardware libre Arduino en sus diferentes versiones. Dispone de buenas prestaciones a precio reducido.
- **683XX (Freescale):** surgido a partir de la popular familia 68k, a la que se incorporan algunos periféricos. Son microcontroladores de altas prestaciones.
- **PIC (Microchip):** familia de microcontroladores de gran popularidad. Fueron los primeros microcontroladores RISC.
- **ATmega2560 (Atmel):** empleado en la versiones más completa de Arduino , el Mega2560. Destaca por su gran cantidad de entradas y salidas, así como de diferentes periféricos.

3.1.1.9. MERCADO DE LOS MICROCONTROLADORES

Aunque en el mercado de la microinformática la mayor atención la acaparan los desarrollos de los microprocesadores, lo cierto es que se venden cientos de microcontroladores por cada uno de aquéllos.

Producción mundial de microcontroladores por año:

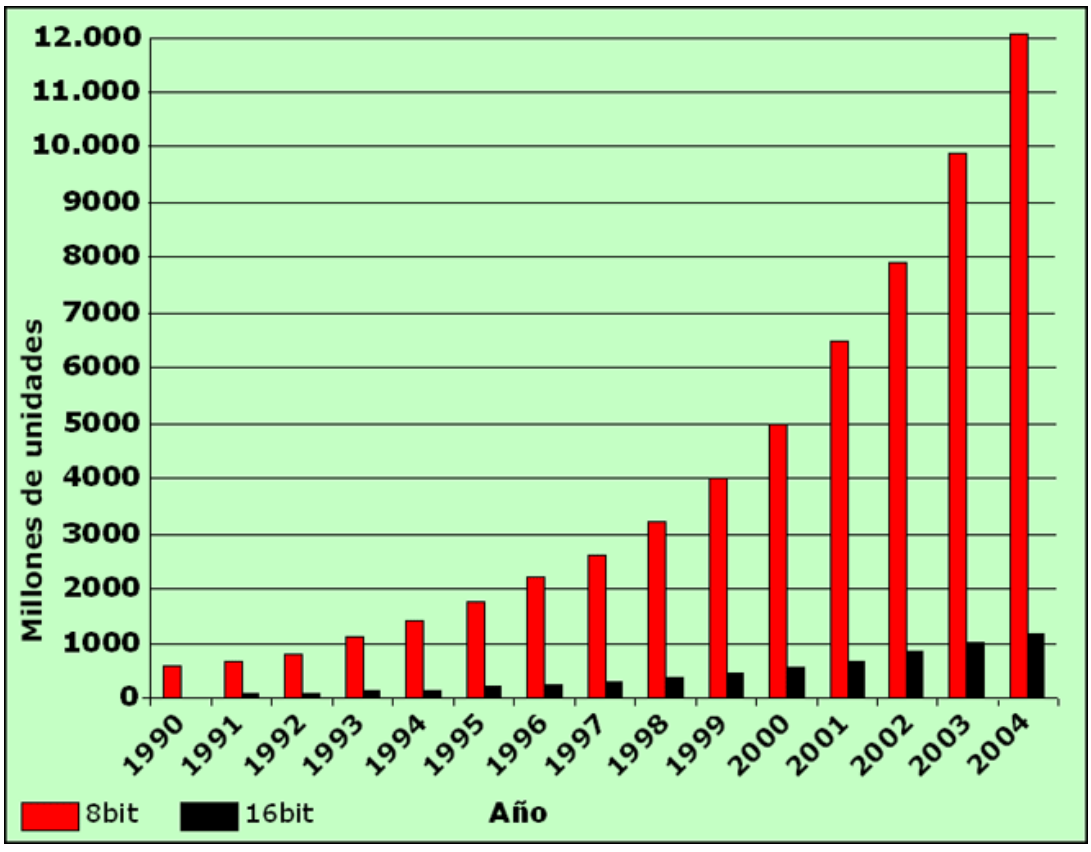


Fig.3.7. *Diagrama de la producción mundial anual de microcontroladores*

Existe una gran diversidad de microcontroladores. Quizá la clasificación más importante sea entre microcontroladores de 4, 8, 16 ó 32 bits. Aunque las prestaciones de los microcontroladores de 16 y 32 bits son superiores a los de 4 y 8 bits, la realidad es que los microcontroladores de 8 bits dominan el mercado y los de 4 bits se resisten a desaparecer. La razón de esta tendencia es que los microcontroladores de 4 y 8 bits son apropiados para la gran mayoría de las aplicaciones, lo que hace absurdo emplear micros más potentes y consecuentemente más caros.

Uno de los sectores que más tira del mercado del microcontrolador es el mercado automovilístico. De hecho, algunas de las familias de microcontroladores actuales se desarrollaron pensando en este sector, siendo modificadas posteriormente para adaptarse a sistemas más genéricos. El mercado del automóvil es además uno de los más exigentes: los componentes electrónicos deben operar bajo condiciones extremas de vibraciones, choques, ruido, etc. y seguir siendo fiables. El fallo de cualquier componente en un automóvil puede ser el origen de un accidente.

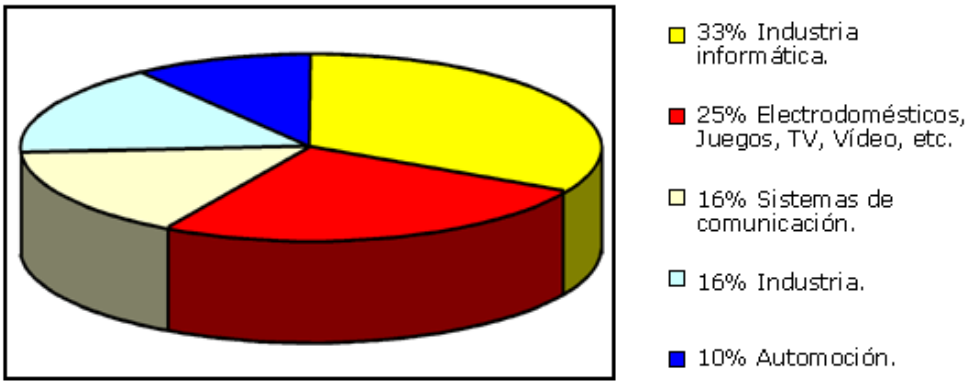


***Fig.3.8**.Uso del microcontrolador en las áreas de mayor difusión*

3.1.1.10. APLICACIONES Y VENTAJAS

El bajo consumo se considera a menudo un aspecto fundamental en un producto ecológico, pero la naturaleza del bajo consumo rara vez se detalla o se cuantifica. Los requisitos del microcontrolador de bajo consumo variarán dependiendo de la aplicación y de cómo se utilizará el microcontrolador en la aplicación. Su uso puede clasificarse en tres áreas principales:

- ❖ Modo de mínimo consumo – Se usará en aplicaciones como termostatos alimentados mediante batería. El modo de mínimo consumo define el nivel más bajo de energía disponible para controlar el visualizador LCD. Esta reducción en el consumo de energía permite prolongar la vida de la batería.
- ❖ Corriente activa consumida – Para aplicaciones como un contador de electricidad, el nivel y la naturaleza del bajo consumo se refiere a la corriente activa consumida por el sistema durante su funcionamiento.
- ❖ Aplicaciones determinadas por el tiempo – Hay sistemas que exigen mantener la fecha y la hora, con independencia de que haya una fuente de alimentación principal del sistema, como un contador de electricidad si hay un fallo del suministro.

Los requerimientos para una solución basada en un microcontrolador, dando así inteligencia, eficiencia y características sofisticadas, se han vuelto cada vez más importantes en la industria de consumo debido a diferentes causas;

- Los consumidores exigen más características y mejor eficiencia
- La competencia en el mercado se ha intensificado

- El mercado global se ha incrementado
- Los gobiernos han introducido leyes más exigentes para cuidar el ambiente.

En la naturaleza de las soluciones discretas no existe flexibilidad ni la habilidad de integrar y desarrollar plataformas de desarrollo comunes para generar diferentes modelos. Mientras que esto puede ser logrado al basar los desarrollos en el uso de microcontroladores.

Los productos que para su regulación y funcionamiento incorporan un microcontrolador, disponen de las siguientes ventajas:

- Aumento de prestaciones, debido al mayor control sobre un determinado elemento, esto representa una mejora considerable en el mismo.
- Gran fiabilidad, ya que al remplazar este dispositivo por un elevado número de elementos reduce considerablemente el riesgo de averías y se precisan menos ajustes.
- Disminución del tamaño en el producto acabado, la integración del microcontrolador en un chip disminuye el volumen, la mano de obra y los stocks.
- Mayor flexibilidad, puesto que las características de control están programadas, su modificación sólo necesita cambios en el programa de instrucciones o software.

3.1.1.11. LA ELECCIÓN DEL MICROCONTROLADOR

A la hora de decidirse en términos generales sobre qué tipo o familia de microcontroladores emplear hay que tener en cuenta varios factores, como por ejemplo:

- **Experiencia previa:** Si ya se ha trabajado con algún microcontrolador en particular, lo mejor es ver que nuevas posibilidades ofrecen los diversos fabricantes que trabajen con ese microcontolador como núcleo.

  Por ejemplo, el 8051 fué muy popular hace algun tiempo, y hay muchos microcontroladores actuales que derivan de este, como pueden ser los AT89 de Atmel, MCS251 de Intel, DS8 de Maxim (Dallas), P8 de Philips y MSC12 de Texas Instruments. Lo bueno es que salvo en determinados detalles se mantiene el conjunto de instrucciones, modos de direccionamiento, nombres de registros, y en definitiva la filosofía de trabajo del 8051. Con pequeños cambios podrían recuperarse programas diseñados para el 8051 con estos microcontroladores.

- **Documentación existente**: Este es un factor importante si se quiere conocer bien el tipo de microcontrolador elegido y su entorno de desarrollo. Además, si existe una amplia literatura de aplicaciones podrán utilizarse programas y diseños ya realizados para adaptarlos a nuestras necesidades. Mediante libros especializados, revistas de electrónica y sobre todo Internet, puede encontrarse la información necesaria sobre cualquier microcontrolador.

- **Herramientas de desarrollo disponibles y su precio**: Uno de los factores que más importancia tiene a la hora de seleccionar un microcontrolador entre todos los demás es el soporte tanto software como hardware de que dispone. Un buen conjunto de herramientas de desarrollo puede ser decisivo en la elección, ya que pueden suponer una ayuda inestimable en el desarrollo del proyecto. Algunos fabricantes de microcontroladores ofrecen paquetes IDE de calidad de forma completamente gratuita, como política para inclinarse por el uso de sus microcontroladores. Ejemplos de ello son AVR studio de Atmel, Code Warrior de Freescale (Motorola), MPLAB de Microchip o Eclipse de Texas Instruments. Estos paquetes IDE gratuitos permiten programar en código ensamblado, puesto que los compiladores de lenguaje de alto nivel (BASIC y C) no suelen ser gratis.

- **Precio del microcontrolador**: Como es lógico, los fabricantes de microcontroladores compiten duramente para vender sus productos. Para una producción a gran o mediana escala de dispositivos que utilizan un microcontrolador, una diferencia de precio en el mismo de algunos céntimos es importante (el consumidor deberá pagar además el coste del empaquetado, el de los otros componentes, el diseño del hardware y el desarrollo del software).

Estos puntos anteriores se refieren al tipo de familia o fabricante, si lo que queremos es, más concretamente, un tipo de microcontrolador nos deberemos fijar en las siguientes características;

- **Entradas, salidas y recursos internos**: Uno de los aspectos más atractivos de los microcontroladores es que la circuitería externa puede reducirse al mínimo. Para determinar las necesidades de entradas y salidas así como los recursos del sistema es conveniente dibujar un diagrama de bloques del mismo, de tal forma que sea sencillo identificar la cantidad y tipo de señales a controlar. Habrá que tener en cuenta:Número de entradas y salidas necesarias;

  - Número y tamaño (8, 16 ó 32 bits) de los temporizadores necesarios.
  - Necesidad de un CAD o CDA, incluyendo la resolución y número de entradas.
  - Necesidad de puertos de comunicaciones (I2C, RS232, USB, bus CAN, SPI u otros).
  - Necesidad de una o más salidas PWM.
  - Necesidad de interfaces específicas como la de control LCD.

- **Velocidad y consumo**: Actualmente pueden encontrarse modelos de microcontroladores que pueden utilizar velocidades de reloj de hasta 100 MHz, pero además de eso, en la velocidad de ejecución del programa repercute la arquitectura del microcontrolador, siendo más rápido uno con arquitectura RISC que otro con CISC.

  Las velocidades altas incrementan las interferencias electromagnéticas radiadas y el consumo de los microcontroladores al estar la mayoría realizados con tecnologías CMOS, por lo que habrá que tener en cuenta la velocidad si el consumo es importante en la aplicación. Por todo esto el uso de velocidades altas debe reservarse para cuando sea necesario que el microcontrolador realice cálculos críticos en un tiempo limitado. En ese caso debemos asegurarnos de seleccionar un dispositivo suficientemente rápido para ello.

  Hay que tener en cuenta que algunos productos que incorporan microcontroladores están alimentados con baterías, lo más conveniente puede ser que el microcontrolador esté en estado de bajo consumo pero que despierte ante la activación de una señal (una interrupción) y ejecute el programa adecuado para procesarla. En las situaciones donde un dispositivo se relaciona con el entorno humano suele utilizarse el modo de bajo consumo, como por ejemplo en un mando a distancia de un televisor, que la mayor parte del tiempo no está haciendo nada, de manera que cuando el usuario pulsa una tecla el microcontrolador pasa al modo normal y ejecuta las operaciones necesarias.

- **Memoria:** Para determinar las necesidades de memoria de nuestra aplicación debemos separarla en memoria volátil (RAM), memoria no volátil (ROM, Flash, etc.) y memoria no volátil modificable (EEPROM).

  Este último tipo de memoria puede ser útil para incluir información específica de la aplicación como un número de serie o parámetros de calibración.

  El tipo de memoria a emplear vendrá determinado por el volumen de ventas previsto del producto: de menor a mayor volumen será conveniente emplear Flash, EEPROM, OTP y ROM.

Los dos primeros tipos han sido pensados y diseñados para ser utilizados en etapas de desarrollo o en pequeñas series, para una producción en masa a pequeña escala es preferible utilizar el tipo OTP (que puede programarse como los dos tipos anteriores pero no se puede borrar y es normalmente más barato). El último tipo, ROM, necesita ser programado mediante una máscara por el fabricante de manera que sólo es práctico para cuando se necesiten varios miles de dispositivos idénticos. En cuanto a la cantidad de memoria necesaria puede ser necesario realizar una versión preliminar de la aplicación y a partir de ella hacer una estimación de cuánta memoria volátil y no volátil es necesaria y si es conveniente disponer de memoria no volátil modificable.

- **Ancho de palabra**: El criterio de diseño debe ser seleccionar el microcontrolador de menor ancho de palabra que satisfaga los requerimientos de la aplicación. Los modelos de 4 bits han desaparecido prácticamente del mercado de manera que utilizar un microcontrolador de 8 bits supone la mejor elección si el programa a desarrollar sólo controla unas pocas entradas y salidas y no utiliza cálculos complejos ni accede a grandes bases de datos. También resultan perfectos si el ancho de los datos es de un byte. Los microcontroladores de 16 y 32 bits, deberán utilizarse si se realizan cálculos matemáticos o científicos, una gestión de Entrada/Salida potente o si se necesita un espacio de direccionamiento muy elevado. Si una aplicación necesita un microcontrolador con más de 8 bits, es recomendable utilizar microcontroladores de 32 bits frente a los de 16 bits dada la poca diferencia de precio que actualmente existe entre ellos.

- **Disponibilidad**: Hoy es muy fácil realizar compras por medio de catálogos por correo o a través de Internet, incluso pueden solicitarse componentes directamente al fabricante. El problema está en el número de dispositivos que se deben pedir. El fabricante sólo nos atenderá si se solicitan cantidades realmente grandes, aparte de los problemas sobre licencias, permisos o aduanas que puedan surgir. La venta por catálogo mediante correo o por Internet dentro del país resulta muy interesante apenas se compre el suficiente material como para amortizar los gastos de transporte. También debemos considerar que cuanto más popular sea el microcontrolador que elijamos menos problemas vamos a tener en este aspecto.

- **Diseño del circuito y de la PCB:** La selección de un microcontrolador concreto condicionará el diseño del circuito de manera que debe tenerse en cuenta que quizá usar un microcontrolador barato encarezca el resto de componentes del diseño.

  Tampoco debemos olvidarnos del encapsulado, podríamos elegir un determinado modelo de microcontrolador y luego encontrarnos que en lugar de venir con el tradicional encapsulado DIL sólo esté disponible en encapsulados PLCC o PGA, si bien siempre podremos utilizar un zócalo adecuado. Tampoco se podría trabajar manualmente con encapsulados BGA. Afortunadamente, de momento, estos problemas se dán sólo con los últimos modelos de microcontroladores con muchas patillas.

  En cuanto al diseño de la placa de circuito impreso (PCB) a no ser que el esquema sea simple o se afine mucho en el mismo será necesario el uso de puentes (su uso está mal visto) o de placas de c.i. de doble cara. Si se utilizan componentes SMD conviene saber que la primera generación de componentes SMD tiene una separación de terminales de 1,27 milímetros que todavía pueden soldarse manualmente con paciencia y una punta fina pero los últimos circuitos SMD tienen una separación de 0,64 milímetros paro lo cual es necesario utilizar máquinas de soldadura.

3.2. PLATAFORMA ARDUINO

3.2.1. Elección de la plataforma Arduino

Arduino se basa en una plataforma denominada open hardware que reúne en una pequeña placa de circuito impreso (PCB) los componentes necesarios para conectar con el mundo exterior y hacer funcionar un microcontrolador Atmega.

Actualmente hay varios modelos de sistemas Arduino que van cambiando de microcontrolador, siendo los primeros el Atmega8 y el Atmega168. Al ser Open-Hardware, tanto su diseño como su distribución son libres. Es decir, puede utilizarse sin inconvenientes para desarrollar cualquier tipo de proyecto sin tener que adquirir ningún tipo de licencia.

La placa es de muy fácil montaje, con pocos componentes periféricos al microcontrolador. Una vez armada la placa con sus componentes, resta colocar en ella el microcontrolador y programarla.

En la actualidad, el fenómeno Arduino está creciendo rápidamente, al igual que la diversidad de modelos que puede elegir el cliente. Hay dos cuestiones principales que hacen la gran diferencia a la hora de elegir dichos modelos: la primera es el tipo de microcontrolador a utilizar, y la segunda es el modo de comunicación que poseerá la placa Arduino con el ordenador.

Tal como dijimos antes, los tipos de microcontroladores son dos: Atmega8 y Atmega168. La diferencia entre ambos es la capacidad de memoria interna que poseerán para almacenar el programa que diseñemos e introduzcamos en él.

En cuanto a la comunicación de Arduino con el ordenador, encontramos que ésta se realiza por Puerto Serie (RS232), Puerto USB (utilizando un FT232BL para la interconexión), o por el sistema ICSP (In Circuit Serial Program) en aquellos casos en que el deseo del usuario sea una unidad autónoma (stand alone) sin necesidad de interacción con el ordenador para su actividad y desarrollo de funciones.

Recordemos que la comunicación se utiliza tanto para la interacción de Arduino con el ordenador (cuando el programa grabado en el dispositivo así lo requiera) como para la programación del microcontrolador.

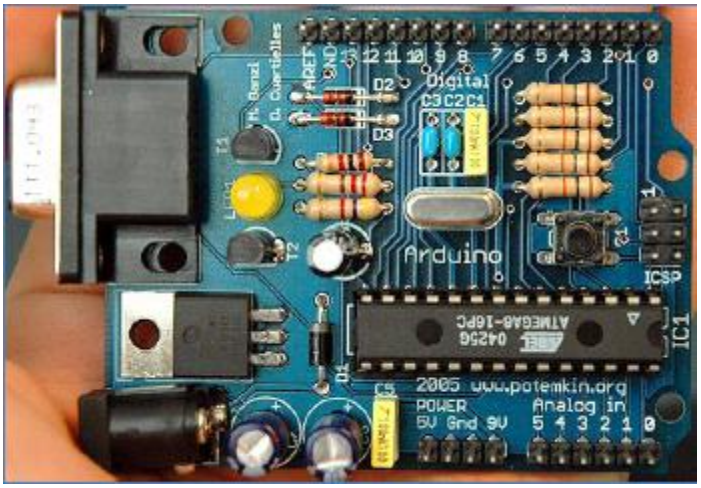


Fig.3.9. *Primera placa de Arduino, denominada Serial.*

Un ejemplo sencillo de unidad autónoma es una alarma con sensores mecánicos, infrarrojos y actuadores. En cambio, si se trata de una unidad que interactúa con el ordenador, puede ser una pequeña central meteorológica que vaya almacenando datos y mostrando en pantalla (en tiempo real) la información obtenida en cada instante.

Por último, al software encargado de "crear" el programa que hará funcionar al microcontrolador lo podemos descargar también gratuitamente desde el sitio oficial de Arduino. Actualmente se encuentra disponible la versión Arduino 0015, en versiones disponibles para Windows, MAC y Linux (32 bit).

La placa armada (comprada o hecha por nosotros mismos) consta de aquello que figura en la imagen anterior: un regulador de tensión para brindar al microcontrolador una tensión estabilizada de alimentación (5Volts), el conector de comunicaciones ICSP, y las 6 entradas analógicas para sensores de cualquier tipo, como ser potenciómetros, sensores magnéticos, termocuplas, LDRs, optoacopladores, fototransistores, y cuanto sensor analógico se nos ocurra.

Vale aclarar que también puede conectarse allí la salida de cualquier amplificador operacional que haga las veces de buffer de entrada al sistema, brindando a la entrada seleccionada una mejor adaptación de impedancias, junto con una buena aislación y separación entre bloques circuitales.

Por último, encontramos las I/O digitales que sirven para activar algún relé, luces, motores, etc.

Esta plataforma fue creada por los ingenieros Banzi y Cuartielles, entre otros.

Los microcontroladores disponen de numerosas ventajas y aplicaciones, aunque que no pueden competir con esta completa plataforma, que reúne las características propias de microcontrolador más las que añaden los componentes adicionales que forman la placa.

3.2.2. Diferentes modelos de Arduino

Desde el año 2005, cuando apareció el primer prototipo de Arduino como tal, muchos han sido los modelos que se han implementado, aportando cada uno determinadas características que lo diferenciaban del resto.

En ocasiones, las nuevas placas servían de versiones mejoradas de las anteriores, por lo que a nivel de fabricación algunos modelos han desaparecido.

En este apartado, se tratarán exclusivamente las plataformas que en la actualidad se tienen más utilidad, debido a que son las versiones más modernas y potentes:

Arduino Nano

La plataforma Nano es una pequeña y completa placa que implementa el microcontroladores de Atmel, ATmega168, en su revisión 2.X o ATmega328 en su revisión 3.0. Para su correcto uso se recomienda conectar la plataforma a una placa de prototipos.

Dispone similares características funcionales que Arduino Uno, pero con una presentación diferente. No tiene conector para alimentación externa y funciona mediante un cable USB Mini-B en vez del cable estándar, tipo B. Nano es diseñado y producido por la empresa estadounidense Gravitech; Arduino Nano que es como el Arduino Mini pero aún más pequeño si cabe.

Es la nueva generación de placas que permite un rápico prototipado sobre una protoboard. ésta vez, incorpora un conector mini USB, un chip ATMega328, 2 entradas analógicas más que la placa Arduino Diecimila y un conector ICSP para programarlo mediante un programador externo si se desea, sin necesidad de cablear el contector externamente.

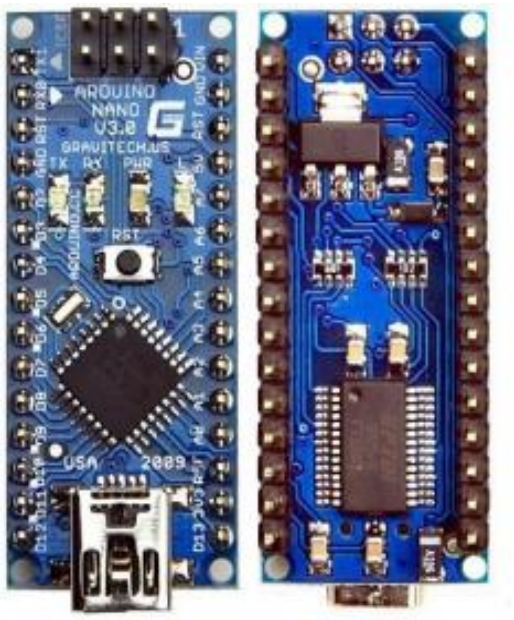


Fig.3.10. *Arduino Nano v3 vista frontal y posterior*

Características de Arduino Nano	
Microcontrolador	Atmel ATmega168 o ATmega328
Tensión de Operación (nivel lógico)	5 V
Tensión de Entrada	7 – 12 V *(6 – 20 V límite)*
Pines E/S Digitales	14 *(6 con salida PWM)*
Entradas Analógicas	8
Corriente máx por cada pin de E/S	40 mA
Memoria Flash	16 KB (ATmega168) o 32 KB (ATmega328) de los cuales 2KB son usados por el bootloader
SRAM	1 KB (ATmega168) o 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) o 1 KB (ATmega328)
Frecuencia de reloj	16 MHz
Dimensiones	18.5mm x 43.2mm

Fig.3.11. *Tabla de especificaciones técnicas del Arduino Nano*

Arduino Leonardo

La placa Arduino Leonardo es una plataforma que dispone del microcontrolador ATmega32u4, fabricado por la empresa Atmel. Dispone 20 entradas/salidas digitales (7 de las cuales se pueden emplear como salidas PWM), además de 12 entradas analógicas. También implementa un cristal oscilador de 16MHz, una conexión de tipo micro USB, una toma de corriente, un cabezal ICSP y un botón de reset. Contiene todo lo necesario para utilizar el microcontrolador, sólo es necesario conectarlo al ordenador mediante USB o alimentarlo con un transformador o batería para empezar a trabajar con él.

Leonardo, se diferencia del resto de placas Arduino en el microcontrolador que utiliza, el ATmega32u4 que incorpora comunicación USB, por lo que no requiere de un segundo procesador para esta comunicación. Esto permite que aparezca conectado como un periférico más, junto con un puerto serie virtual (COM).

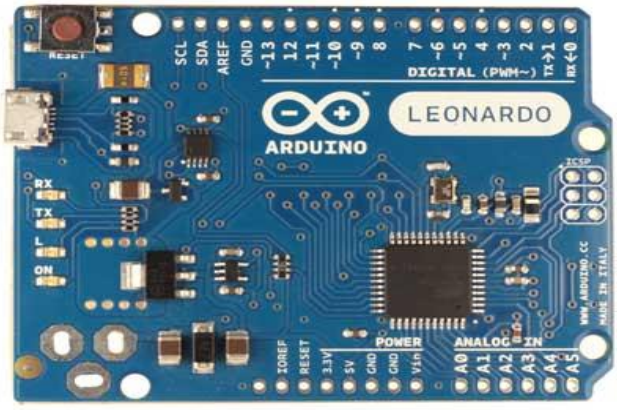


***Fig.3.12**. Vista frontal del Arduino Leonardo*

Características de Arduino Leonardo	
Microcontrolador	Atmel ATmega32u4
Tensión de Operación (nivel lógico)	5 V
Tensión de Entrada	7 – 12 V *(6 – 20 V límite)*
Pines E/S Digitales	20 *(7 con salida PWM)*
Entradas Analógicas	12
Corriente máxima por cada pines de E/S	40 mA
Corriente máxima para pin 3.3 V	50 mA
Memoria Flash	32 KB (ATmega32u4) de los cuales 4 KB son usados por el bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Frecuencia de reloj	16 MHz
Dimensiones	68.6 mm x 53.3mm

***Fig.3.13.** Tabla de especificaciones técnicas del Arduino Leonardo*

Arduino Mega 2560

Arduino Mega 2560 es una placa equipada con un microcontrolador fabricado por Atmel, el ATmeg2560. Está formada por 54 entradas/salidas digitales (de las cuales 14 proporcionan salida PWM), 16 entradas digitales, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz, conexión USB, entrada de corriente mediante jack, conector ICSP y botón de reset. Contiene todo lo necesario para utilizar el microcontrolador, sólo es necesario conectarlo al ordenador mediante USB o alimentarlo con un transformador o batería para empezar a trabajar con él.

Mega 2560, es una versión actualizada que remplaza a Arduino Mega, compatible con la mayoría de "shields" diseñados para Arduino Uno y versiones anteriores de iguales dimensiones.

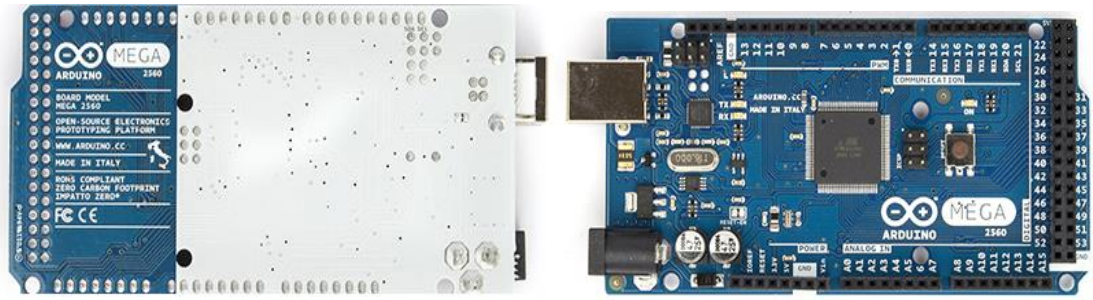


Fig.3.14. *Vista posterior (izda.) y frontal (dcha.) de la placa de Arduino Mega 2560*

Características de Arduino Mega 2560	
Microcontrolador	Atmel ATmega2560
Tensión de Operación (nivel lógico)	5 V
Tensión de Entrada	7 – 12 V *(6 – 20 V límite)*
Pines E/S Digitales	54 *(15 con salida PWM)*
Entradas Analógicas	16
Corriente máx por cada PIN de E/S	40 mA
Corriente máxima para pin 3.3 V	50 mA
Memoria Flash	256 KB (Atmega2560) de los cuales 8 KB son usados por el bootloader
SRAM	8 KB (Atmega2560)
EEPROM	4 KB (Atmega2560)
Frecuencia de reloj	16 MHz
Dimensiones	101.6 mm x 53.3mm

Fig.3.15. *Tabla de especificaciones técnicas del Arduino Mega 2560*

Arduino Uno

Arduino Uno es una plataforma equipada con el microcontrolador ATmega 328 de Atmel. Está compuesta por 14 entradas/digitales (6 de las cuales se pueden emplear como salidas PWM), además de 6 entradas analógicas. También implementa un cristal oscilador de 16MHz, una conexión de USB tipo B, una toma de corriente, conector ICSP y un botón de reset.

Contiene todo lo necesario para utilizar el microcontrolador, sólo es necesario conectarlo al ordenador mediante USB o alimentarlo con un transformador o batería para empezar a trabajar con él.
Uno, se diferencia del resto de versiones de Arduino en que difiere de todas las placas anteriores puesto que no utiliza el chip de controlador de FTDI USB a puerto serie. En cambio, cuenta con el Atmega8U2 programado como un convertidor de USB a puerto serie.

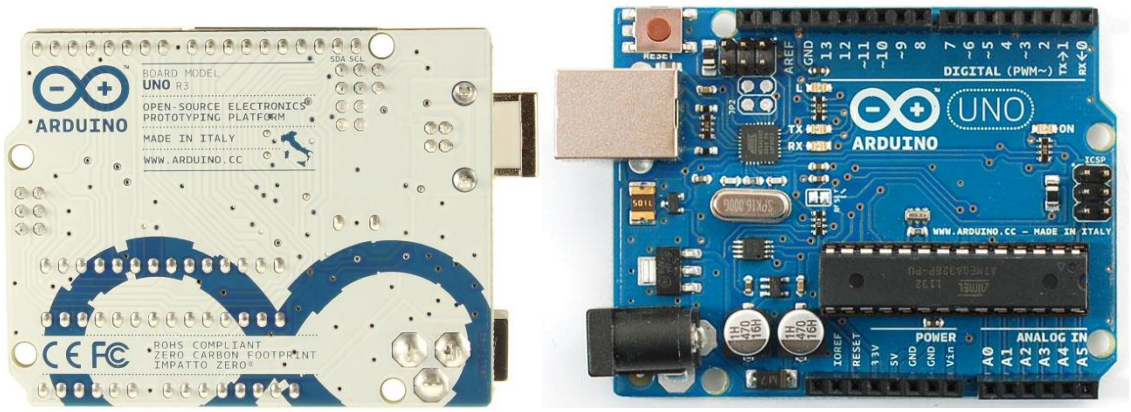


***Fig.3.16**. Vista posterior (izda.) y frontal (dcha.) del Arduino Uno rev3*

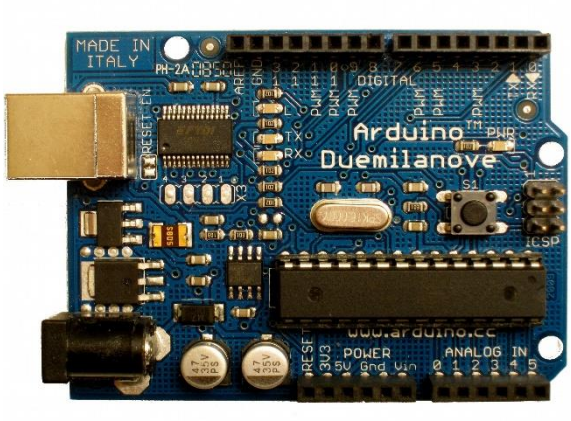
Características de Arduino Uno	
Microcontrolador	Atmel ATmega328
Tensión de Operación (nivel lógico)	5 V
Tensión de Entrada	7 – 12 V *(6 – 20 V límite)*
Pines E/S Digitales	14 *(6 con salida PWM)*
Entradas Analógicas	6
Corriente máx por cada PIN de E/S	40 mA
Corriente máxima para pin 3.3 V	50 mA
Memoria Flash	32 KB (Atmega328) de los cuales 0.5 KB son usados por el bootloader
SRAM	2 KB (Atmega328)
EEPROM	1 KB (Atmega328)
Frecuencia de reloj	16 MHz
Dimensiones	68.6 mm x 53.3mm

***Fig.3.17.** Tabla de especificaciones técnicas del Arduino Uno rev3*

Si bien hemos hecho hincapié en los modelos de Arduino más conocidos y usados, existen otros muchos con diversas aplicaciones más especializadas.

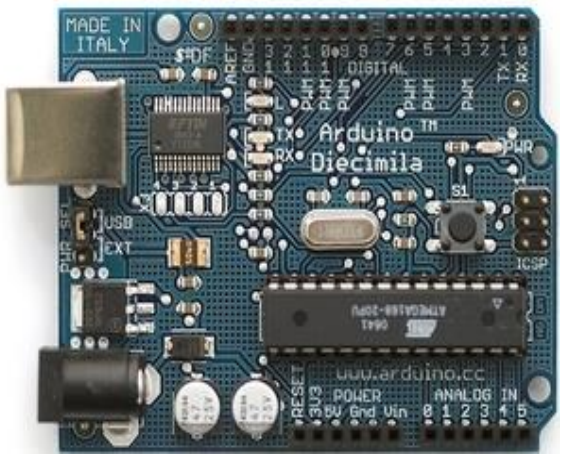
Aquí hacemos alusión a varios de ellos;

Arduino Duemilanove



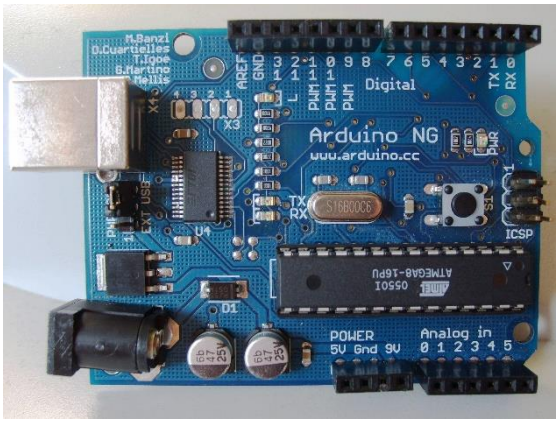
El Duemilanove automáticamente selecciona la fuente de alimentación adecuada (USB o externa), eliminando la necesidad de usar un jumper (especie de conmutador) de selección de fuente como ocurría en placas anteriores. Para que resulte cómodo se puede cortar la pista para deshabilitar el auto-reset y soldar un jumper en el corte para habilitarlo cuando sea necesario. Diseñado originalmente en base al ATmega168, pero a partir de marzo del 2009 empezó a comercializarse con el ATmega328p.

Arduino Diecimila

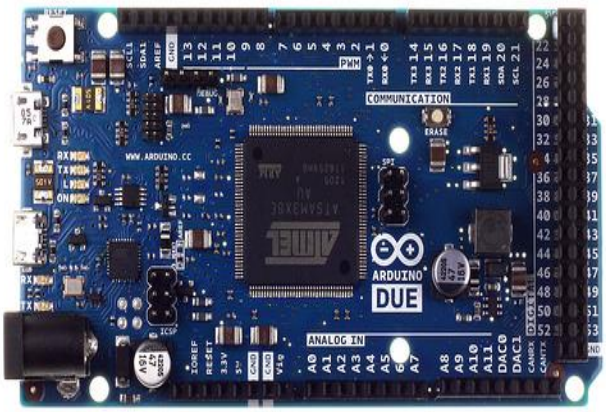


La principal diferencia en el Arduino Diecimila es que puede ser reseteado desde el PC, sin necesidad de ser reseteado físicamente usando el botón de reset de la placa. El Diecimila usa un regulador de baja caída de tensión lo cual reduce el consumo de la placa cuando se alimenta con una fuente externa (Adaptador de pared o batería). La placa posee un fusible reseteable que protege el puerto USB de tu PC contra cortocircuitos y sobre tensiones. También posee pines hembra para la línea de reset y de 3.3V.

Arduino NG

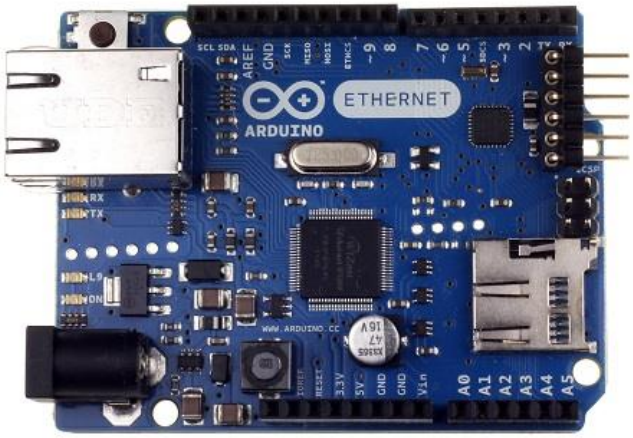


Usa el conversor serie a USB FTDI FT232RL, el cual necesita menos componentes externos que el FT232BM. Este también posee un LED en el pin 13, aunque en su última revisión se eliminó este LED porque podía interferir con la comunicación SPI.
Originalmente comercializado con el ATmega8, paso a producirse en base al ATmega168



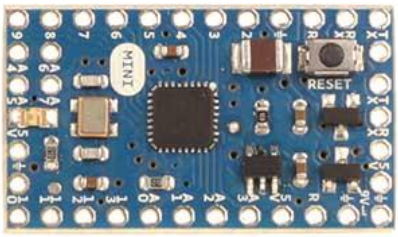
Arduino Due

El Arduino Due es una placa electrónica basada en el Atmel SAM3X8E ARM Cortex-M3 de la CPU. Es la primera placa Arduino basado en un microcontrolador núcleo ARM de 32 bits. Cuenta con 54 pines digitales de entrada / salida ,12 entradas analógicas, 4 UARTs, un reloj de 84 MHz, una conexión USB OTG capaz, 2 DAC, 2 TWI.A diferencia de otras placas Arduino la tensión máxima que los pines de E / S pueden tolerar es 3.3V. Proporcionar voltajes más altos, como 5V a un pin de E / S podría dañar la placa.



Arduino Ethernet

Cuenta con 14 pines digitales de entrada / salida, 6 entradas analógicas, un oscilador de 16MHz, una conexión RJ45, un conector de alimentación, una cabecera ICSP, y un botón de reinicio.La Ethernet se diferencia de otras placas en que no tiene un chip integrado controlador de USB a serie, pero tiene una interfaz Wiznet Ethernet. Esta es la misma interfaz que se encuentra en el escudo Ethernet.



Arduino Mini

Es una pequeña placa de desarrollo basado originalmente en el ATmega168, pero usado con el 328. Destinado a usarse cuando el espacio es un bien escaso. Cuenta con 14 pines digitales de entrada /, 8 entradas analógicas, y un oscilador de 16MHz. Se puede programar con el adaptador de serie USB u otro USB o RS232 a TTL adaptador serie.

3.2.3. Arduino Uno. Elección

Tras el análisis de los diferentes modelos disponibles actualmente de la plataforma Arduino, se realizó una tabulación comparativa de sus características y se analizó que versión podría ser la más conveniente para su utilización en el prototipo del control por radio.

A continuación, se comparan los parámetros de mayor importancia a la hora de obtener un prototipo eficiente y con viabilidad:

Modelo Arduino	NANO	LEONARDO	UNO	MEGA 2560
E/S Digitales	14	20	14	54
Entradas Analógicas	8	12	6	16
Microcontrolador	ATmega168 (SMD) o ATmega328 (SMD)	ATmega32u4 (SMD)	ATmega328	ATmega2560 (SMD)
Dimensiones	18.5mm x 43.2mm	68.6 mm x 53.3mm	68.6 mm x 53.3mm	101.6 mm x 53.3mm
Acepta Shields	No	Si	Si	Si
Precio (IVA no incluido)	33 €	18 €	20 €	39 €

Fig.2.18. *Tabla de comparativas referente a los arduinos Nano, Leonardo, Uno y Mega 2560*

Atendiendo al número de entradas y salidas digitales que ofrecen las alternativas propuestas, no se excluye ningún modelo, ya que disponen del número mínimo o superan las necesarias para la realización del prototipo.
En cuanto al número de entradas analógicas, tampoco supone una limitación que elimine a algún candidato.
Al analizar los diferentes microcontroladores implementados, destacan los que disponen de mayores recursos, como el ATmega328 y el ATmega32u4.

Puesto que el proyecto es un prototipo, será sometido a múltiples experimentos de comportamiento y a su continua depuración para la corrección de errores, las plataformas Arduino equipadas con microcontroladores de montaje superficial o SMD, tienen un punto en contra ya que ,en el caso de sustitución del microcontrolador debido a una avería, en los de tecnología de montaje superficial habría que sustituir toda la plataforma Arduino, lo que aumentaría los costes. La versión Uno es la única con microcontrolador intercambiable.

Una de las diferencias más importante de la tarjeta Arduino UNO respecto a sus predecesoras, es que no utiliza el convertidor USB-serie de la firma FTDI. Por el contrario, integra un microcontrolador Atmega 8U2 programado como un convertidor o puente de USB a serie donde se incluyen los drivers necesarios.

Se basa en un microcontrolador Atmel ATmega320 de 8 bits a 16Mhz que funciona a 5v. 32KB son correspondientes a la memoria flash (0,5KB reservados para el bootloader), 2KB de SRAM y 1KB de EEPROM. En cuanto a memoria es una de las placas más limitadas, pero no por ello resulta insuficiente

3.2.3.1. CARACTERÍSTICAS

El Arduino Uno es el modelo de referencia para la plataforma Arduino y es compatible con la gran mayoría de los shields existentes.

Arduino UNO es una placa con microcontrolador basada 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, una cabecera ISCP, y un botón de reset. Contiene todo lo necesario para utilizar el microcontrolador; simplemente conectarse al ordenador a través del cable USB o alimentarlo con un transformador o una batería para empezar a trabajar con él.

El Arduino UNO difiere de todos sus precedentes en que no usa el chip driver FTDI USB-to-serial. En vez de esto viene con un Atmega16U2 programado como convertidor USB-to-serial.

En el esquemático de la figura vemos detalladamente las líneas de pines y conexiones internas así como los comparadores y puertas que implementan la placa del Uno Rev3.

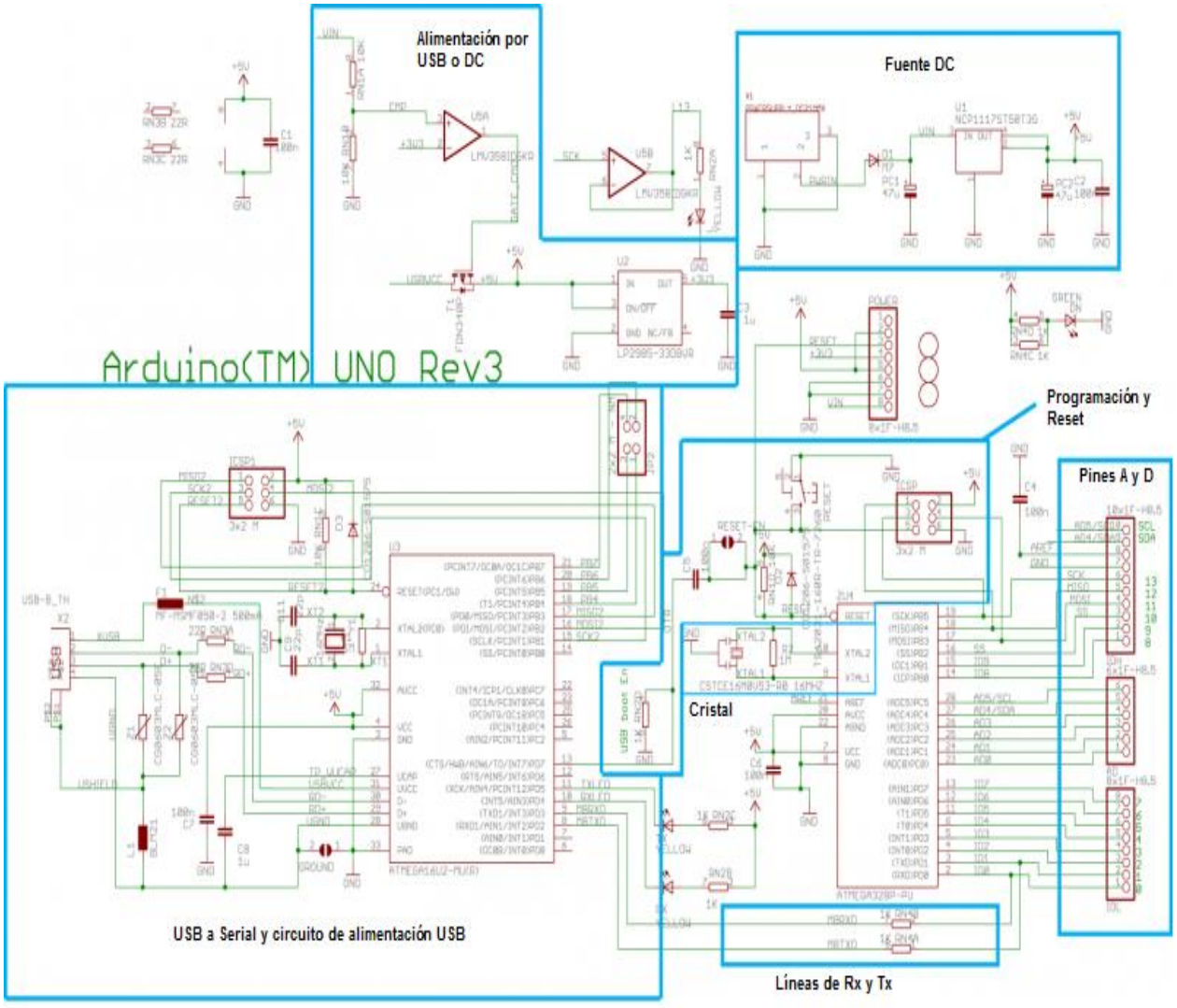


Fig.3.19. *Esquemático Arduino Uno Rev3*

Todas las características de esta placa estarán implementadas en casi todas las placas restantes, a excepción de algunas.

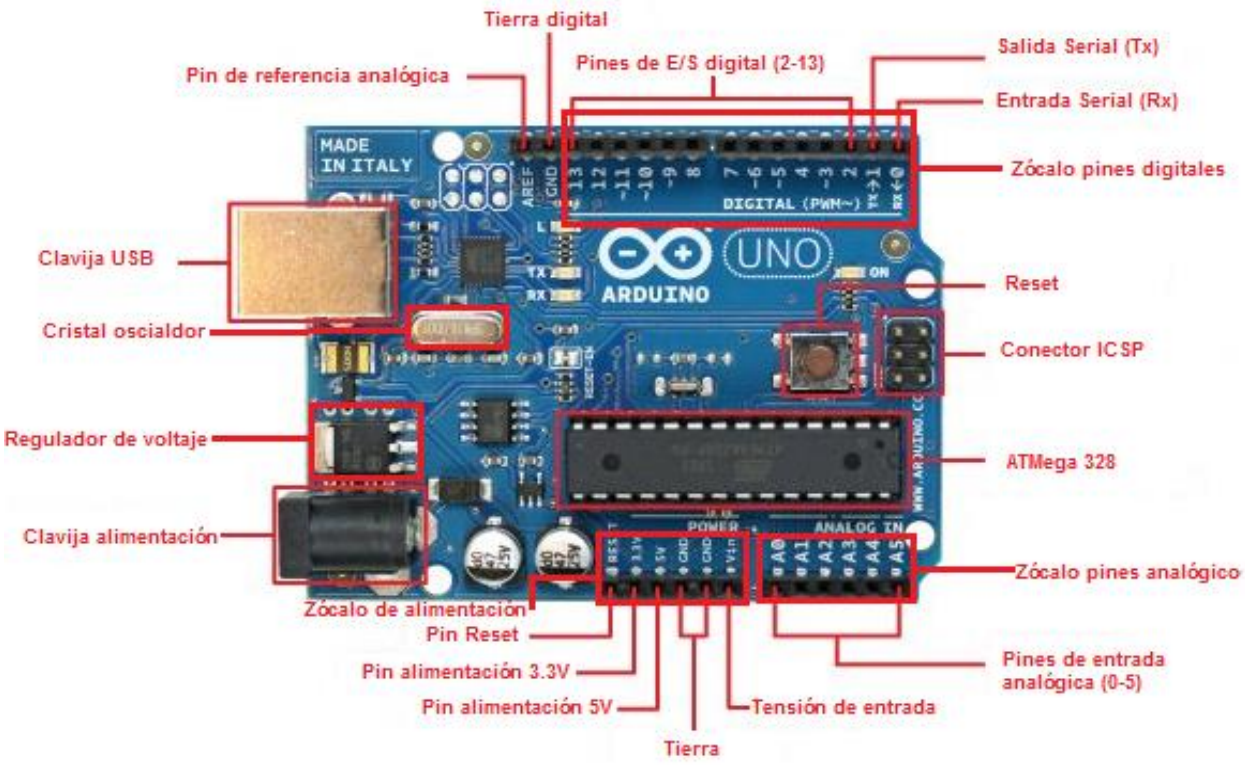


Fig.3.20. *componentes del Arduino Uno*

A continuación, en los siguientes apartados, describiremos brevemente cada una de las características de los elementos que conforman la placa de Arduino Uno, desde el pin AREF, pasando por el botón de reset hasta los pines de alimentación; estas características, en conjunto global, hacen del Arduino Uno Rev3, la mejor elección.

3.2.3.2. ALIMENTACIÓN

El Arduino Uno puede ser alimentado a través de la conexión USB o con una fuente de Alimentación externa. La fuente de alimentación se selecciona automáticamente.

La alimentación externa puede ser suministrada tanto por un conversos AC/DC como por una batería. El adaptador puede ser conectado enchufando un conector de 2.1mm en la clavija de alimentación de la placa. Los cables desde una batería se pueden insertar en los cabezales de pin GND y Vin del conector de alimentación.

La placa puede funcionar con un suministro externo de 6 a 20 voltios. Si se suministra con menos de 7V, sin embargo, el pin de 5V puede suministrar menos de cinco voltios y la placa puede ser inestable al igual que si se utiliza más de 12 V, el regulador de voltaje se puede sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes:

- **VIN:** Voltaje de entrada a la placa Arduino cuando se trata de utilizar una fuente de alimentación externa (en oposición a 5 voltios de la conexión USB u otra fuente de alimentación regulada). Puede suministrar tensión a través de este pin, o, si el suministro de tensión es a través de la toma de alimentación, se a él a través de este pin.

- **5V**: Con este pin la fuente de alimentación regulada utilizada para alimentar el microcontrolador y otros componentes de la placa. Esto puede venir de VIN o ser suministrada por USB u otra fuente de 5V regulada.
- **3V3:** suministro de 3,3 voltios por el regulador de la placa. Soporta una corriente máxima es de 50 mA.
- **TIERRA:** pines de toma de tierra.
- **RESET:** Suministrar un valor LOW (0V) para reiniciar el microcontrolador. Típicamente usado para añadir un botón de reset a los shields que no dejan acceso a este botón en la placa.

3.2.3.3. MEMORIA

El ATmega328 está basado un microcontrolador RISC, combinando 32 KB ISP flash una memoria con la capacidad de leer-mientras-escribe, 1 KB de memoria EEPROM, 2 KB de SRAM, 23 lineas de E/S de propósito general, 32 registros de proceso general, tres temporizadores flexibles/contadores con modo de comparación, interrupciones internas y externas, programador de modo USART, una interfase serial orientada a byte de 2 cables, SPI puerto serial, 6-canales 10-bit Conversor A/D, "watchdog timer" programable con oscilador interno, y cinco modos de ahorro de energía seleccionables por software.

El dispositivo opera entre 1.8 y 5.5 voltios. Por medio de la ejecución de instrucciones en un solo ciclo de reloj, el dispositivo alcanza una respuesta de 1 MIPS, balanceando consumo de energía y velocidad de proceso.

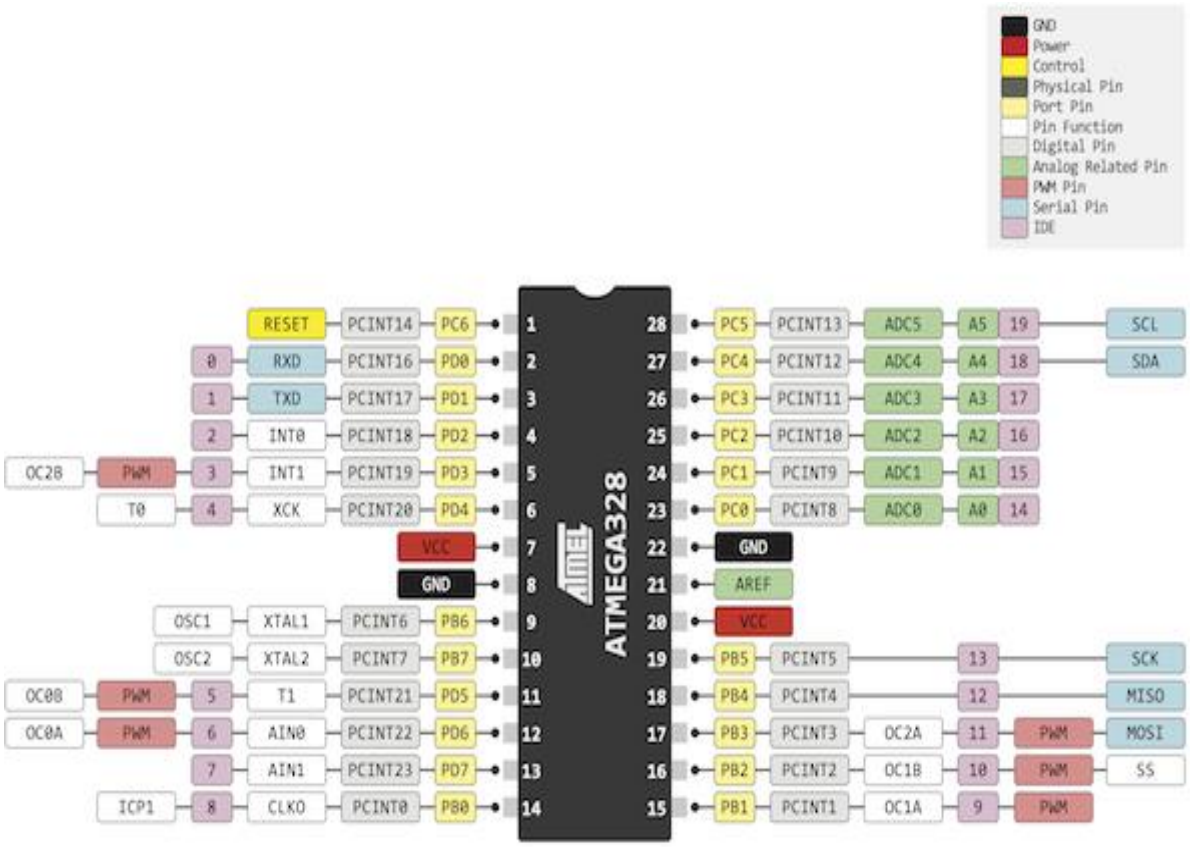


Fig.3.21. *Esquema de pines del microcontrolador ATMega328*

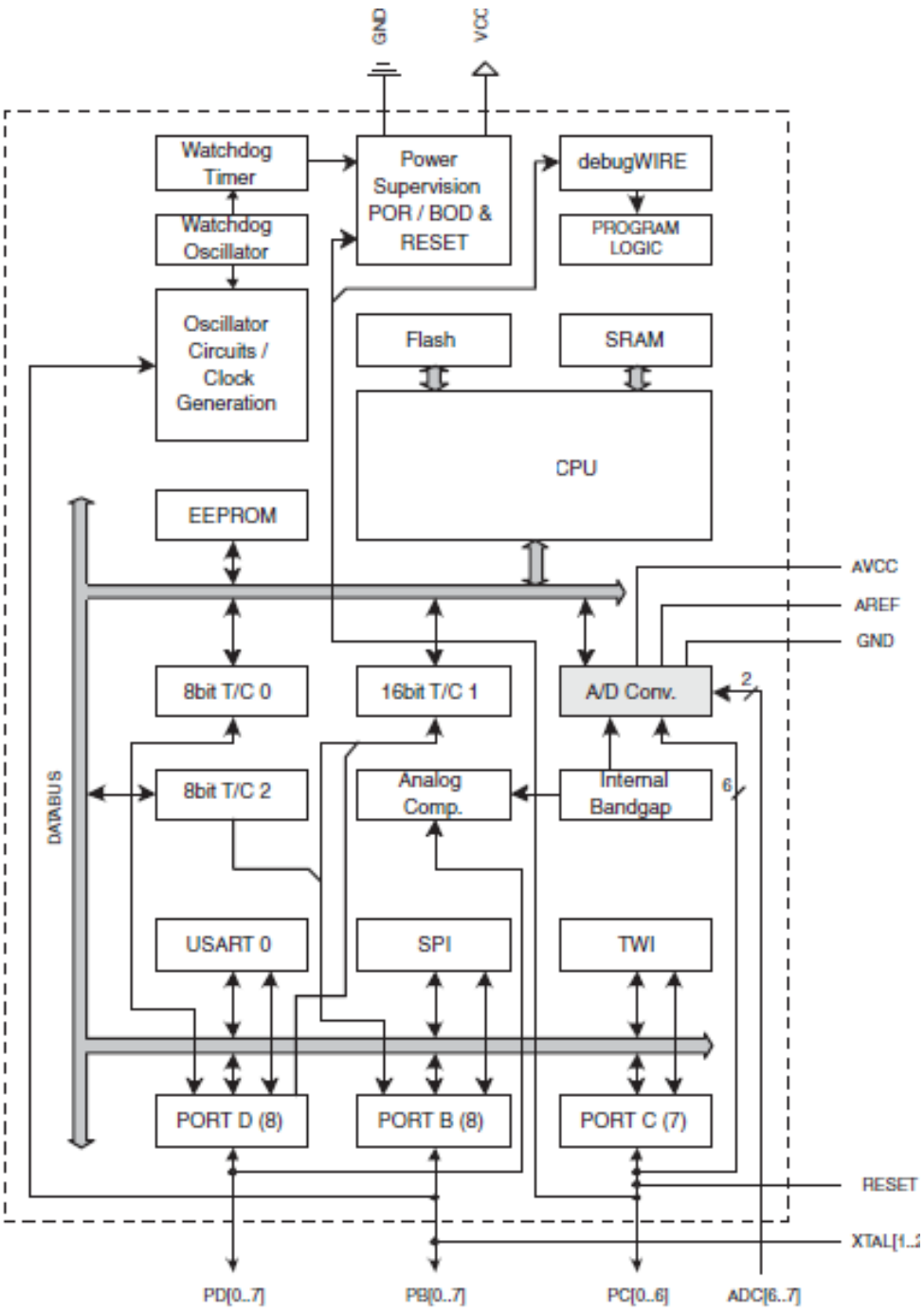


Fig.3.22. *Diagrama de bloques de procesamiento de datos en el ATMega328*

3.2.3.4. ENTRADAS Y SALIDAS

Cada uno de los 14 pines digitales en el Arduino Uno se puede utilizar como una entrada o salida, utilizando las funciones pinMode () , digitalWrite () , y digitalRead () . Funcionan a 5 voltios. Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia de pull-up de 20 a 50 kOhm.

Además, algunos pines tienen funciones especializadas:

- **Serial**: 0 (RX) y 1 (TX). Se utiliza para recibir (RX) y transmitir datos en serie (TX) TTL. Estos se encuentran conectadas a los pines correspondientes de la USB-to-TTL.
- **Interrupciones externas:** correspondientes a los pines 2 y 3 Estos pines pueden configurarse para activar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio en el valor. Se corresponde con la función attachInterrupt ().
- **PWM:** pines 3, 5, 6, 9, 10, 11 los cuales proporcionan una salida PWM de 8 bits con la función analogWrite ().
- **SPI:** pines 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estos pines admiten la comunicación SPI utilizando la librería SPI.
- **LED**: En el pin 13 hay un LED incorporado conectado al pin digital 13. Cuando el pasador es de alto valor, el LED está encendido, cuando el pasador es bajo, es apagado.

El Arduino Uno tiene 6 entradas analógicas, etiquetadas de A0 a A5, cada una de las cuales proporcionan 10 bits de resolución (es decir, 1.024 valores diferentes). Por defecto se miden desde 0 a 5 voltios, aunque es posible cambiar el extremo superior de su rango usando el pin AREF y la función analogReference (). Además también tiene algunos pines tienen funciones especializadas como son:

- **TWI**: Pin A4 o A5 o SDA
- **SCL**: Soporte del protocolo de comunicaciones I2C (TWI) usando la librería Wire.

En la placa vemos que existen también los pines de voltaje de referencia y reseteo:

- **AREF**. Voltaje de referencia para las entradas analógicas. Se utiliza con la función analogReference ().
- **Reset**. Se presiona cuando se quiere reajustar el microcontrolador. Normalmente se utiliza para añadir un botón de reinicio para los shields que bloquean la placa.

3.2.3.5. COMUNICACIÓN

El Arduino Uno tiene una serie de instalaciones para comunicarse con un ordenador, otro Arduino u otros microcontroladores.

El ATmega328 ofrece comunicación serial UART TTL (5V), que está disponible en los pines digitales 0 (RX) y 1 (TX). Un ATmega16U2 canaliza esta comunicación en serie a través de USB y aparece como un puerto COM virtual para el software en el ordenador.

El programa 16U2 utiliza los controladores USB COM estándar, y no se necesita ningún controlador externo, siin embargo, en Windows, es necesario un archivo .inf .El software de Arduino incluye un monitor de serie que permite a los datos ser enviados hacia y desde la placa Arduino.Los LEDs RX y TX de la placa parpadean cuando se están transmitiendo datos y USB a la conexión USB a serie al ordenador (pero no para la comunicación en serie en los pines 0 y 1).

Una biblioteca SoftwareSerial permite la comunicación en serial en cualquiera de los pines digitales del Uno.

El ATmega328 también es compatible I2C (TWI) y SPI. El software de Arduino incluye una librería Wire para simplificar el uso del bus I2C. Para la comunicación SPI se utiliza la librería SPI.

3.2.3.6. PROGRAMACIÓN

El Arduino Uno se puede programar con el software de Arduino. El ATmega328 en la Arduino Uno viene precargado con un gestor de arranque (bootloader) que le permite cargar nuevo código a él sin el uso de un programador de hardware externo. Se comunica mediante el protocolo original STK500 (archivos de cabecera C).

Aunque podemos obviar el gestor de arranque y programar el microcontrolador a través del ICSP (In-Circuit Serial Programming).

3.2.3.7. RESET AUTOMÁTICO

En lugar de requerir pulsar físicamente el botón de reinicio antes de que una carga, el Arduino Uno está diseñado de una manera que permite que sea restablecido por el software que se ejecuta en un ordenador conectado.

Una de las líneas de control de flujo de hardware (DTR) de theATmega8U2 / 16U2 está conectado a la línea de restablecimiento de los ATmega328 a través de un condensador de 100 nanofaradios. Cuando esta línea se verifica a nivel bajo (0V), en la línea de restablecimiento pasa el tiempo suficiente para restablecer el chip.

El software de Arduino utiliza esta capacidad para permitir que usted cargue código con sólo pulsar el botón de nivel alto en el entorno Arduino. Esto significa que el gestor de arranque puede tener un tiempo de espera más corto, ya que el descenso de DTR puede ser bien coordinado con el inicio de la subida.

Esta configuración tiene otras implicaciones. Cuando el Arduino Uno está conectado ya sea a un ordenador con Mac OS X o Linux, se restablece cada vez que se realiza una conexión a la misma desde el software (a través de USB).

Mientras que está programado para ignorar los datos malformados (como por ejemplo en la compilación de nuevo código), se interceptará los primeros bytes de datos enviados a la placa después de abrir una conexión. Si un programa habilitado en la placa recibe una configuración u otros datos cuando se inicia por primera vez hay que asegurarse de que el software con el que se comunica espera un segundo después de abrir la conexión y antes de enviar estos datos.

El Arduino Uno contiene una pista que se puede cortar para deshabilitar el reinicio automático. Los extremos a ambos lados de la pista se pueden soldar juntos para volver a habilitarlo.

Están nombrados como "RESET-ES". También puede ser capaz de desactivar el reinicio automático mediante la conexión de una resistencia de 110 ohmios de 5V a la línea de reposición.

3.2.3.8. PROTECCION USB CONTRA SOBRETENSIONES

El Arduino Uno tiene un multifusible reajustable que protege a los puertos USB de su ordenador cortocircuitos y sobretensiones.
Aunque la mayoría de los ordenadores ofrecen su propia protección interna, el fusible proporciona una capa adicional de protección.
Si hay más de 500 mA en el puerto USB, el fusible rompe automáticamente la conexión hasta que el corto o la sobrecarga desaparezcan.

3.2.3.9. CARACTERISTICAS FISICAS

La longitud máxima y la anchura del PCB del Arduino Uno son 2,7 y 2,1 pulgadas, respectivamente, con el conector USB y la clavija de alimentación que sobresalen de las dimensiones anteriores.

Cuatro orificios para los tornillos permiten que la placa pueda estar unida a casi cualquier superficie u objeto.Tenga en cuenta que la distancia entre los pines digitales 7 y 8 es de 160 milésimas de pulgada (0,16 "), no un múltiplo par de la separación de 100 milésimas de pulgada de los otros pines.

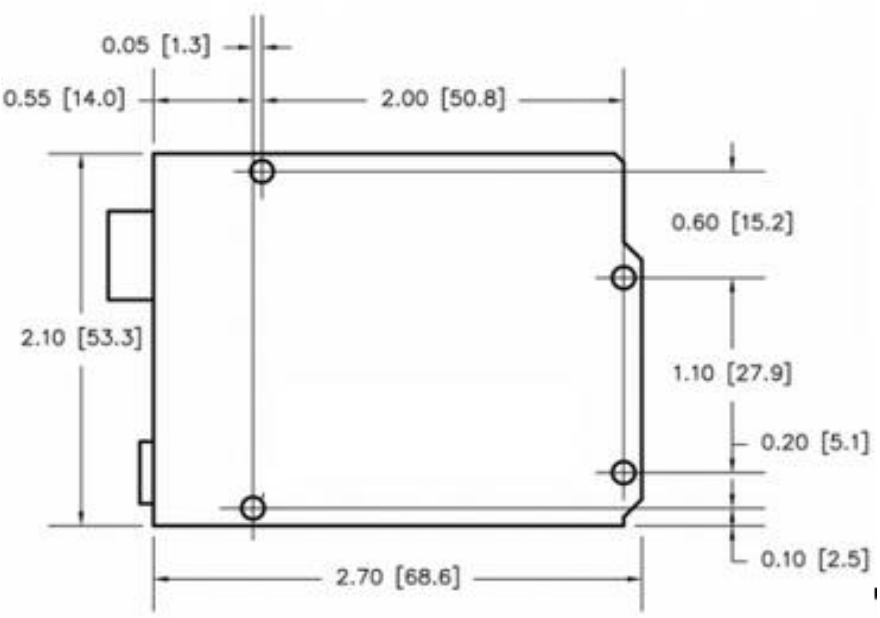


Fig.3.23. *Medidas acotadas en pulgadas de la planta de la placa Arduino Uno*

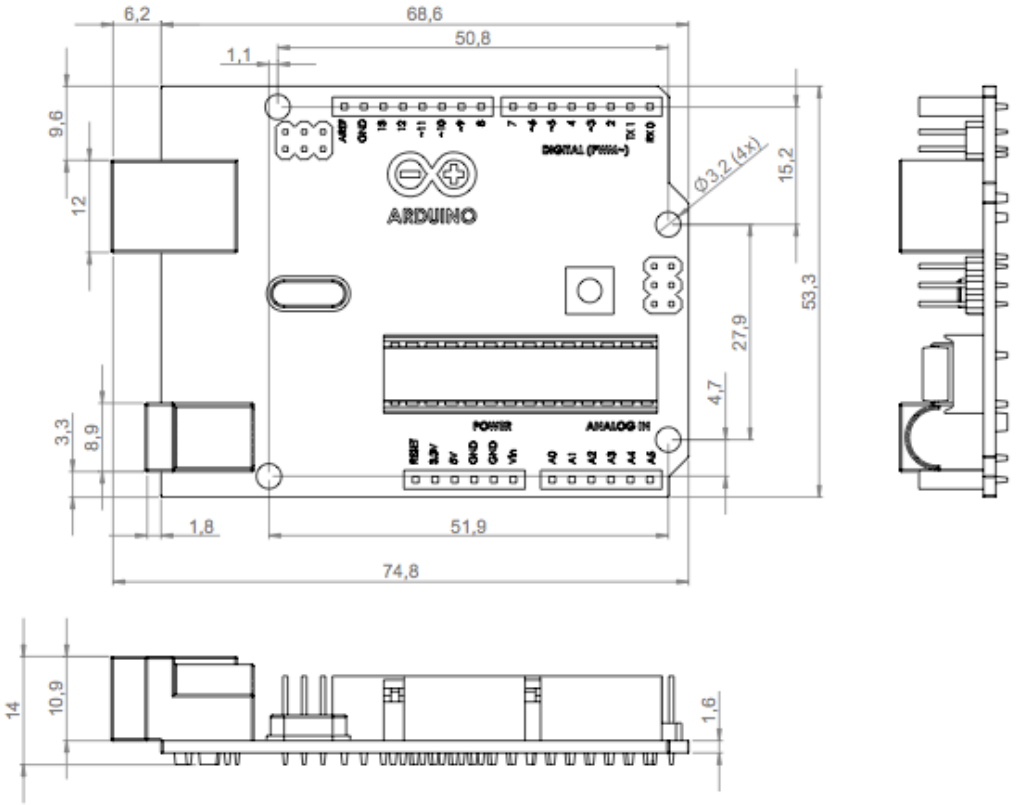


Fig.3.24. *Medidas acotadas en mm de planta, alzado y perfil de la placa Arduino Uno*

3.2.4. L293D. Inversor de giro

En la construcción del vehículo y consiguiente puesta en marcha, surge un problema, el movimiento del mismo generado por las señales del joystick, será siempre hacia adelante, ya sea girando a ambos lados o en línea recta.

Esto lleva a buscar soluciones para conseguir que el movimiento del motor del coche se invierta logrando ir hacia atrás. Para ello existen varias salidas como son la conexión-desconexión manual de los bornes del motor, ya que para invertir el sentido de giro es necesaria la inversión de polaridad, o el puente en H.

Obviamente, la primera opción es del todo inviable, se elige pues, el puente en H, en este caso, cuádruple, además no de montaje analógico, sino en su versión en microcontrolador digital; el L293D.

Debemos tener en cuenta que, al desconectar un motor, colapsa el campo magnético de las bobinas y se produce un pico de tensión de polaridad opuesta, por ello, no debe cambiarse bruscamente el sentido de giro de un motor cargado, es mejor frenarlo primero.

Capaz de conducir corrientes bidireccionales de hasta 1 amperio en el modelo L293 y hasta 600 mA en el modelo L293D y con tensiones que van desde los 4.5V hasta los 36V en ambos modelos.

Consta de dos puentes en H completos (4 medios puentes) construidos con transistores bipolares, incluye diodos para absorber la fcem al desconectar, conducen hasta 600mA continuos y 1.2A de pico.

Las entradas son de tipo TTL y se activan por parejas, es decir, desde la pata Enable 1,2EN, activamos las entradas 1 y 2 y desde la pata Enable 3,4EN activamos la 3 y la 4. Cada par de entradas forma un puente en H completo (Full-H bridge).

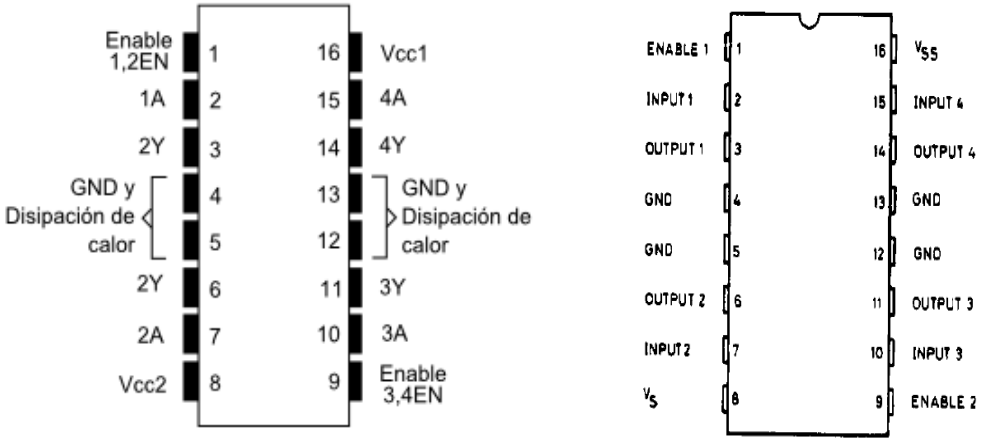


Fig.3.25. *Integrado inversor L293D*

La asignación y conexión de pines del integrado con ambos motores así como el Arduino Uno será la siguiente:

1. Enable 1; conectado a 5V
2. Input 1; conectada a Motor pin 1 lógico
3. Output 1; conectado al terminal Motor 1
4. Tierra
5. Tierra
6. Output 2; conectado al terminal Motor 2
7. Input 2; conectada a Motor pin 2 lógico
8. Vs; conectado a la alimentación del motor (ej. 5V o 9V)
9. Enable 2; conectado a 5V
10. Input 3; conectada a Motor pin 3 lógico
11. Output 3; conectada al terminal del motor 3
12. Tierra
13. Tierra
14. Output 4; conectado al terminal Motor 4
15. Input 4; conectado a Motor pin 4 lógico
16. Vss conectado a 5V

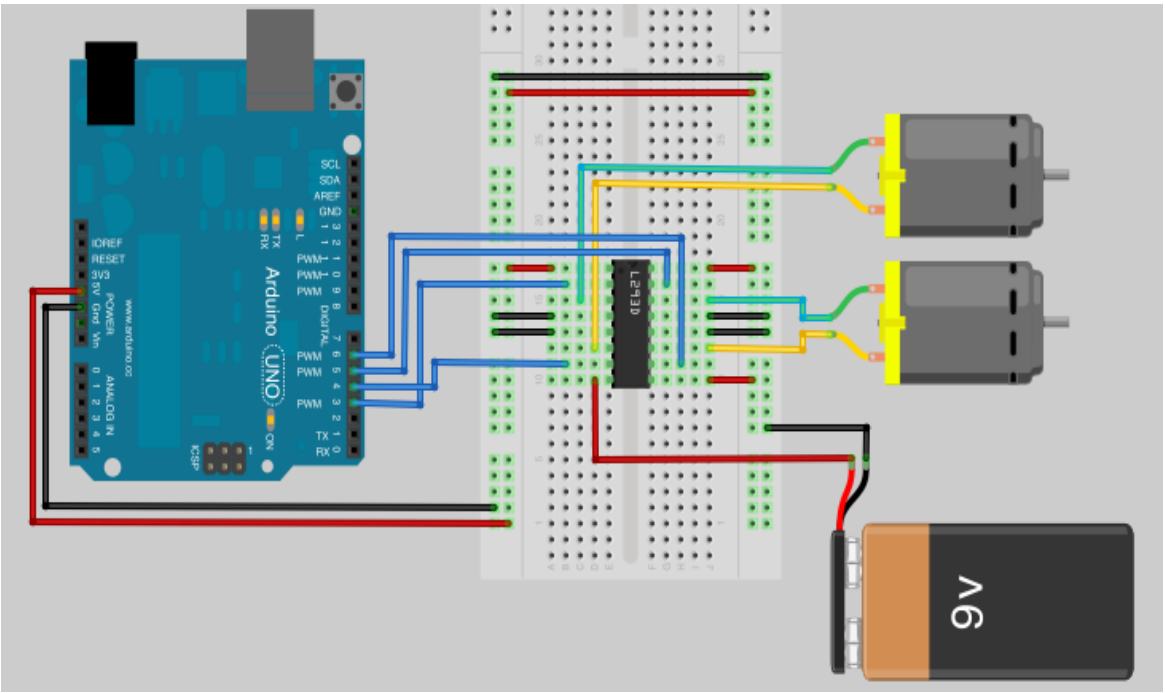


***Fig. 3.26**. Montaje de conexión Arduino Uno y motores DC con integrado L293D*

No se tienen que confundir las dos tensiones con las que se va a alimentar el integrado; una es la tensión de trabajo del propio chip Vcc1 ubicada en la patilla 16 y que debe ser superior a 4.5V y no debe superar los 7V y, otra es la tensión con la que es capaz de dirigir las cargas o motores que, en este caso como digo va de 4.5 a 36V (Patilla 8 Vcc2).

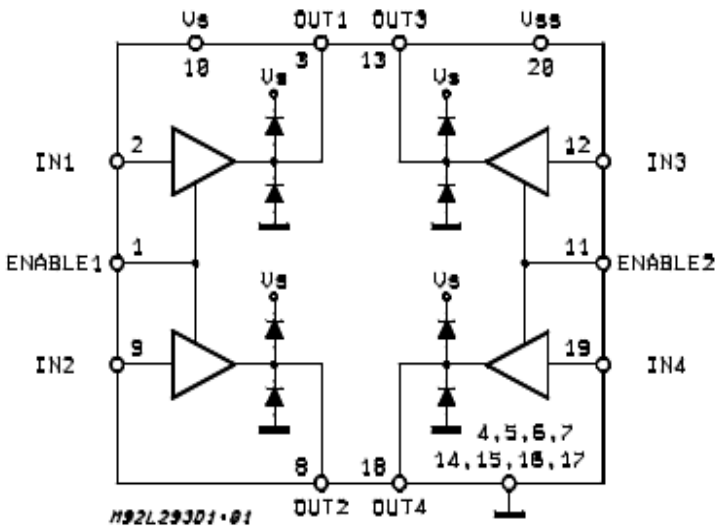
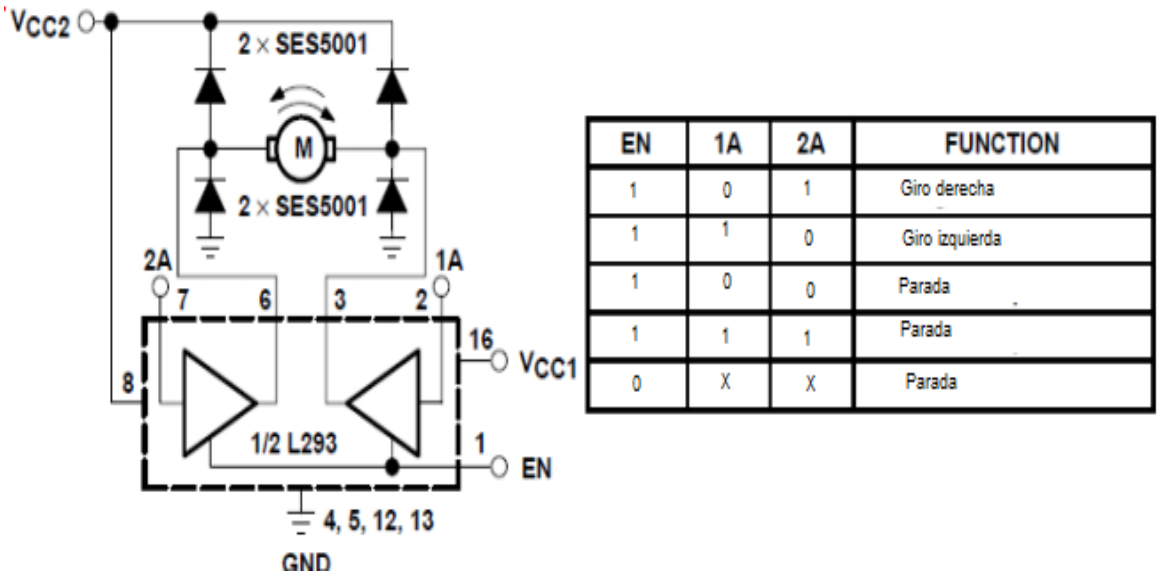


Fig. 3.27. *Esquema funcionamiento lógico de L293D*



EN	1A	2A	FUNCTION
1	0	1	Giro derecha
1	1	0	Giro izquierda
1	0	0	Parada
1	1	1	Parada
0	X	X	Parada

Fig.3.28. *Tabla de verdad según señal de E/S digitales en el L293D*

Para la construcción del puente en H y su conexión a ambos motores del coche, se hará una pequeña tarjeta donde estarán colocados y soldados, tanto el integrado como sus conexiones. El integrado irá colocado sobre un zócalo previamente soldado a dicha tarjeta ya que es muy sensible a la temperatura y podría dañarse en el proceso de soldadura directa a su patillaje.

Aunque en la tabla de verdad aparezca en una de las opciones, la entrada Enable a 0, siempre estará a 1 ya que estará conectada directamente a la alimentación de la placa por lo que se ahorrará un estado lo que facilitará el trabajo a la hora de programar el código.

Al igual que las patillas Enable, la patilla de Vss estará también conectada a la alimentación de la placa, recordando que está alimentación será de 5V.

Para las pruebas, antes de conectar y soldar todo el montaje, se conectará tanto la patilla Vss como las tierras respectivamente a la una fuente de alimentación externa, más tarde, esta fuente será sustituida por una batería de 9V colocada sobre el coche para la alimentación autónoma.

A la hora de la programación, asignaremos al joystick 4 valores de 2 bits cada uno, es decir, si se pulsa hacia arriba que será por ejemplo la primera asignación por la cual el motor se moverá hacia adelante; se tendrá 01, hacia abajo, la segunda asignación, en la que el motor se mueve hacia atrás, tendremos el 10, los valores 00 y 11 los asignaremos a la parada, en estos dos últimos estados, el motor no se accionará en ninguno de los sentidos.

Ahora bien, si se necesita un giro a la izquierda y hacia adelante, el motor izquierda se mantendrá parado mientras que el derecho girará hacia adelante, si el giro a izquierdas es hacia atrás, lo que cambia ahora únicamente será que el motor derecho girará hacia atrás.

Igualmente, si se quiere un giro a la derecha y hacia adelante, el motor derecho estará parado mientras que el izquierdo girará hacia adelante. Si el giro a derechas es hacia atrás, el motor derecho se mantendrá en parada y el motor izquierdo girará hacia atrás.

Este es el funcionamiento básico general del movimiento completo del vehículo según los valores introducidos por el joystick y codificados en el programa.

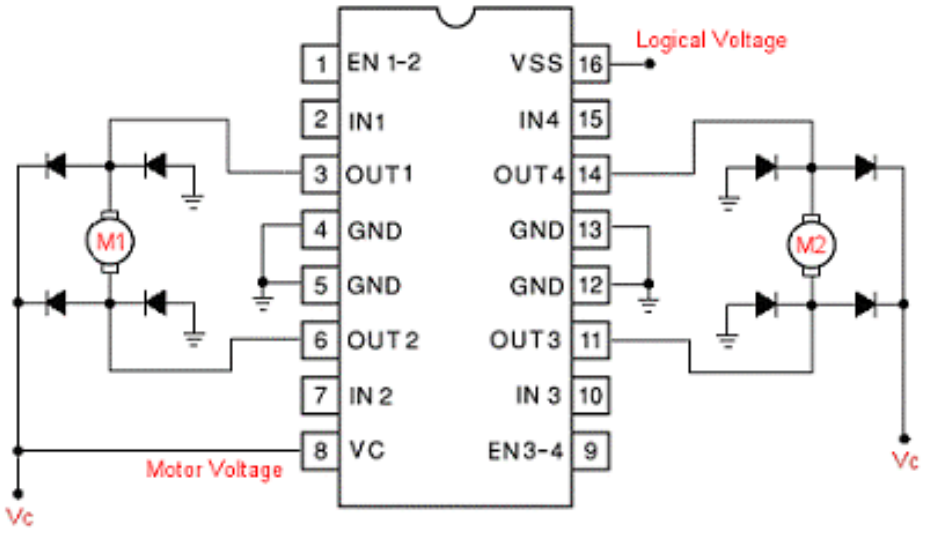


***Fig.3.29**. Conexión del L293D a ambos motores*

A la hora de implementar el circuito en la tarjeta, se podrá hacer de manera básica, o de manera más avanzada, es decir, conectando directamente las patillas del integrado a los motores a través con cableado o bien, introduciendo condensadores para la protección contra sobretensiones.

En realidad, y a no ser que se le introdujese al integrado una tensión elevada, no hay peligro de destrucción de la circuitería, como se ha comentado anteriormente, este nuevo modelo de integrado inversor, L293D, ya lleva diodos de protección.

3.3. COMUNICACION INALAMBRICA

El hacer que dos o más Arduinos sean capaces de comunicarse entre sí de forma distante a través de una red inalámbrica abre muchas posibilidades:

- Permite manejar sensores remotos de temperatura, presión, alarmas, etc…
- Comunicarse con Robots de control y vigilancia desde 10 a 500 metros de distancia
- Control remoto y monitorización de edificios cercanos y maquinaría.
- Vehículos autónomos de todo tipo.

En este apartado nos centramos en todo lo relativo a la transmisión de datos usada en el proyecto, desde el tipo de chip utilizado hasta su protocolo de transmisión.

3.3.1. El transceptor NRF24L01

Los transceptores NRF24L01 son una serie de módulos de radio de 2,4 GHz que se basan en el chip de Nordic Semiconductor nRF24L01.

El Nordic nRF24L01 integra un completo transceptor RF de 2,4 GHz, un sintetizador de RF (radiofrecuencia) y toda la lógica de banda de base incluyendo un acelerador de protocolo por hardware Enhanced ShockBurst™ con una interfaz SPI de alta velocidad para el controlador de la aplicación.

El módulo Transceptor de baja potencia y corto alcance (200 metros más o menos) está disponible en una tarjeta compatible con Arduino y con antena integrada.

El nRF24L01 es un transceptor altamente integrado de muy baja potencia (ULP) 2Mbps RF para la banda ISM de 2,4 GHz (Industrial, Científica y Médica) los cuales están optimizados para ser líder en la industria a través de modo activo, modo de reposo y modos de tiempo de despertador con picos de corriente TX / RX inferior a 14mA, un modo de apagado por debajo del µA, administración avanzada de energía, y un rango de suministro de 1,9 a 3,6 V

El nRF24L01 ofrece una verdadera solución ULP permitiendo meses o años de vida de la batería cuando se ejecuta en pilas de litio o AA / AAA.

El protocolo acelerador del hardware ShockBurst ™, además, descarga las funciones de protocolo de tiempo crítico desde la aplicación del microcontrolador habilitando la implementación de la conectividad inalámbrica avanzada y más robusta.

No se requieren filtros de bucle, resonadores, o diodos varactor VCO externos, sólo un bajo costo ± cristal 60ppm, circuitos de coincidencia, y la antena.

El Nordic nRF24L01 está disponible en un paquete QFN compacto de 4x4 mm 20-pin.

La tensión interna de los reguladores asegura una alta tensión de rechazo de alimentación (PSRR) y un amplio rango de tensión de abastecimiento.

3.3.1.1. CARACTERISTICAS

• Radiotransmisión

- Banda de operación ISM de 2.4GHz
- 126 canales RF
- Pines comunes de Rx/Tx
- Modulación GFSK
- Velocidad de transmisión de datos de 1 a 2 Mbps
- Espacio de 1MHz de canal no solapado a 1Mbps
- Espacio de 2MHz de canal no solapado a 2Mbps

• Transmisión

- Tensión de salida programable: 0, -6, -12 or -18dBm
- Tensión de salida de 11.3mA a 0dBm.

• Recepción

- Filtros de canal integrados
- 12.3mA a 2Mbps
- Sensibilidad de -82dBm a 2Mbps
- Sensibilidad de -85dBm a 1Mbps
- Ganancia LNA programable

• Sintetizador de RF

- Sintetizador completamente integrado
- Archivo de bucle no externo; VCO varactor, diodo o resonador
- Cristal de ±60ppm y 16MHz

• ShockBurst™ mejorado

- Longitud de carga dinámica de1 a 32 bytes
- Manipulación de paquetes de datos automática
- Autotransmisión de paquetes de datos
- 6 buses de datos MultiCeiver™ de 1:6

- **Gestión de la energía**
  - Regulador de tensión integrado
  - Rango de alimentación de 1.9 a 3.6V
  - Modo inactivo de reinicio rápido para el mantenimiento avanzado de la energía
  - Modo Stand-by a 22uA
  - Modo inactivo a 900nA
  - Maximo de 1.5ms de inicio rápido desde modo inactivo
  - Maximo de 130us de inicio rápido desde modo stand-by

- **Interfaz de Host**
  - Hardware SPI de 4 pines
  - Maximo 8Mbps
  - 3 Tx de 32 bytes separados y FIFOs Rx
  - Entradas de alimentación de 5V

- **Compact 20-pin 4x4mm QFN package**

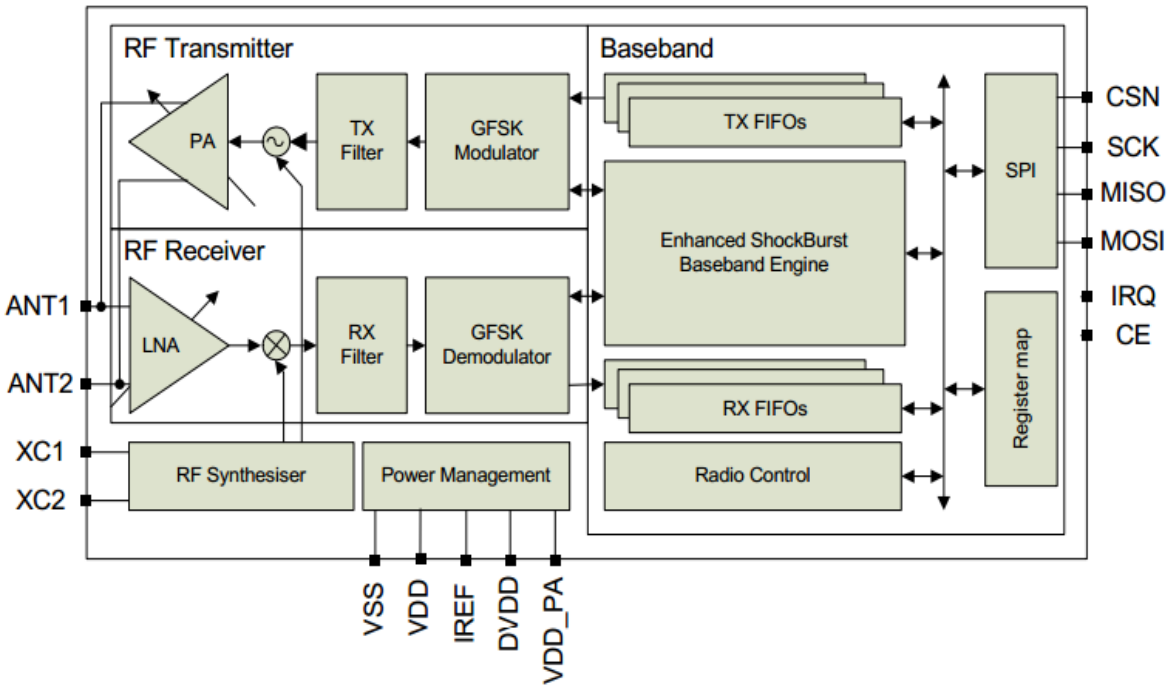


Fig.3.30. *Diagrama de bloques del nRF24L01*

3.3.1.2. ASIGNACION DE PINES

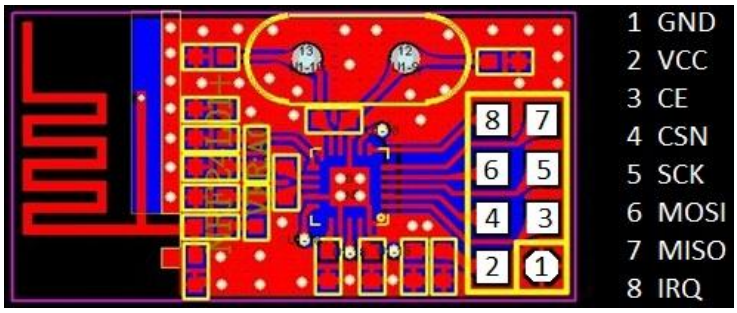


***Fig.3.31**. Layout de la planta (vista superior) del Nrf24L01 con asignación de pines*

- **CONEXIÓN CON EL ARDUINO**

Signal	MODULO RF	COLOR	PIN ARDUINO LIBRERIA RF24	PIN ARDUINO LIBRERIA MIRF	MEGA2560 pin
GND	1	Negro	GND	GND	por librería
VCC	2	Rojo	3.3V	3.3V	por librería
CE	3	Amarillo	9	8	por librería
CSN	4	Verde	10	7	por librería
SCK	5	Azul	13	13	52
MOSI	6	Naranja	11	11	51
MISO	7	Violeta	12	12	50
IRQ	8	–	2 *		por librería

***Fig.3.32.** Pines para conexión en el módulo Nrf24L01*

```
Arduino UNO
         3V3 or 5V----VCC (3.3V to 7V in)
     pin D8-----------CE (chip enable in)
  SS pin D10----------CSN (chip select in)
 SCK pin D13----------SCK (SPI clock in)
MOSI pin D11----------SDI (SPI Data in)
MISO pin D12----------SDO (SPI data out)
         IRQ (Interrupt output, not connected)
         GND----------GND (ground in)
```

***Fig. 3.33**. Pines para conexión en el Arduino Uno rev3*

Vemos que el Pin 8 IRQ no se utiliza por la mayoría del software, pero la biblioteca RF24 tiene un ejemplo que lo utiliza.

El pin VCC debe ir conectado a 3.3V no a 5V, aunque el propio Arduino puede funcionar a 5V y la señal I/O va a funcionar bien. Arduino UNO y versiones anteriores tienen una salida de 3,3 V que se puede utilizar para la versión de baja potencia de estos módulos, pero las versiones de alta potencia deben tener un suministro de 3,3 V por separado.

Hace falta un regulador de 3.3V con potencia más alta para alimentar el módulo con amplificador de emisión, no podemos conectarlo directamente al Arduino.

La conexión física entre el nRF24L01 y Arduino requiere 3,3 V, los 3 x pines SPI (SCK, SDI, SDO), un pin Chip Enable y un pin para Slave Selec.

A continuación vemos el esquema de conexión de pines entre el módulo RF Nrf24L01 y el Arduino Uno rev3 utilizados en nuestro proyecto.

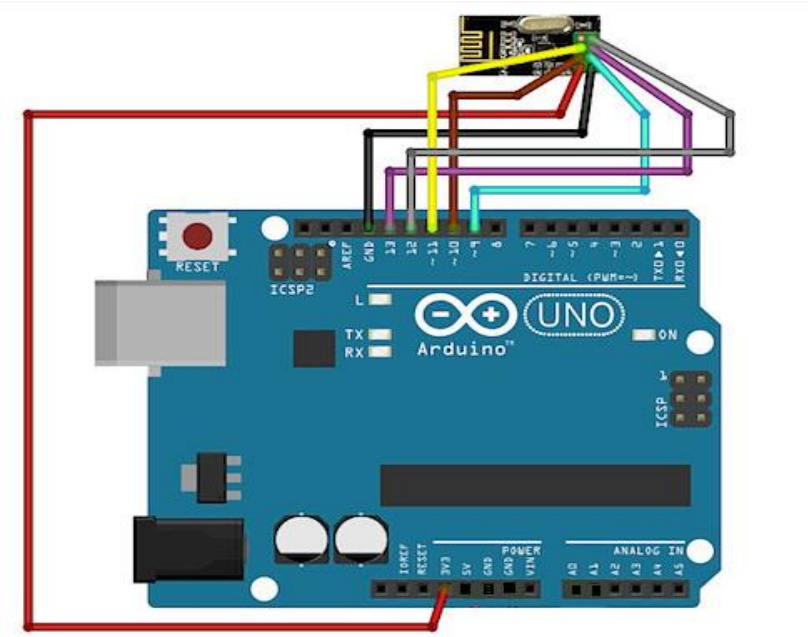


***Fig.3.34**. Vista del conexionado completo de la placa Arduino con el módulo nRF24L01*

3.3.1.3. RANGO COBERTURA

El rango 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. Pero este valor es para espacio abierto entre unidades funcionando a 250KHz, en interiores, el alcance es mucho menor debido a las paredes, etc.

Hay unidades con un preamplificador de antena para el receptor y un amplificador de potencia para el transmisor y cuentan con antena externa. La comunicación entre una unidad de este tipo y varias unidades de bajo consumo producirá mucho mejor resultado que utilizar dos unidades de baja potencia.

Hay módulos adicionales que se añaden al transmisor como amplificadores de potencia y preamplificadores al receptor para conseguir distancias más largas, aseguran que pueden llegar hasta 1 km.

Estos módulos utilizan una antena externa que puede ser una antena simple que esté directamente conectada o un cable conectado a una antena con más ganancia o directividad.

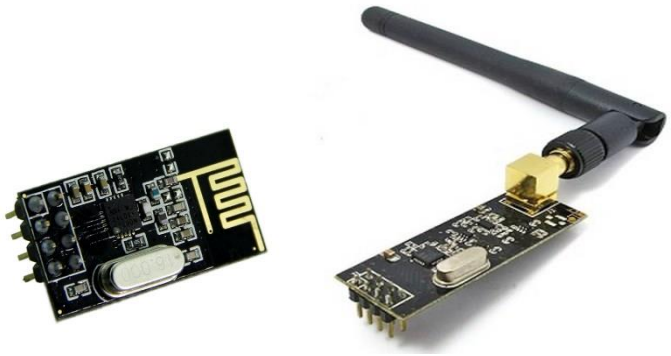


***Fig.3.35**. Nrf24l01 con antena en Zigzag (izda.) y con amplificador de potencia en transmisión (dcha.)*

3.3.1.4. PARAMETROS DE CONTROL

En este apartado se describirán los diferentes modos en los que el módulo de radio transmisor puede operar así como los parámetros utilizados en cada uno de ellos.

El Nrf24L01 ha sido construido como una máquina de estados que controla las transiciones entre los diferentes modos de operación del módulo. La máquina de estados toma las entradas sobre el registro de valores definidos por el usuario así como de señales internas.

El Nrf24l01 puede ser configurado de 4 maneras principales; a continuación, describiremos brevemente cada una de ellas partiendo desde una visión global de los diferentes modos visto en un diagrama de estados.

El diagrama de estado mostrará los modos de operación y acceso a ellos. El Nrf24L01 no está definido hasta que VDD alcanza un valor igual o mayor a 1.9V.

Cuando esto ocurre, el módulo pasa de power(activo) a Reset donde permanece hasta que se cambia a modo inactivo (power down). Incluso cuando el nrF24L01 entra en este modo inactivo, el MCU puede controlar el chip a través del SPI junto con el pin Enable del chip dando lugar a tres tipos de estados;

- “Recommended operating mode” o modo de operación recomendable; estado usado normalmente durante una operación.

- “Possible operating mode” o modo de operación posible; está permitido usar este modo como estado pero no durante una operación normal.

- “Transition state” o estado de transición; es un estado de tiempo limitado usado durante el inicio de ejecución del oscilador y el establecimiento del PLL.

Vemos lo descrito anteriormente de manera más clara en el diagrama de estado que se muestra a continuación;

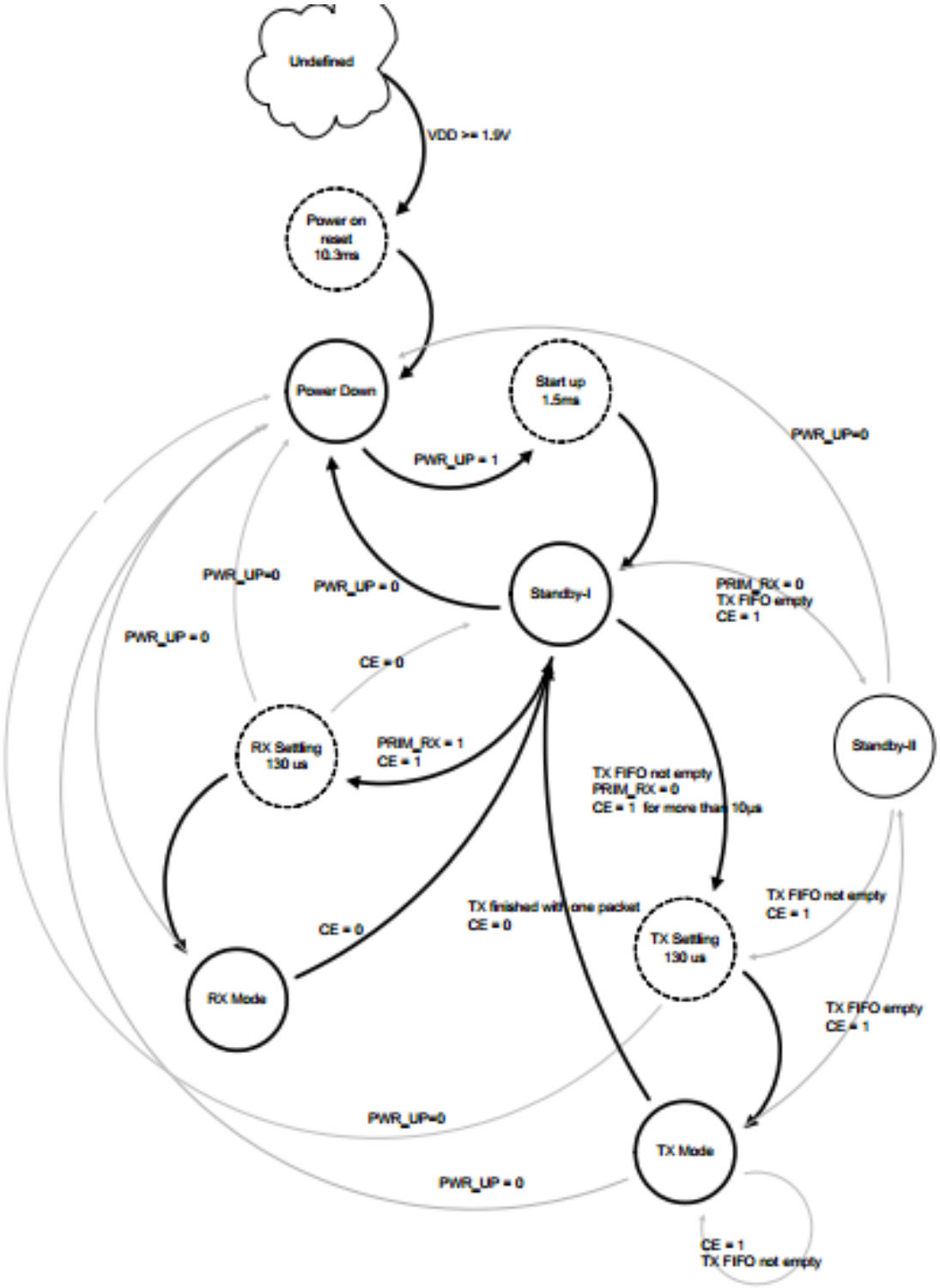
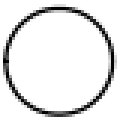
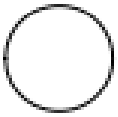
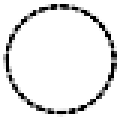


Fig.3.36. *Diagrama de estados del control por radio en el módulo Nrf24L01*

Donde tenemos que;

	Estado indefinido
	Modo de operación recomendado
	Modo de operación posible
	Estado de transición
	Vía recomendada entre los modos de operación
	Posible vía alternativa entre los modos de operación
CE=1	Condición del pin de señal
PWR_DN=1	Condición del bit de estado
Tx FIFO empty	Sistema de información

Los 4 modos de operación aludidos anteriormente son:

- POWER DOWN: el módulo se encuentra deshabilitado con la mínima corriente. Todos los valores del registro disponibles desde el SPI son mantenidos y el SPI puede ser activado
- STAND-BY: programando el bit PWR_UP en la CONFIG el dispositivo entrara en el modo de stand-by usado para minimizar el consumo de corriente media mientras ocurren inicializaciones. En este modo es cuando el cristal oscilador se activa y donde el Nrf24l01 vuelve de Tx o Rx cuando CE está establecido a nivel bajo.
- MODO RX: es un modo activo donde el módulo actúa como receptor. Para acceder a este modo el módulo debe tener el bit PWR_UP y PRIM_RX a nivel alto así como el pin CE. En este modo, se demodulan las señales del canal RF ya que el protocolo de banda base está buscando constantemente paquetes de datos válidos, si es encontrado, comprobando la dirección y validando el CRC, la información es llevada al espacio existente en el FIFO del Rx, si el FIFO está lleno, esta información será descartada.
  El Nrf24L01 mantiene el modo Rx hasta que el MCU lo configura en el modo Stand-by o en Power Down.
  En el modo Rx se detecta una señal disponible puesta a nivel alto cuando una señal de RF es detectada dentro del rango de la frecuencia de recepción del canal. La señal debe ser modulada mediante FSK para una detección segura.
  El Carrier Detect (CD) es puesto a nivel alto cuando una señal de RF es detectada en el módulo Rx, esta señal de RF debe estar presente al menos 128µs antes de que el CD sea puesto a 1 (nivel alto).

- MODO TX: es un modo activo en el que el Nrf24L01 transmite un paquete de datos. Para activar este modo, el módulo debe de tener a nivel alto el bit PRIM_UP y establecer a nivel bajo el bit PRIM RX, una carga en el FIFO TX y un pulso a nivel alto en el bit CE durante más de 10µs.

  El Nrf24L01 permanece en este modo hasta que finaliza la transmisión. Si el bit CE=0, el módulo vuelve al modo de Stand-by, si CE=1, la siguiente acción es determinada por el estado del FIFO TX; si éste no está vacío, el módulo permanece en modo Tx transmitiendo el siguiente paquete de datos, si, por el contrario, estuviera vacío, regresaría al modo Stand-by.
  El transmisor PLL del módulo nRF24L01 opera en bucle abierto en el modo Tx.

  Es importante no dejar nunca el módulo en modo Tx durante más de 4ms cada vez.
  Si la autoretransmisión está habilitada, el Nrf24L01 nunca estará en modo Tx el tiempo suficiente como para desobedecer esta norma.

3.3.1.5. COMUNICACIÓN SERIAL

La comunicación serial consiste en el envío de un bit de información de manera secuencial, esto es, un bit a la vez y a un ritmo acordado entre el emisor y el receptor.

La comunicación serial en computadores ha seguido los estándares definidos en 1969 por el RS-232 (Recommended Standard 232) que establece niveles de voltaje, velocidad de transmisión de los datos, etc. Por ejemplo, este protocolo establece un nivel de -12v como un uno lógico y un nivel de voltaje de +12v como un cero lógico (por su parte, los microcontroladores emplean por lo general 5v como un uno lógico y 0v como un cero lógico).

Existen en la actualidad diferentes ejemplos de puertos que comunican información de manera serial (un bit a la vez). El conocido como “puerto serial” ha sido gradualmente reemplazado por el puerto USB (Universal Serial Bus) que permite mayor versatilidad en la conexión de múltiples dispositivos. Aunque en naturaleza serial, no suele referenciarse de esta manera ya que sigue sus propios estándares y no los establecidos por el RS-232.

La mayoría de los microcontroladores, entre ellos Arduino, poseen un puerto de comunicación serial. Para comunicarse con los computadores personales actuales que poseen únicamente puerto USB requieren de un dispositivo “traductor”. Arduino emplea el integrado FT232R, el cual es un convertidor USB-Serial. A través de este integrado el microcontrolador puede recibir y enviar datos a un computador de manera serial.

La parte física encargada de la comunición serial es la UART (Universal Asynchronous Receiver and Transmitter). Los microcontroladores Atmega8/168/328, en los cuales está basado Arduino, disponen de un dispositivo compatible llamado USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) que permite tanto la comunicación asincrónica como sincrónica.

En la comunicación asincrónica, la velocidad de envío de los datos es acordada a priori entre el emisor y el receptor. En la comunicación sincrónica, el envío de los datos es sincronizado por el emisor a partir de un pulso constante de reloj (Clock), con cada pulso envía un nuevo dato.

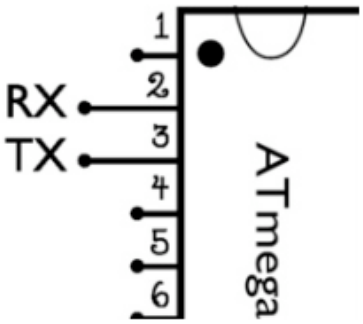


***Fig.3.37**. Pines de comunicación serial en los microcontroladores Atmega8/ 168/328*

En la comunicación serial, el puerto serial envía y recibe bytes de información un bit a la vez. Aun y cuando esto es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias. Por ejemplo, la especificación *IEEE 488* para la comunicación en paralelo determina que el largo del cable para el equipo no puede ser mayor a 20 metros, con no más de 2 metros entre cualesquier dos dispositivos; por el otro lado, utilizando comunicación serial el largo del cable puede llegar a los 1200 metros.
Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII.

Las características más importantes de la comunicación serial son la velocidad de transmisión, los bits de datos, los bits de parada, y la paridad. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales:

a. **Velocidad de transmisión *(baud rate)*:** Indica el número de bits por segundo que se transfieren, y se mide en baudios (*bd*). Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión. Cuando el protocolo hace una llamada a 4800 ciclos de reloj, entonces el reloj está corriendo a 4800 Hz, lo que significa que el puerto serial está muestreando las líneas de transmisión a 4800 Hz. Las altas velocidades se utilizan cuando los dispositivos se encuentran uno junto al otro.

b. **Bits de datos:** Se refiere a la cantidad de bits en la transmisión. El número de bits que se envía depende en el tipo de información que se transfiere. Por ejemplo, el ASCII estándar tiene un rango de 0 a 127, es decir, utiliza 7 bits; para ASCII extendido es de 0 a 255, lo que utiliza 8 bits. Si el tipo de datos que se está transfiriendo es texto simple (ASCII estándar), entonces es suficiente con utilizar 7 bits por paquete para la comunicación. Un paquete se refiere a una transferencia de byte, incluyendo los bits de inicio/parada, bits de datos, y paridad. Debido a que el número actual de bits depende en el protocolo que se seleccione, el término paquete se usa para referirse a todos los casos.

c. **Bits de parada:** Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes

d. **Paridad:** Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico.

Por ejemplo, si la información a transmitir es 011 y la paridad es par, el bit de paridad sería 0 para mantener el número de bits en estado alto lógico como par. Si la paridad seleccionada fuera impar, entonces el bit de paridad sería 1, para tener 3 bits en estado alto lógico.

La paridad marcada y espaciada en realidad no verifican el estado de los bits de datos; simplemente fija el bit de paridad en estado lógico alto para la marcada, y en estado lógico bajo para la espaciada. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizados.

3.3.2. Interfaz SPI

Es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. Se basa en la comunicación síncrona.
El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj (comunicación sincrónica).

Es un estándar establecido por Motorola que utiliza un bus de 4 líneas para interconectar dispositivos periféricos de baja y media velocidad. La comunicación se realiza siguiendo la estructura de un modelo maestro/esclavo donde el maestro selecciona al esclavo y comienza el proceso de transmisión/recepción de información.
SPI constituye un bus full dúplex, es decir, que se puede enviar y recibir información de manera simultánea, lo cual, eleva la tasa de transferencia de los datos. Aquí no existe ningún medio de direccionamiento de los dispositivos esclavos y mucho menos reconocimiento de la recepción de los datos, así que, no existe forma alguna para que el dispositivo maestro pueda detectar la presencia de un esclavo o establecer un medio de control del flujo de datos.

Se definen dos parámetros CPOL (Clock Polariity) y CPHA (Clock Phase) con los cuales se determina, con respecto al pulso del reloj, el momento en el cual se considera válido un dato de entrada o se genera un dato de salida. Estos dos parámetros ofrecen hasta cuatro combinaciones distintas las cuales constituyen los cuatro modos de trabajo del bus SPI.
Para lograr una comunicación exitosa cada par maestro-esclavo dentro del bus tiene que operar en el mismo modo, el cual, está definido generalmente por el dispositivo esclavo.

Incluye una línea de reloj, dato entrante, dato saliente y un pin de select, que conecta o desconecta la operación del dispositivo con el que uno desea comunicarse. De esta forma, este estándar permite multiplexar las líneas de reloj.

El SPI es un protocolo síncrono. La sincronización y la transmisión de datos se realizan por medio de 4 señales:

- **SCLK** (Clock): Es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit.
- **MOSI** (Master Output Slave Input): Salida de datos del maestro y entrada de datos al esclavo.
- **MISO** (Master Input Slave Output): Salida de datos del esclavo y entrada al maestro.
- **SS/Select:** Para seleccionar un esclavo, o para que el maestro le diga al esclavo que se active.

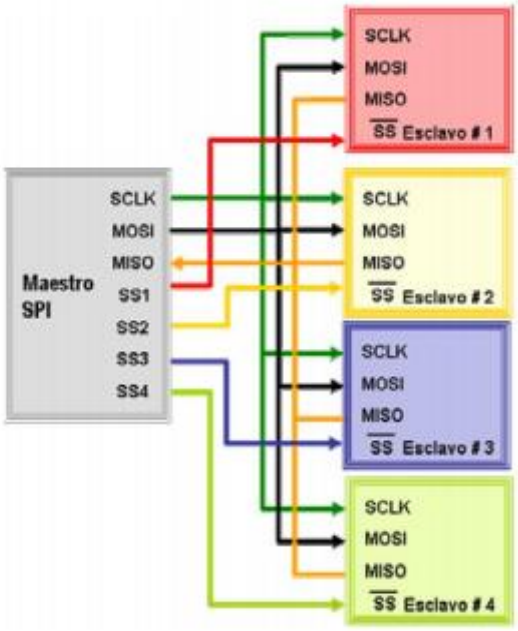


Fig.3.38. *Conexionado en el estándar de comunicación síncrona SPI*

El SPI maestro (servidor) inicializa el ciclo de comunicación cuando se coloca a nivel bajo el Selector de esclavo (SS-Selector Slave)o cliente.

Maestro y esclavo preparan los datos para ser enviados a sus respectivos registros de desplazamiento y el maestro genera el pulso del reloj en el pin SCK para el intercambio de datos. Los datos son siempre intercambiados desde el Maestro al Esclavo en MOSI, y desde Esclavo al Maestro en MISO.

Después de cada paquete de datos el Maestro debe sincronizar el esclavo llevando a 'alto' el selector de Esclavo, SS.

- Cuando se configure como Maestro, la interfaz SPI no tendrá un control automático de la línea SS. Este debe ser manejado por el software antes de que la comunicación pueda empezar; escribiendo un byte en el registro de la SPI comienza el reloj de la SPI, y el hardware cambia los 8 bits dentro del Esclavo. Después de cambiar un Byte, el reloj del SPI habilita el fin de la transmisión (SPIF).

  Si la interrupción del SPI está habilitada (SPIE) en el registro SPCR, una interrupción es requerida. El maestro podría continuar con el cambio del siguiente byte escribiendo dentro del SPDR, o señalizando el fin del paquete colocando en alto el Esclavo seleccionado, SS. El último byte llegado se mantendrá en el registro Buffer para luego usarse.

- Cuando lo configuramos como un Esclavo, la interfaz ISP permanecerá en stand-by con MISO en tres-estados siempre y cuando el pin SS este deshabilitado. En este estado mediante el software se podría actualizar el contenido del registro SPDR, pero los datos no serán desplazados por la llegada del pulso de reloj en el pin SCK hasta que el pin SS no sea habilitado (‘0’). Será visto como un byte completamente desplazado en el fin de la transmisión cuando SPIF se habilite.

  Si la interrupción SPI está habilitada, una interrupción es solicitada. El Esclavo podría continuar colocando nuevos datos para ser enviados dentro del SPDR antes de seguir leyendo los datos que va llegando. El último byte que entra permanecerá en el buffer para su posterior uso.

El MSTR en SPCR puede ser modificado por el usuario pudiendo determinar la dirección del pin SS.
Si SS es configurado como salida, el pin es una salida general la cual no afecta el sistema SPI ya que típicamente, el pin SS será manejado desde el Esclavo.
Si es como entrada, este debe ser enviado a nivel alto para asegurar la operación SPI del maestro.

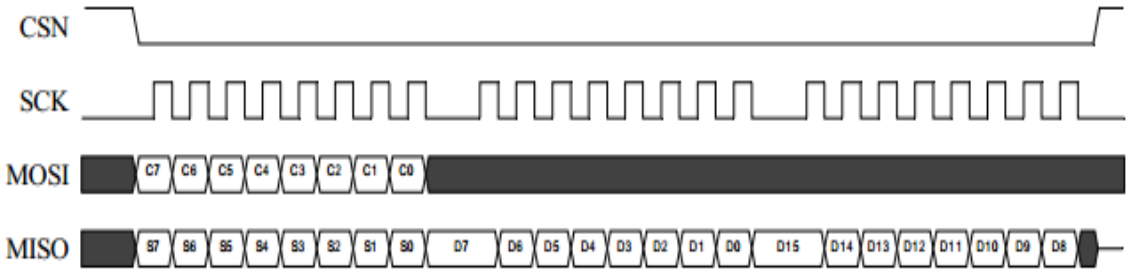


***Fig.3.39**. Operación de lectura SPI*

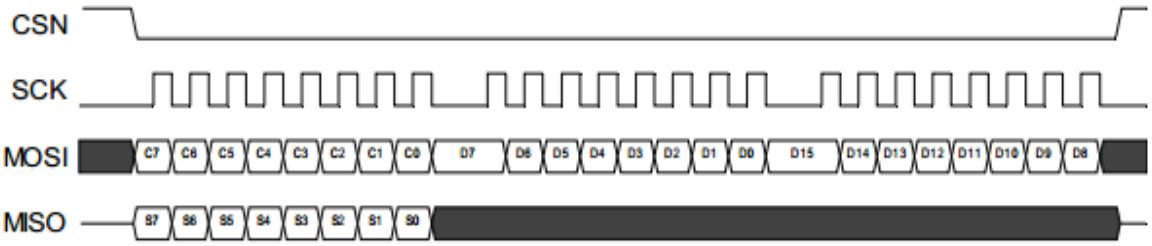


***Fig.3.40**. Operación escritura SPI*

3.3.2.1. TOPLOGIA MAESTRO-ESCLAVO

Existen tres tipos de arquitecturas básicas de comunicación que determinan cómo un nodo de una red se comunica con otro dentro de la misma red; maestro-esclavo, punto a punto (p2p) y Cliente/Servidor.

En este caso, trabajaremos con comunicación serie de arquitectura maestro-esclavo. Describiremos esta arquitectura y sus características a continuación.

En la arquitectura maestro-esclavo existe una relación donde un simple nodo ("maestro") inicia y controla una sesión con uno o más dispositivos ("esclavos"). Originalmente diseñado para redes de computadoras mainframe dónde la mainframe era la computadora maestra y las terminales "tontas" eran las esclavas.

La arquitectura maestro/esclavo no es muy comúnmente usada en redes modernas excepto en casos aislados (por ejemplo, emulación de terminal).

3.3.2.2. COMUNICACIÓN FULL-DUPLEX

Al contar con varios microcontroladores esclavos, se necesita que cada uno de ellos este reportando los datos obtenidos de cada proceso, sin embargo, como no se sabe cuándo se necesitará dicha información, se requieren de dos canales, uno independiente del otro, para podrá transmitir y recibir al mismo tiempo la información.

El término Full Duplex se refiere a que un sistema puede transmitir y recibir información simultáneamente. Existen dos frecuencias una para transmitir y otra para recibir.
El nodo 1 es el maestro, por lo tanto tiene el control de la red y el asigna el permiso para transmitir. Un par de cables están conectados del nodo trasmisor Maestro a todos los controladores receptores esclavos. En el otro sentido, un par de cables conectan a todos los esclavos al receptor del Maestro. Todos los esclavos deben leer lo que el maestro envía, pero solo uno va a poder responder y lo hace a través de los cables opuestos.

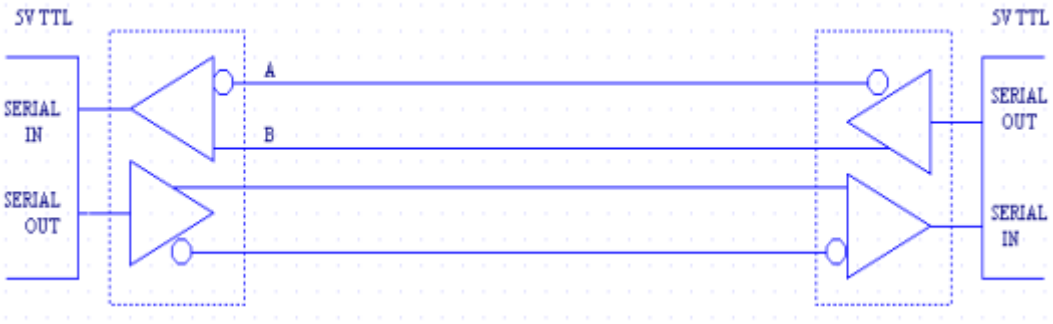


Fig.3.41. *Comunicación en Full-Dúplex*

3.3.3. Protocolo mejorado ShockBurst™

La familia nRF24 de Nordic tiene dos modos de funcionamiento diferentes, modo directo y el modo ShockBurst.

La idea con la tecnología ShockBurst ™ es poner la mayor nivel bajo manejo en el chip NRF posible protocolo sin quitar ninguna flexibilidad del usuario.

Aunque los dispositivos nRF240 utiliza una tasa de bit de 1 Mbit / s en la transmisión, es posible, para un microcontrolador de bajo costo, manejar y operar un sistema de RF basado en estos dispositivos.

Un microcontrolador de bajo costo puede utilizar el dispositivo nRF240 como un "registro avanzado" donde los datos que se transmiten sólo tienen que ser transmitidos a una velocidad establecida por el micro-controlador.

En modo de recepción es aún más simple, el microcontrolador será notificado por el nRF24 cuando un paquete válido ha llegado, pudiendo entonces tener los datos su propia velocidad de reloj. En ambos casos no hay necesidad de una sincronización precisa o la operación de alta velocidad. Este le dará al usuario la posibilidad de utilizar un microcontrolador de bajo coste con un oscilador interno RC, eliminando la necesidad de un cristal externo en el microcontrolador.

Desde el nRF2401 se está haciendo todo manejo con protocolo a nivel bajo, como el muestreo de bits, comprobación de direcciones y el cálculo de suma de comprobación; esto le llevará de nuevo al hecho de que el microcontrolador puede ejecutar en una velocidad aún más baja para desempeñar sus funciones, por lo que es posible elegir entre los microcontroladores más baratos que hay.

Existen dos ventajas principales que nos ofrece el ShockBurst como son:

- Bajo consumo de corriente

Si un microcontrolador está limitado a los datos de reloj de entrada y salida a una velocidad de 10kbit/s, es colocado en un diseño junto con una radio y esta radio tiene un consumo de corriente de 8mA durante la transmisión, vamos a calcular la energía por bit transmitido que este sistema utiliza: 8mA / 10kbit/s = 0.8uAs / bit.

Vamos a continuación, a colocar el mismo microcontrolador junto con un dispositivo nRF240 en Modo ShockBurst ™: El nRF240x tiene una velocidad de transmisi1 Mbit/s. Desde que el nRF240 se queda en modo de espera cuando el paquete se registró en su Registro de salida de datos, sólo utilizará corriente al transmitir el paquete.

El dispositivo nRF24L01utiliza 8mA en su modo de transmisión, pero a partir de que la tasa de bits transmitida es de 1 Mbit/s se usarán sólo 8mA / 1 Mbit/s = 8nAs / bit, es decir centésimas veces la energía utilizada por el primer diseño.

Esto significa que el diseño con el dispositivo nRF240x será capaz de utilizar una batería 100 veces más tiempo que el otro diseño transmitiendo la misma cantidad de datos.

Además del hecho de que el nRF240x utilizará muy poca energía por bit transmitido, también se ahorrará consumo de corriente debido al hecho de que el microcontrolador puede hibernar cuando la nRF240x está recibiendo o transmitiendo datos. Esto no es posible sin la Tecnología ShockBurst ™

- Bajo coste

Desde que la tecnología ShockBurst ™ hace posible el uso de un micro de bajo coste controlador con sólo un oscilador RC interno, la necesidad de componentes externos alrededor el microcontrolador será mínima. El nRF240en sí también tiene muy pocos componentes externos, por lo que la factura total de material serán extremadamente baja.

Dado que el ShockBurst es configurable, el módulo Nrf24l01 puede trabajar en dos modos, ya comentados anteriormente:

- **Modo transmisor**

1. El microcontrolador de bajo coste externo será configurará primeramente el dispositivo nRF24 para utilizar el canal derecho, potencia de salida y si se va a utilizar cíclico redundante Marque (CRC) o no.

2. El microcontrolador decide que va a transmitir un paquete a través del dispositivo nRF240, y por lo que habilita en chip a nivel alto.

3. El microcontrolador instaura su paquete de datos en el registro de salida de datos, utilizando el Reloj y el pin de datos en el nRF240x. El paquete contiene la dirección del destinatario y la carga útil.

4. Si CRC está activada, el SBE calculará el CRC de los datos que están siendo enviados a la velocidad del reloj, y el resultado se adjunta al paquete transmitido.

5. La transmisión del paquete comenzará 202us después de que la señal de habilitación haya sido puesta a nivel bajo por el microcontrolador.

- **Modo receptor**

1. El microcontrolador configurará el nRF2401 antes de cualquier operación. Durante esta configuración, al nRF2401 se le asignará lo que es su propia dirección, la longitud del paquete que deberá recibir, la tasa de bits (250kbit/s o 1 Mbit/s) es deberá utilizar en la transmisión y si desempeñará CRC o no. Después configuración del nRF2401, el microcontrolador puede ir a hibernación.

2. En este momento un transmisor comienza a transmitir un paquete a este receptor.

3. En el demodulador nRF2401 los bits están sincronizados en un registro FIFO de 256 bits de ancho registrados con la misma velocidad de bits que se utiliza en transmisión.

4. Cada vez que un nuevo bit ingresa en el registro FIFO del demodulador, los N primeros bits de la FIFO se compararán con su propia dirección.

5. Si los bits coinciden con el CRC, se calculará la suma de comprobación de la totalidad del paquete y se comparará si el resultado es igual a los bits de CRC en el paquete recibido. Si el CRC no coincide, la SBE continuará comparando los bits entrantes con propia dirección hasta que una nueva correspondencia haya sido encontrada, y luego se repetirá el cálculo del CRC.

6. Si tanto la dirección como el CRC coinciden, la parte de carga útil del paquete, que es todo el paquete excepto los bits de dirección N y los bits CRC, se copiará en los Datos Fuera Registro. Al mismo tiempo, la señal de datos preparados o Data Ready (DR1) se establecerá a nivel alto.

7. el microcontrolador con la señal de datos DR1 conectada a uno de su pines de interrupción de E / S ahora puede despertar de su estado de hibernación, sabiendo que una nueva paquete de datos ha llegado.

8. El micro-controlador ahora puede generar una señal de reloj en el pin de reloj del nRF2401, y la carga útil recibida se irá al registro de datos de salida. Cuando toda la carga útil ha salido, la señal DR1 irá a nivel bajo.

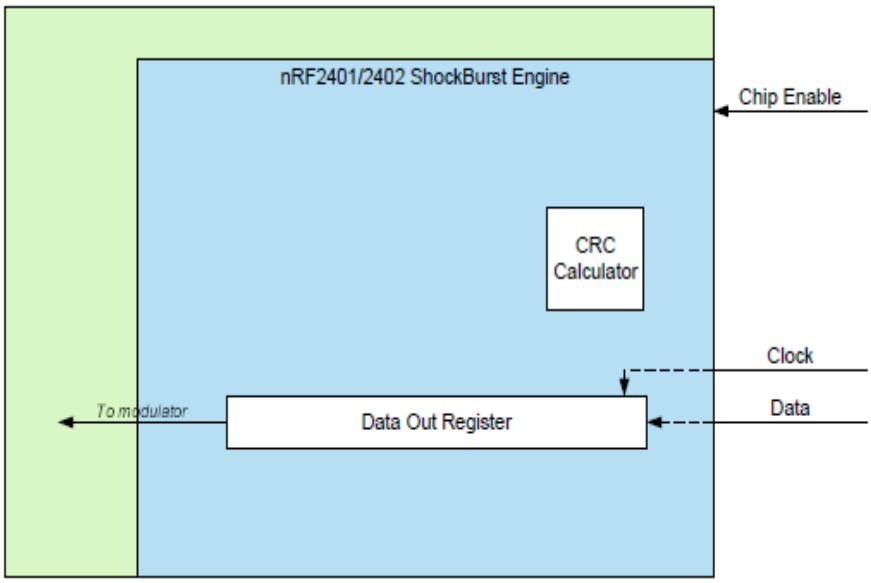


Fig.3.42.*ShockBurst en modo de transmisión*

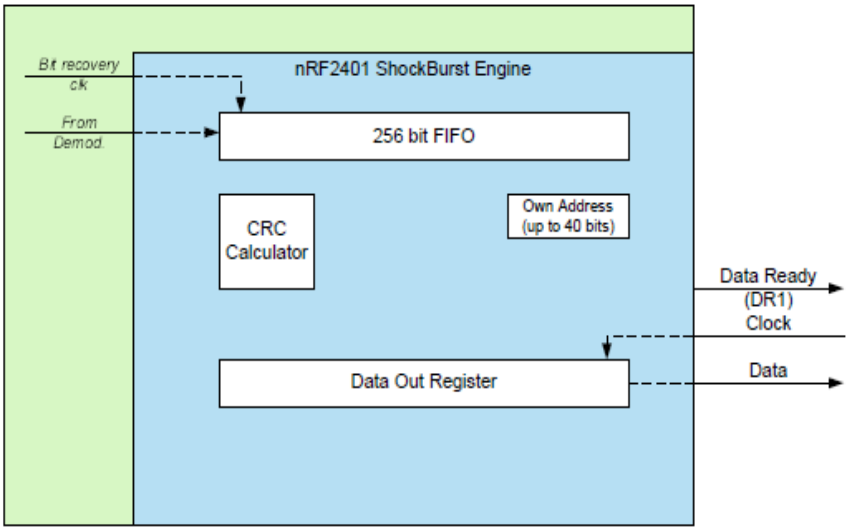


Fig.3.43. *ShockBurst en modo de recepción*

3.3.4. Modulación GFSK

La GFSK (Gaussian Frequency Shift Keying) es un tipo de modulación digital donde un 1 lógico es representado mediante una desviación positiva (incremento) de la frecuencia de la onda portadora, y un 0 mediante una desviación negativa (decremento) de la misma. La GFSK se encuentra en muchos estándares como Bluetooth, DECT y Wavenis.

En la modulación digital, a la relación de cambio a la entrada del modulador se le llama bit-rate y tiene como unidad el bit por segundo (bps).

A la relación de cambio a la salida del modulador se le llama baud-rate. En esencia el baud-rate es la velocidad o cantidad de símbolos por segundo.

La modulación GFSK es una versión mejorada de la modulación por desplazamiento de frecuencia (FSK) donde la señal moduladora solo varía entre dos valores de tensión discretos formando un tren de pulsos donde uno representa un "1" o "marca" y el otro representa el "0" o "espacio", los dos valores binarios se representan con dos frecuencias diferentes (f1 y f2) próximas a la frecuencia de la señal portadora, fp.

Generalmente f1 y f2 corresponden a desplazamientos de igual magnitud pero en sentidos opuestos de la frecuencia de la señal portadora.

$$v(t) = \begin{Bmatrix} V_p \ \text{sen}(2\pi \ f_1 \ t) \text{ para un "1" binario} \\ V_p \ \text{sen}(2\pi \ f_2 \ t) \text{ para un "0" binario} \end{Bmatrix}$$

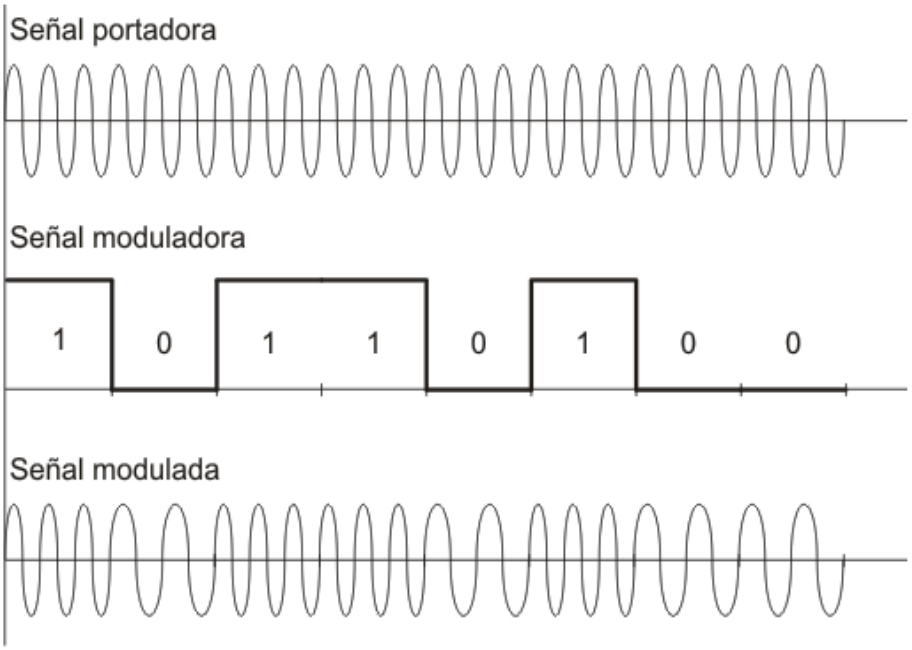
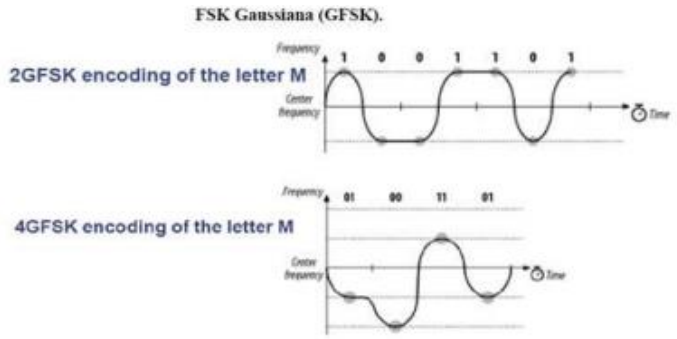


***Fig.3.44**. Señales modulación FSK*

En la GFSK la información es pasada por un filtro gausiano antes de modular la señal. Esto se traduce en un espectro de energía más estrecho de la señal modulada, lo cual permite mayores velocidades de transferencia sobre un mismo canal.

Ahora la modulación 2GFSK es un tipo de modulación donde un 1 lógico es representado mediante una desviación positiva (incremento) de la frecuencia de la onda portadora, y un 0 mediante una desviación negativa (decremento) de la misma proporcionando una velocidad de 1 Mbps.

Para la modulación 4GFSK la misma se da en cuatro niveles, incrementando así la velocidad a 2 Mbps.



Frequency-hopping spread spectrum

Data rate	Modulation	Symbol rate	Bits/symbol
1 Mbps	Two-level GFSK	1 Msps	1
2 Mbps	Four-level GFSK	1 Msps	2

***Fig.3.45.** Tabla comparativa de modulaciones 2GFSK y 4GFSK*

3.3.5. Estructura FIFO

Una cola es una estructura de datos en la que el primer dato en entrar es el primer dato en salir. Es decir, es una estructura FIFO (First In First Out).
La memoria FIFO (First-in-First-Out) ha progresado desde funciones lógicas bastante simples hasta los buffer de alta velocidad que incorporan grandes bloques de SRAM.
El FIFO es un tipo especial de buffer utilizado para almacenar la carga útil que se transmite (TX FIFO) o que se recibe lista para incorporarla al registro de datos de salida.
Los FIFO son accesibles tanto en el modo PTX y el modo de PRX. Los siguientes FIFO están presentes en nRF24L01:

• TX de tres niveles, FIFO 32 bytes
• RX de tres niveles, FIFO 32 bytes

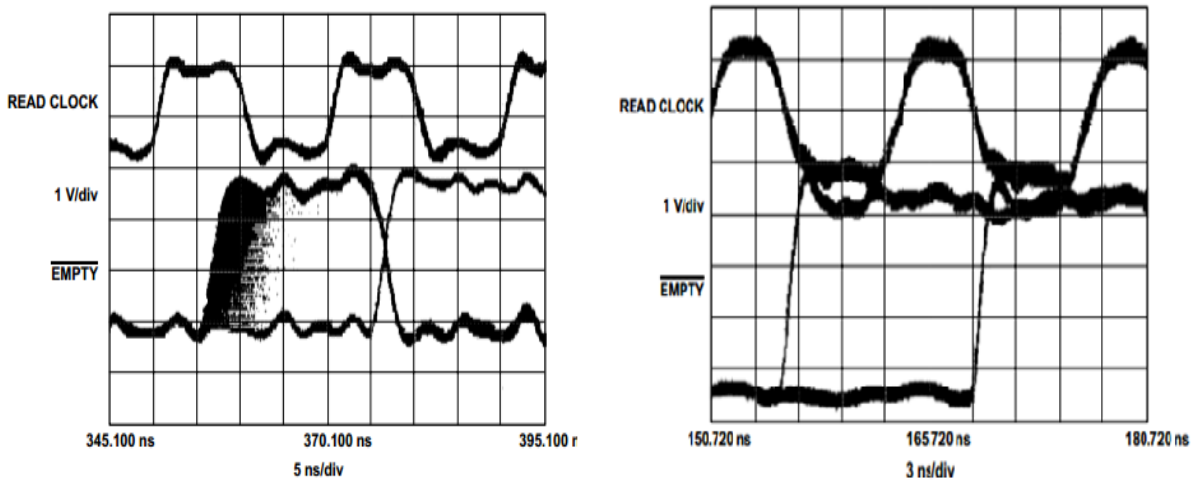


Fig.3.46. *Señales FIFO con un único nivel de sincronización (izda.) y con 3 niveles (dcha.)*

Ambos FIFO tienen un controlador y son accesibles a través de la SPI utilizando comandos dedicados SPI.

Una TX FIFO en PRX puede almacenar la carga útil para los paquetes ACK para tres dispositivos diferentes PTX. Si el TX FIFO contiene más de una carga útil, éstas se manejan usando la FIFO. El TX FIFO en un PRX se bloquea si todas las cargas pendientes están dirigidas a los buses de datos de manera que se pierde el enlace a la PTX. En este caso, la MCU puede vaciar el TX FIFO utilizando el comando FLUSH_TX.

El RX FIFO en PRX puede contener carga útil de hasta tres dispositivos PTX diferentes.
Un FIFO TX en PTX puede tener hasta tres cargas útiles almacenados.
El TX FIFO puede ser escrito por tres comandos, W_TX_PAYLOAD, modo W_TX_PAYLOAD_NO_ACK en PTX y modo W_ACK_PAYLOAD en el modo de PRX.

Los tres comandos dan acceso al registro TX_PLD.
El RX FIFO puede ser leído por el comando R_RX_PAYLOAD tanto en modo PTX y PRX. Este comando da acceso al registro RX_PLD.
La carga útil en TX FIFO en un PTX no se elimina si el MAX_RT IRQ se habilita.

Hay varias formas de implementar una cola en la memoria de un computador. Una forma simple consiste en almacenar los datos en posiciones de memoria adyacentes y utilizar punteros para el principio y el fin de la cola.

Cuando un elemento se añade a la cola, el puntero de la parte posterior se ajusta para que señale al nuevo elemento. De manera similar, cuando un elemento se elimina de la cola, se ajusta el puntero delantero para que señale al nuevo primer elemento.

El problema de este método para implementar las colas es que las posiciones de memoria que ocupan, varían a medida que se añaden y eliminan elementos de la misma. La solución consiste en asignar un área fija para almacenar la cola y permitir que se mueva en esta área de manera circular. Un área de almacenamiento de esta forma se denomina buffer circular.

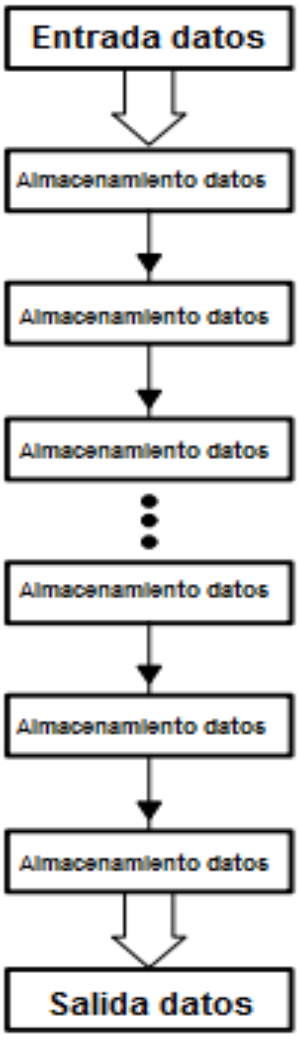


Fig. 3.47. *Flujo de datos FIFO unidireccional*

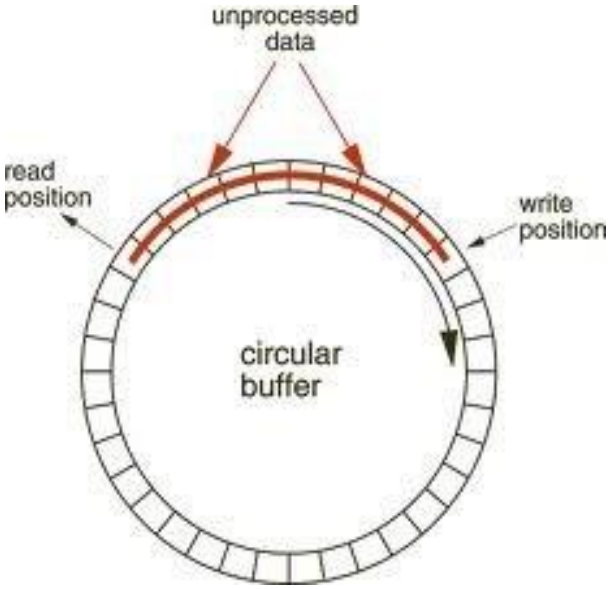


Fig.3.48. *Flujo de datos FIFO circular*

CAPITULO 4

IMPLEMENTACION DE LA ARQUITECTURA HARDWARE

El término hardware hace referencia a cualquier componente físico tecnológico, que trabaja o interactúa de algún modo en un sistema, como interfaz física entre el operario, que envía las señales de ejecución, y el programa, que ejecuta estas instrucciones deseadas.

En este capítulo nos meteremos de lleno en la arquitectura hardware del diseño del proyecto, esto es, hablaremos de los diferentes elementos que lo componen, la forma de funcionamiento de cada uno de ellos, la disposición de los mismos y su manipulación hasta conseguir el resultado final deseado.

Al igual que en resto de capítulos, diferenciaremos las diferentes fases y etapas por las que pasa el montaje y diseño del proyecto así como las características de cada uno de sus componentes al detalle tanto del coche RC receptor como del mando de control emisor.

Comenzaremos pues con una breve introducción de los materiales utilizados en la construcción del prototipo de coche a controlar así como las ventajas que lo caracterizan frente al resto de prototipos existentes en el mercado.

4.1. KIT ROBOT-CAR ARDUINO 2WD

En este capítulo, se hablará del prototipo donde estarán integrados todos los materiales; el kit robot-car para Arduino, el módulo receptor. Aclarar que aquí, se hablará sobre el coche y su montaje pero, se ha de tener en cuenta que el módulo receptor se compone, aparte del coche, que hará las veces de soporte, de un Arduino Uno, un integrado inversor de giro L293D, un transceptor de radiofrecuencia nRF24L01 y las alimentaciones necesarias que, en este caso, serán una pila de 9V para las entradas Enable del integrado inversor de giro anteriormente citado y cuatro pilas AAA de 1´5V para la alimentación de 6V del Arduino (lo alimentaremos a la tensión mínima).

Este kit dotará de comodidad de manera que todos los componentes colocados sobre y en la placa estén en una disposición que optimice el espacio útil de la misma y la manipulación de dichos elementos.

El kit viene compuesto de varios elementos, esto dará juego a la hora de usarlos para diferentes fines; el kit se compone de un par de ruedas de dirección principales (2WD), una rueda loca, la tornillería, la placa de acrílico, el cajetín para las baterías de 1,5V, dos motores de continua y los cables de conexión de los motores a las ruedas. Además vienen incorporados dos encoders incrementales de velocidad que se podrán colocar junto a cada una de las ruedas de dirección para un control de velocidad si se desea.

En este proyecto, estos enconders no serán incluidos ya que nuestro objetivo será la programación y el control mediante los módulos de radiofrecuencia nRF24L01 de los que ya hemos hablado detalladamente con en los capítulos anteriores.

4.1.1. Arquitectura de los componentes

Como ya hemos dicho anteriormente, el kit viene compuestos por diferentes elementos, algunos imprescindibles, otros no, dependiendo de los objetivos del proyecto.
En este apartado enumeraremos y estudiaremos todos y cada uno de ellos con sus características físicas, materiales y sus utilidades.

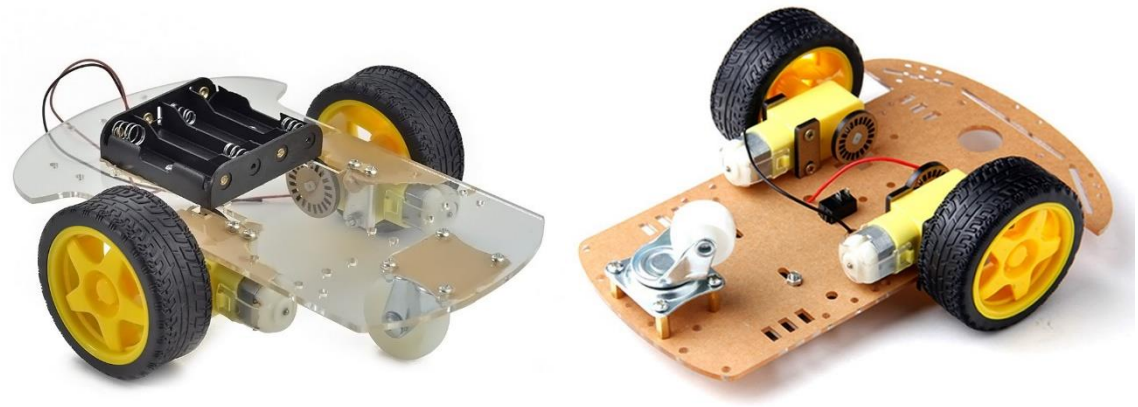


Fig.4.1. *Vistas superior (izda.) e inferior (dcha.) del robot car Arduino 2WD*

El kit se compone de los siguientes elementos y características:

- Modelo: 081827
- Color: Negro + Amarillo
- Material: metacrilato
- 1 x chasis del automóvil inteligente
- 2 x llantas de coches
- 1 x rueda loca
- 1 x compartimento de las pilas 4 AA
- 2 x encoders de velocidad
- 2 x Motorreductores o motores DC 3 ~ 12V (120RPM con tensión de 3V)
- 4 x tornillos cortos cabeza plana
- 4 x Tornillos largos
- 8 x tornillos cortos
- 8 x tuercaa
- 1 x 3D plano de montaje
- Dimensiones: 21,0 cm x 15,0 cm x 0,66 cm
- Peso: 272 g

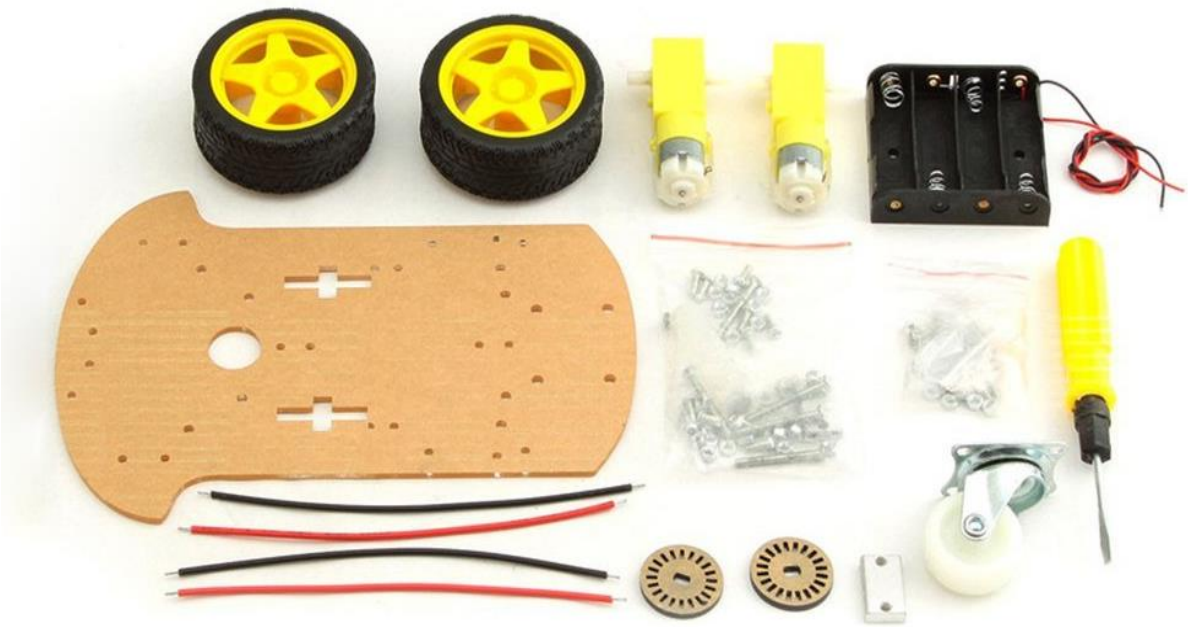


Fig.4.2. *Elementos del kit car Arduino 2WD*

- **PLACA DE METACRILATO**

La placa de metacrilato o acrílico translúcido viene con papel adhesivo protector. Dentro de los plásticos de ingeniería el metacrilato podemos encontrarlo como polimetilmetacrilato, también conocido por sus siglas PMMA.

La placa de acrílico se obtiene de la polimerización del metacrilato de metilo y la presentación más frecuente que se encuentra en la industria del plástico es en gránulos o en placas. Los gránulos son para el proceso de inyección o extrusión y las placas para termoformado o para mecanizado.

Compite en cuanto a aplicaciones con otros plásticos como el policarbonato (PC) o el poliestireno (PS), pero el acrílico se destaca frente a otros plásticos transparentes en cuanto a resistencia a la intemperie, transparencia y resistencia al rayado.

Por estas cualidades es utilizado en la industria del automóvil como el faro del coche, iluminación, ect. Las ventajas de este material son muchas pero las que lo diferencian del vidrio son: bajo peso, mejor transparencia, menor fragilidad.

De los demás plásticos se diferencia especialmente por su mejor transparencia, su fácil moldeo y su posible reparación en caso de cualquier raya superficial. La posibilidad de obtener fibras continuas de gran longitud mediante un proceso de fabricación relativamente barato hacen junto con su elevada transparencia que sea un material muy empleado para la fabricación de fibra óptica.

El PMMA no es tóxico si está totalmente polimerizado. Su componente el MMA (monómero de metacrilato de metilo) sí lo es en fase líquida.

Entre sus propiedades se destacan:

- Transparencia de alrededor del 93 %. El más transparente de los plásticos.
- Alta resistencia al impacto, de unas diez a veinte veces la del vidrio.
- Resistente a la intemperie y a los rayos ultravioleta. No hay un envejecimiento apreciable en diez años de exposición exterior.
- Excelente aislante térmico y acústico.
- Ligero en comparación con el vidrio (aproximadamente la mitad), con una densidad de unos 1190 kg/m3 es sólo un poco más denso que el agua.
- De dureza similar a la del aluminio: se raya fácilmente con cualquier objeto metálico, como un clip. El metacrilato se repara muy fácilmente con una pasta de pulir.
- De fácil combustión, no es autoextinguible. No produce ningún gas tóxico al arder por lo que se puede considerar un producto muy seguro para elementos próximos a las personas al igual que la madera.
- Gran facilidad de mecanización y moldeo.
- Se protege su superficie con una película de polietileno para evitar que se raye al manipularlo.
- Se puede mecanizar en frío pero no doblar (serrado, esmerilado, acuchillado, pulido, etc.). Para doblarlo hay que aplicar calor local o calentar toda la pieza.
- Presenta gran resistencia al ataque de muchos compuestos pero es atacado por otros, entre ellos: Acetato de etilo, acetona, ácido acético, ácido sulfúrico, alcohol amílico, benzol, butanol, diclorometano, triclorometano (cloroformo), tolueno.

La placa del kit viene recortada con unas medidas específicas de manera que todos los componentes a montar quedan perfectamente colocados en ella.

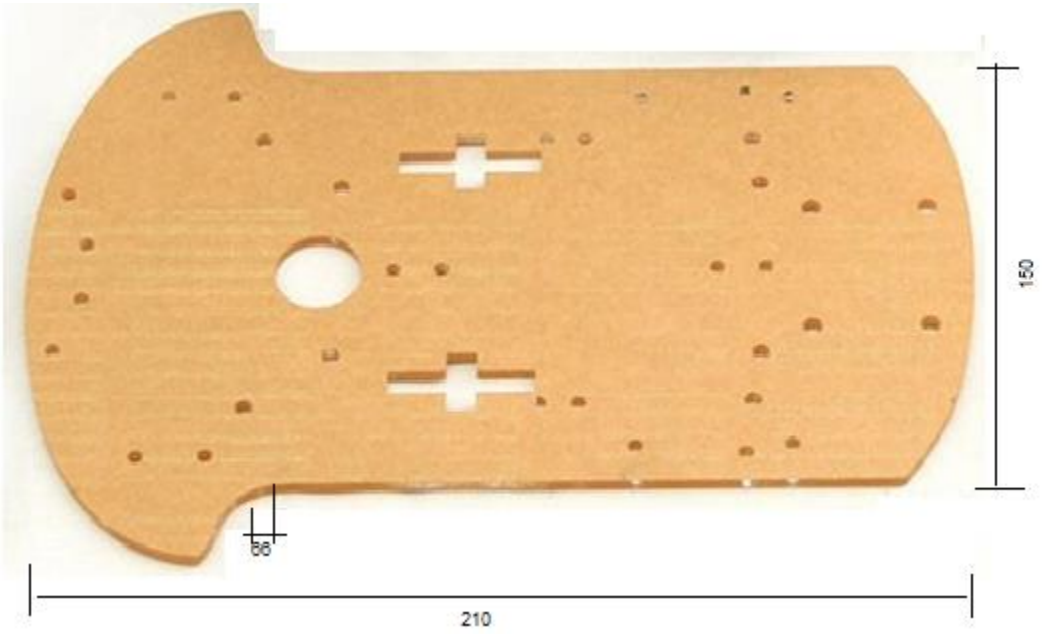


Fig.4.3.*Medidas placa metacrilato*

- **RUEDAS DE DIRECCÍON**

Las ruedas del coche tienen bastantes mejoras respecto a modelos anteriores; tiene una llanta forrada de esponja acrílica de manera que hacen que la fricción respecto al suelo se reduzca, así como la introducción de una mejor banda de rodadura interna.

En este proyecto, el coche contara de dos ruedas de dirección, 2WD, de manera q la tercera rueda, la rueda loca, estará comandada por la dirección de los motores.

Las medidas de la rueda son; radio 32mm y ancho 22mm.

El dibujo de la cubierta ayudará al vehículo a no deslizarse en superficies lisas o mojadas, ayudando a la estabilidad del mismo en el movimiento.

Fig.4.4. *Medidas de rueda de dirección*

- **RUEDA LOCA**

También conocida como rueda de apoyo, la tendremos en el extremo de la plancha del coche de manera que, alineada a la altura del radio de las dos ruedas motrices, dotarán al vehículo de estabilidad en tres puntos.

La rueda giratoria hace que vehículos de cualquier tamaño y peso sean fáciles de dirigir. Permite cambios de dirección y asegura la maniobrabilidad incluso en espacios reducidos. Esta movilidad hace inadecuada para viajar largas distancias en línea recta. Por este motivo, a menudo se utiliza en combinación con ruedas fijas.

Además de la movilidad, la capacidad de permanecer firmemente en su lugar es extremadamente importante para muchos dispositivos.

Las ruedas giratorias pueden girarse verticalmente y proporcionan maniobrabilidad a máquinas y aparatos. Una horquilla (horquilla giratoria) va acoplada al elemento de fijación a través de un cojinete (cabeza giratoria). El elemento de fijación se monta firmemente en el aparato. La horquilla conserva su capacidad de giro.

Para que la horquilla pueda girarse fácilmente., la rueda normalmente se monta con una distancia horizontal entre los ejes del cojinete giratorio y la rueda.

Esta distancia se denomina voladizo y siempre y cuando haya sido concebida correctamente, permite una fácil rotación de la rueda, sin accesorios adicionales, proporcionándole a la misma un movimiento estable con desplazamiento recto.

Como elementos de fijación son muy efectivas las platinas, las espigas de acero para tubos, así como la fijación por tornillo a través de un agujero pasante en el soporte giratorio.

Las medidas de la rueda son; radio de 10mm, ancho de 12mm y pletina de 30mm de largo y 25 de ancho de centro a centro.

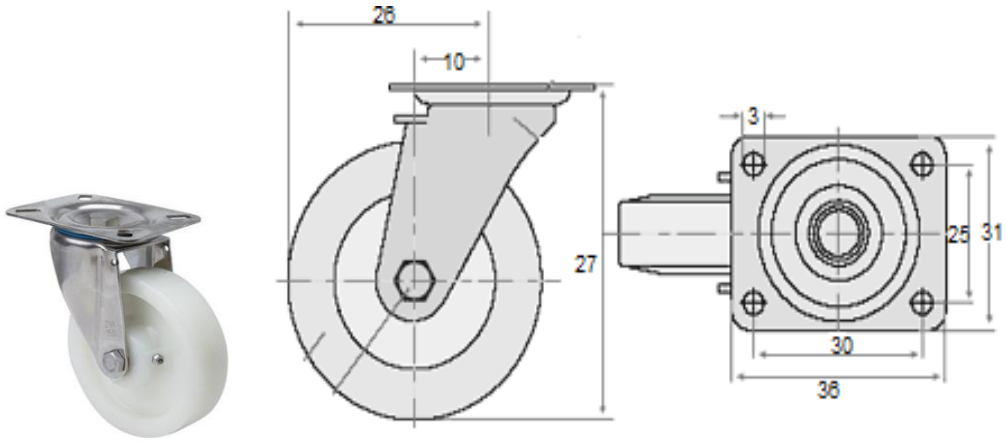


***Fig.4.5**. Rueda giratoria, medidas en vistas de alzado (izda.) y planta superior (dcha.)*

- **MOTOR DC**

En el vehículo, los dos motores de continua que utilizaremos serán uniaxiales conectados a cada rueda de dirección respectivamente.

Todos los robots incluyen algún sistema capaz de producir movimiento siendo los más corrientes los motores de corriente continua (DC) y los servos motores o servos. Los primeros se utilizan casi siempre junto con un sistema de engranajes que reducen la velocidad y proporcionan mayor fuerza.

El segundo sistema y quizás el más extendido en robótica, consiste en la utilización de servos de radiocontrol que tienen la gran ventaja de ser económicos y fáciles de usar en cualquier tipo de robot. Está compuesto de un estator y un rotor. En muchos motores c.c., generalmente los más pequeños, el estator está compuesto de imanes para crear un campo magnético. Para controlar el sentido del flujo de la corriente en los conductores se usa un conmutador que realiza la inversión del sentido de la corriente cuando el conductor pasa por la línea muerta del campo magnético.

La fuerza con la que el motor gira (el par motor) es proporcional a la corriente que hay por los conductores. A mayor tensión, mayor corriente y mayor par motor.

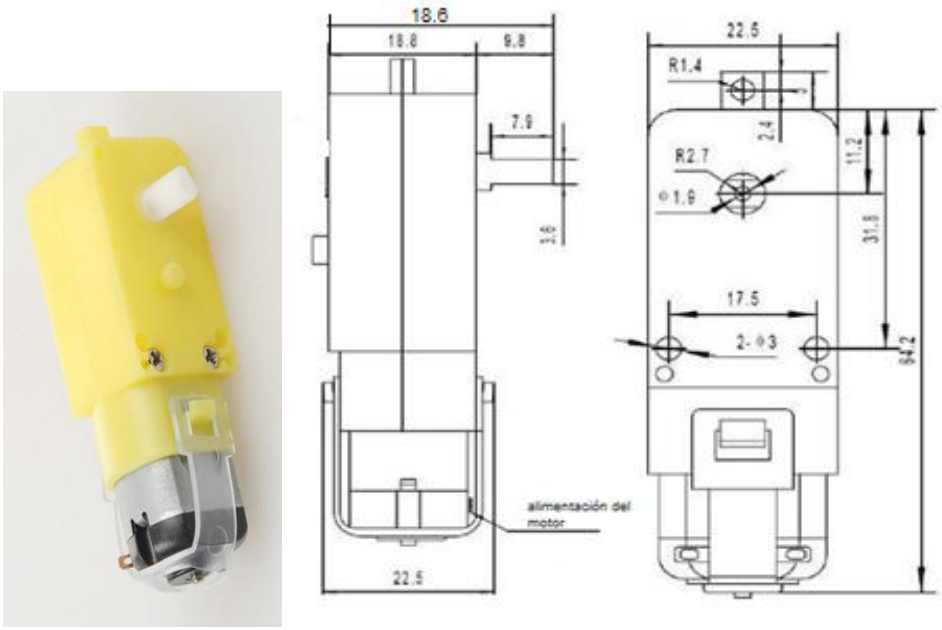


Fig.4.6. *Motor DC Arduino uniaxial; medidas acotadas planta (izda.) y perfil (dcha.)*

- **ZÓCALO PILAS AA**

Para la alimentación de los motores de DC que irán conectados a las ruedas de dirección dispondremos de las baterías de 1, 5V, cuatro pilas en concreto, 6V, para ello necesitaremos tenerlas conectadas y encapsuladas en el zócalo para 4 baterías AA con terminales en clip, de base plana con terminales de 147mm, puntas estañadas y cuadrangular, ideal para cierto tipo de plataformas como por ejemplo, para nuestro diseño ya que no ocupa mucho espacio sobre la plancha.

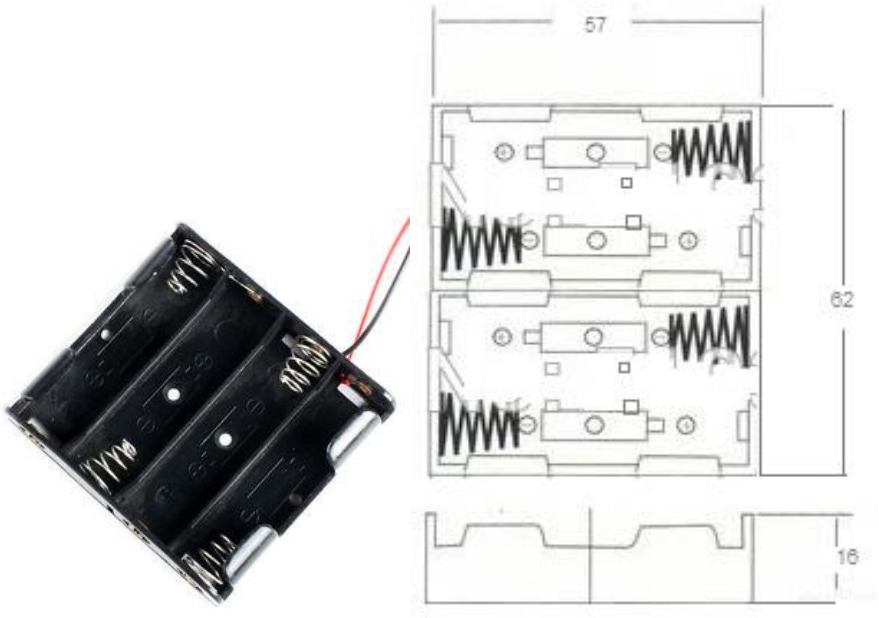


Fig.4.7. *Zócalo para 4 baterías AA; medidas planta (izda.) y alzado (dcha.)*

- **TORNILLERÍA**

En cuanto a la sujeción en el montaje, el kit nos viene con la tornillería precisa para que el vehículo esté correctamente montado. Éste consta de 8 tuercas hexagonales de 2,5mm de radio y ancho de 1,5mm empleadas para el ajuste de los tornillos de las ruedas de dirección y la rueda loca, 4 tornillos largos de 30mm para las ruedas de dirección, 8 tornillos cortos de 10mm de largo para el ajuste superior a la placa de la rueda loca y el ajuste de los motores lateralmente y 4 tornillos cortos de 6 mm le largo para el ajuste inferior de la rueda loca a las tuercas largas de acoplamiento.

Fig.4.8. *Tornillería utilizada*

- **ENCODERS VELOCIDAD**

El encoder que nos trae el kit que, como ya se ha comentado, no va a ser utilizado, es de tipo incremental; Los encoders incrementales generan un número exactamente definido de impulsos por revolución. Éstos indican la medida de la distancia angular y lineal recorrida. Debido al desplazamiento de fase entre las señales A y B (de aproximadamente 90 grados), se puede analizar el sentido de rotación.

Se utilizan dos discos unido al eje, pero este disco es mucho más pequeño marcado con una gran cantidad de líneas de la parte radial como los radios de una rueda.

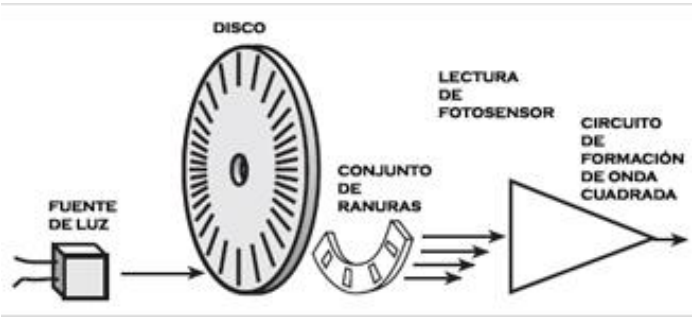


Fig.4.9*. Encoder incremental, funcionamiento*

4.1.2. Construcción del prototipo

Empezaremos por retirar el papel protector autoadhesivo de la placa de metacrilato, procediendo entonces a la colocación de los motores. Para ello, existe una pieza de aleación de aluminio que sirve de unión entre la placa el motor, de manera que se colocará vertical a la placa quedando perpendicular al eje de la rueda conectada al motor.

El agarre de la pieza de aluminio a la placa se hará mediante dos tornillos cortos de 10mm, a su vez, el motor de unirá a la pieza de aluminio mediante dos tornillos largos de 30mm que cruzan el cajetín del motor y se ajustan mediante dos tuercas posteriormente.

Las ruedas se colocarán sin problema en el eje que sobresale del motor para dicho propósito. La colocación de la rueda loca irá partiendo de una plataforma que le ayudará a conseguir las medidas del radio de las ruedas motoras de manera que la distancia de la placa al suelo sea la misma que dicho radio para q no haya descompensación y quede equilibrado el vehículo en los tres puntos; las cuatro tuercas largas de acoplamiento serán encargadas de dar esas medidas de altura.

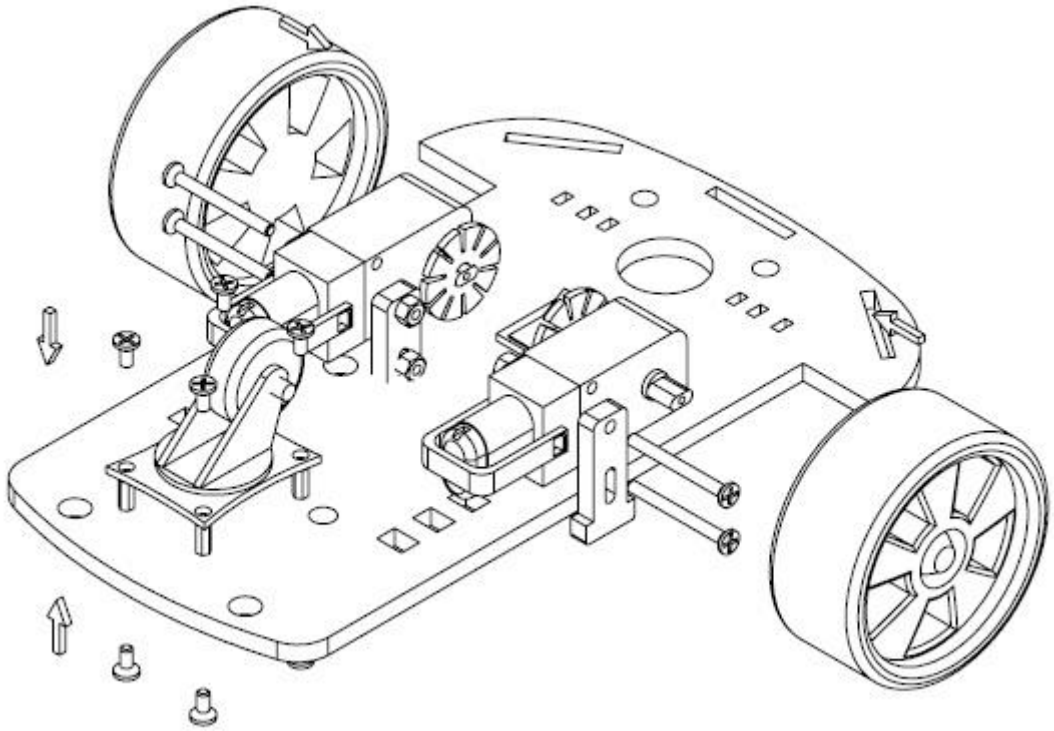


***Fig.4.10**. Detalle del anclaje y colocación de los diferentes elementos*

4.1.3. Alimentación

Un motor DC puede funcionar libremente en ambas direcciones, es muy fácil controlar su velocidad pero no su posición. Tampoco es sencillo hacerlo parar de forma precisa. Viene con dos cables: alimentación y tierra donde serán conectados al cajetín de baterías AA a una tensión de 6V.

Además. Se ha colocado un pequeño interruptor conectado a la pila que alimenta las entradas Enable de integrado para que, estando todas las pilas colocadas, el coche pueda estar apagado sin necesidad de quitar las mismas

En un motor CC, la velocidad y la corriente que necesita el motor dependen de la carga que tenga aplicada.

En este tipo de motor parte de la tensión aplicada se pierde en la resistencia interna (resistencia del devanado de excitación). El resto de la tensión se utiliza para hacer girar el motor.

Cuando la carga de un motor cc aumenta, también aumenta la corriente que consume este. Esta corriente causa una caída de tensión mayor en la resistencia interna del motor (resistencia del devanado excitación).

Como la alimentación del motor CC permanece constante, la tensión aplicada para hacer girar el motor es menor y en consecuencia la velocidad de giro del motor es menor.

$$Vb = Vm - Ia \times Ra$$

Vb: tensión real utilizada hacer girar el motor.
Vm: Tensión aplicada a todo el conjunto motor.
Ra: Resistencia del devanado de excitación (resistencia interna).
Ia: corriente que circula por el motor.
Ia x Ra: es la tensión que se pierde en la resistencia interna del motor CC. Ver que depende directamente de Ia (corriente de alimentación del motor).

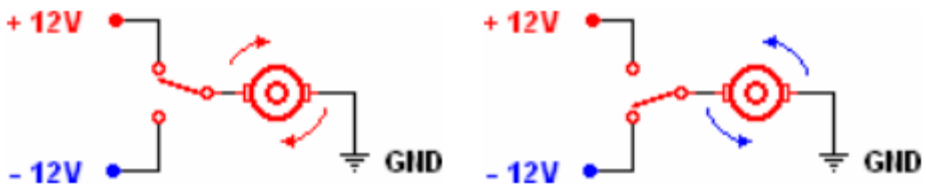
Si la corriente Ia aumenta, Vb disminuye y como la velocidad de giro del motor es proporcional a Vb. Si Vb disminuye entonces la velocidad del motor también.

- Control de Sentido de Giro para Motores-CC:

Existen varias formas de lograr que estos motores inviertan su sentido de giro una es utilizando una fuente simétrica o dos fuentes de alimentación con un interruptor simple de dos contactos y otra es utilizar una fuente común con un interruptor doble es decir uno de 4 contactos, en todos los casos es bueno conectar también un capacitor en paralelo entre los bornes del motor, éste para amortiguar la inducción que generan las bobinas internas del motor.
Si bien, esto no será el sistema utilizado, sino, como se ha explicado anteriormente, el control de motores y si inversión de giro lo ejecutará el integrado L293D conectado a los mismos y al Arduino.

Las conexiones serían así:

Con Fuente Simétrica o Doble Fuente



Con una Fuente Simple

4.2. MANDO DE CONTROL

En este apartado del capítulo cuatro se hablará del mando de control o, lo que es lo mismo, del módulo emisor.
Veremos de qué elementos se compone el mismo, sus características y su montaje en el encapsulado correspondiente.

También veremos los materiales utilizados así como su modo de empleo sobre los elementos del coche, la alimentación requerida y la disposición adecuada para la optimización del espacio según los recursos utilizados.

4.2.1. Arquitectura de los componentes

El mando de control constará de varios elementos indispensables como son, la caja contenedora del material, el Arduino Uno y el transceptor nRF24L01 dentro, y en la parte superior colocaremos el joystick único al Arduino por el cableado.

Se concretará de manera particular sobre la caja contenedora del módulo y su joystick de control ya que, del módulo nRF24L01 ya se ha hecho un apartado completo y será reiterada la información.

- **CAJA CONTENEDORA ABS**

La caja donde incluiremos nuestros materiales para el mando será una caja de medidas acordes a las medidas de los demás de manera que puedan caber dentro.
Esta caja deberá ser resistente a los golpes, a la temperatura, aislante e impermeable, características que nos dará el tipo de plástico llamado ABS o acrilonitrilo butadieno estireno.

El ABS es un plástico muy resistente al impacto (golpes) muy utilizado en automoción y otros usos tanto industriales como domésticos. Es un termoplástico amorfo.
Se le llama plástico de ingeniería, debido a que es un plástico cuya elaboración y procesamiento es más complejo que los plásticos comunes, como son las polioleofinas (polietileno, polipropileno).
Por estar constituido por tres monómeros diferentes se lo denomina terpolímero (copolímero compuesto de tres bloques).

La resistencia al impacto de los plásticos ABS se ve incrementada al aumentar el porcentaje de contenido en butadieno pero disminuyen entonces las propiedades de resistencia a la tensión y disminuye la temperatura de deformación por calor.

El amplio rango de propiedades que exhibe el ABS es debido a las propiedades que presentan cada uno de sus componentes.

El acrilonitrilo proporciona:

- Resistencia térmica
- Resistencia química
- Resistencia a la fatiga
- Dureza y rigidez

El butadieno proporciona:

- Ductilidad a baja temperatura
- Resistencia al impacto
- Resistencia a la fusión

El estireno proporciona:

- Facilidad de procesado (fluidez)
- Brillo
- Dureza y rigidez

Excepto en películas delgadas, es opaco y puede ser de color oscuro o marfil y se puede pigmentar en la mayoría de los colores, obteniéndose partes lustrosas de acabado fino.
La mayoría de los plásticos ABS son no tóxicos e incoloros.
Pueden ser extruidos, moldeados por inyección, soplado y prensado. Generalmente los grados de bajo impacto son los que más fácil se procesan. Los de alto impacto son más dificultosos porque al tener un mayor contenido en caucho los hace más viscosos.

A pesar de que no son altamente inflamables, mantienen la combustión. Hay algunos tipos autoextinguibles para cuando se requiere algún producto incombustible, otra solución consiste en aplicar algún retardante de llama.
Dentro de una variedad de termoplásticos el ABS es importante por sus balanceadas propiedades.

El ABS se destaca por combinar dos propiedades muy importantes como ser la resistencia a la tensión y la resistencia al impacto en un mismo material, además de ser un material liviano.

***Fig.4.11**. Caja contenedora ABS*

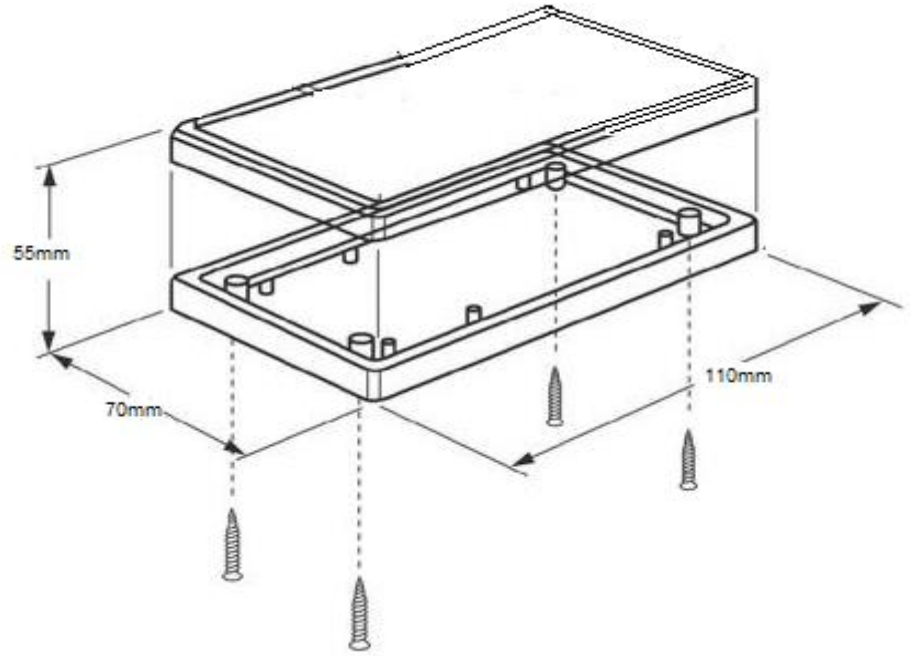


***Fig.4.12**. Medidas acotadas de la caja contenedora*

- **JOYSTICK**

El joystick del mando de control es la interfaz física entre el manipulador y el coche radiocontrolado de manera que el movimiento del mismo será regido por el joystick.
El joystick consta del cabezal ergonómico de rotación que estará colocado sobre el funcionamiento del aparato que se basará en dos potenciómetros que harán las veces de eje X y eje Y respectivamente. Las salidas del joystick serán cinco; las dos de alimentación, GND Y +5V, que irán conectadas al pin GND y al 5V del Arduino, también tenemos las salidas VRx y VRy que irán a los pines de entradas analógicas del Arduino, A0 y A1 respectivamente.

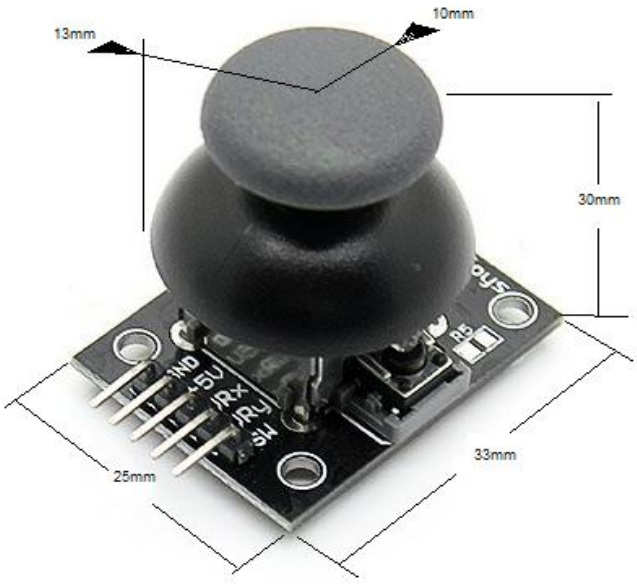


***Fig.4.13.**. Medidas acotadas del thumbstick o joystick*

4.2.2. Construcción del prototipo

Para la construcción del módulo emisor de manera física se comenzará con la adaptación de la caja ABS para la incorporación de Arduino Uno y el transceptor Nrf24L01 dentro de ella.
Se hará un taladro en la cara posterior a la cara de la tornillería de manera que, partiendo de una esquina, se corte la mayoría de la cara con un arco de sierra de marquetería con pluma de espiral dejando un borde de 7mm que hará de soporte o encuadre de la lámina de plástico traslúcido sobre la que se colocará el joystick a la que también se le hará una abertura para introducir el cableado que irá del joystick al Arduino.

Se ha escogido una cara de plástico traslúcido para mostrar la disposición del montaje y circuitería del módulo emisor en su totalidad al igual que lo hace el módulo receptor, ya que el coche carece de carrocería.

Como se ha comentado también en el capítulo 5 para la programación del módulo emisor, el prototipo del mismo será el Arduino conectado al ordenador y el nRF24L01 como transceptor emisor ya que, en el último momento ocurrió un fallo técnico con el joystick físico que, por ausencia de tiempo no pudo ser solventado acudiendo pues a otra solución igual de válida para la optimización del tiempo.

4.2.3. Alimentación

La alimentación del módulo emisor se reducirá, básicamente, a la alimentación del Arduino, al cual serán conectados tanto el joystick como el transceptor de radio.
Como se ha visto en el capítulo anterior, la alimentación del Arduino tiene un rango de operación que va de 6 a 20V, siendo el intervalo óptimo el que va de 7 a 12V.

Dicho esto, y pudiendo alimentar al Arduino con cuatro pilas AAA de 1,5V como ocurre en el módulo receptor, se optará por la alimentación del mismo con una pila de 9V.

CAPITULO 5

IMPLEMENTACIÓN DE LA ARQUITECTURA SOFTWARE

En este capítulo se realiza un análisis del software de control del que se ha dotado al microcontrolador que incorpora la plataforma Arduino.

En este análisis se explica el software del módulo emisor y el módulo receptor por separado, comentando sus principales similitudes y diferencias.

En primer lugar presentaremos el entorno de programación en el que trabajaremos, así como el lenguaje de programación usado y unos ejemplos que nos ayudarán a entender la mecánica de la interfaz de Arduino.
Por último nos meteremos de lleno en el desarrollo del programa que hará que funcione nuestro proyecto radiocontrolado que veremos bien diferenciado en dos bloques, uno para cada módulo; emisor y receptor.

5.1. ENTORNO DE PROGRAMACIÓN

El entorno de desarrollo Arduino contiene un editor de texto para escribir código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús. Se conecta al hardware Arduino para cargar programas y comunicarse con ellos.

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (boot loader) que corre en la placa.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.
Al ser open-hardware, tanto su diseño como su distribución son libres. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.
Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores.

El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing).

Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing,MaxMSP).La plataforma Arduino se programa mediante el uso de un lenguaje propio basado en el popular lenguaje de programación de alto nivel Processing. Sin embargo, es posible utilizar otros lenguajes de programación y aplicaciones populares en Arduino. Algunos ejemplos son Java, Flash (mediante ActionScript), Processing, Pure Data, C, Etc.

Esto es posible debido a que Arduino se comunica mediante la transmisión de datos en formato serie que es algo que la mayoría de los lenguajes anteriormente citados soportan.

Es bastante interesante tener la posibilidad de interactuar Arduino mediante esta gran variedad de sistemas y lenguajes puesto que dependiendo de cuales sean las necesidades del problema que vamos a resolver podremos aprovecharnos de la gran compatibilidad de comunicación que ofrece.

Processing está basado en Java, y está diseñado para ser usado por personas sin conocimientos previos técnicos, ya que existe una interfaz gráfica que simplifica las cosas. El lenguaje nativo de Arduino, C++, no deja de ser una extensión del lenguaje estructurado C para permitir el manejo de objetos.

El software escrito utilizando Arduino se llama sketch. Éstos se escriben en el editor de texto y se guardan con la extensión de archivo .ino. Tiene características para cortar / pegar y para buscar / reemplazar texto. El área de mensajes proporciona información de manera que la verificación y la carga mostrarán los errores.

- La **pantalla** muestra la salida de texto Arduino incluyendo mensajes de error completo y otra información. La esquina inferior derecha de la ventana muestra la actual conexión al puerto serie del arduino. Los botones de la barra de herramientas le permiten comprobar y cargar programas, crear, abrir y guardar sketches, y abrir el monitor serie.

- Las **bibliotecas** proporcionan funcionalidad adicional para uso en sketches, por ejemplo, trabajar con el hardware o la manipulación de los datos. Para utilizar una biblioteca en un proyecto selecciinaremos en el Sketch> Importar biblioteca. Esto insertará una o más declaraciones #include en la parte superior del boceto y compilará la biblioteca con programa. Debido a que las bibliotecas se cargan en el programa des sketch a compilar, aumentan la cantidad de espacio que ocupa. Si un sketch ya no necesita una biblioteca, simplemente eliminaremos sus declaraciones #include de la parte superior de su código.

  Hay una lista de las bibliotecas en la referencia. Algunas bibliotecas se incluyen con el software de Arduino. Otros se pueden descargar de una variedad de fuentes. Desde la versión 1.0.5 del IDE, se puede importar una biblioteca de un archivo zip y utilizarlo en un sketch abierto.

- El **monitor serial** muestra los datos en serie que se envían desde la placa Arduino (USB o puerto serie). Para enviar datos a la placa, introduciremos el texto y haremos clic en el botón "enviar" o presionando enter. Seleccionaremos la velocidad de transmisión en baudios de la lista desplegable que debe coincidir con la que trabajael sketch puesta en la declaración Serial.begin.

- Para la **carga**, antes de subir tu boceto, es necesario seleccionar los elementos correctos yendo a Herramientas> Pantalla y Herramientas> puerto serie. En Windows, es probable que la conexión sea COM1 o COM2 (para una placa de serie) o COM4, COM5, COM7 o superior (para una placa USB) aunque para asegurarnos buscaremos el dispositivo serie USB en la sección de puertos del Administrador de dispositivos de Windows.

  Una vez que se haya seleccionado el puerto serie correcto pulsaremos el botón de carga en la barra de herramientas.

Las placas Arduino actuales se restablecerán automáticamente y comenzarán la carga. Con placas más antiguas (pre-Diecimila) que carecen de auto-reset, tendremos que pulsar el botón de reinicio de la placa justo antes de comenzar la carga. En la mayoría de las placas, veremos que el RX y TX LED parpadean cuando el sketch se carga. El entorno Arduino mostrará un mensaje cuando la carga esté completa, o un error en su defecto.

Al cargar, el Arduino está utilizando un gestor de arranque, un pequeño programa que se ha cargado en el microcontrolador de su placa el cual permite cargar código sin utilizar ningún hardware adicional.

El gestor de arranque está activo durante unos segundos cuando la placa se restablece; luego comenzará la ejecución del sketch subido más recientemente al microcontrolador. El gestor de arranque hará parpadear el LED del pin 13 cuando se inicia.

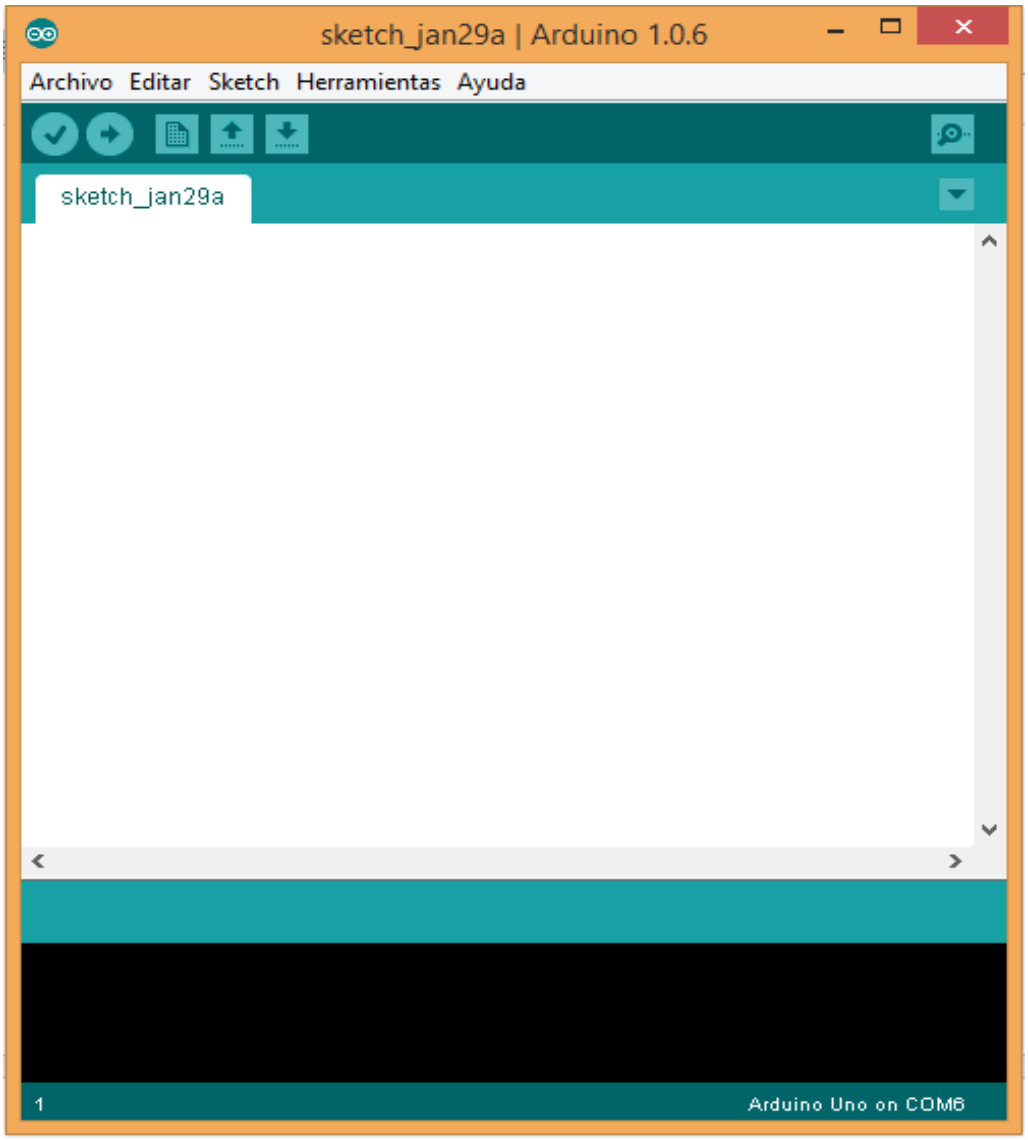


Fig.5.1*.Vista del entorno de programación, sketch, Arduino*

5.1.1. Interfaz de usuario

La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o un programa y comprende todos los puntos de contacto entre el usuario y el equipo.
El principal objetivo de una interfaz de usuario es que éste pueda comunicar información a través de ella hacia algún tipo de dispositivo o sistema. Conseguida esta comunicación, el siguiente objetivo es el que dicha comunicación se desarrolle de la forma más fácil y cómoda posible para las características del usuario que utiliza el servicio.

El programa Arduino brinda al usuario de un entorno de programación de código abierto y una organización de opciones colocadas en pestañas de manera que la ejecución del código se haga de manera sencilla. Al abrir la pantalla del sketch de Arduino, nos aparecerá, a primera vista, una barra de herramientas, éstas serán las acciones básicas con las que daremos forma al programa deseado, así, una vez escrito el código, lo verificaremos, lo cargaremos, veremos cómo interactúa en el puerto serial y lo guardaremos, pudiendo, más tarde, abrirlo de nuevo.

 ***Verificar***

Comprueba que en el código a cargar no existan errores.

 ***Cargar***

Compila el código y lo carga en la placa Arduino E/S

 ***Nuevo***

Crea un nuevo sketch.

 ***Abrir***

Presenta un menú con todos los sketches en la carpeta. Al hacer clic en uno lo abrirá en la ventana actual.

 ***Guardar***

Guarda el boceto.

 ***Serial Monitor***

Abre el monitor serie.

A continuación, vamos a ahondar en la interfaz de usuario viendo los diferentes menús y sus opciones a desplegarlos;

- **Archivo**

En el menú archivo, tendremos también la opción de nuevo, abrir y guardar que ya teníamos en la interfaz directa de la barra de herramientas, además, podremos desplegar la pestaña de ejemplos; estos ejemplos nos acortarán mucho trabajo al programar viendo la dinámica de trabajo de Arduino a través de la carga de programas sencillos.

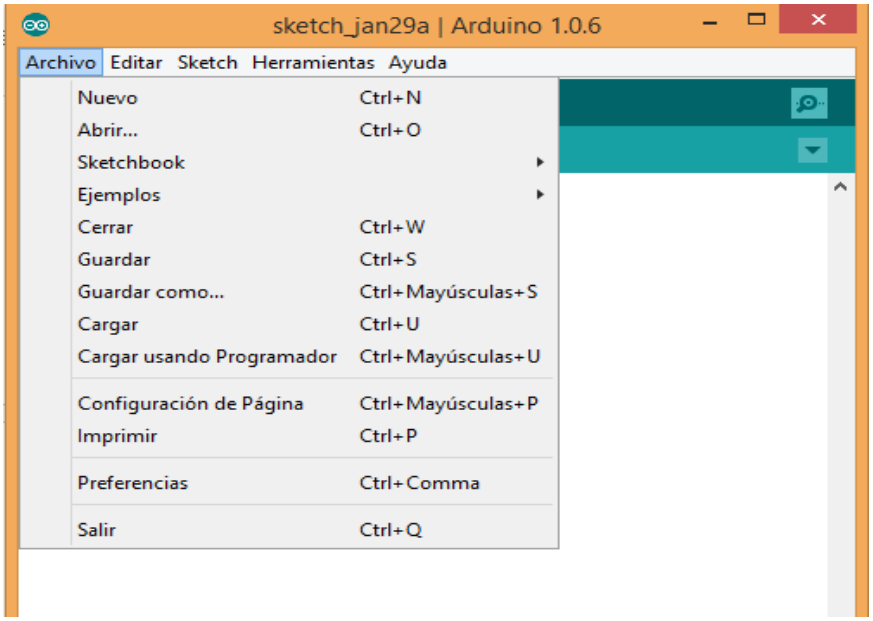


***Fig.5.2**. Menú archivo de interfaz Arduino*

- **Sketch**

En el menú Sketch volvemos a ver que las opciones de la barra de herramientas, en este caso, Verificar y compilar, también están al desplegar la pestaña.Además, vemos que tenemos la opción de Importar Librería, esto es muy necesario cuando estamos programando módulos o shields conectados al Arduino con acciones en librerías muy concretas. En este proyecto, como hemos conectado al Arduino, el módulo nRF24L01, deberemos de carga la librería del mismo, ya que, sin estos drivers, el sistema será incapaz de funcionar correctamente debido a la falta de información a la hora de ejecutar la acción programada.

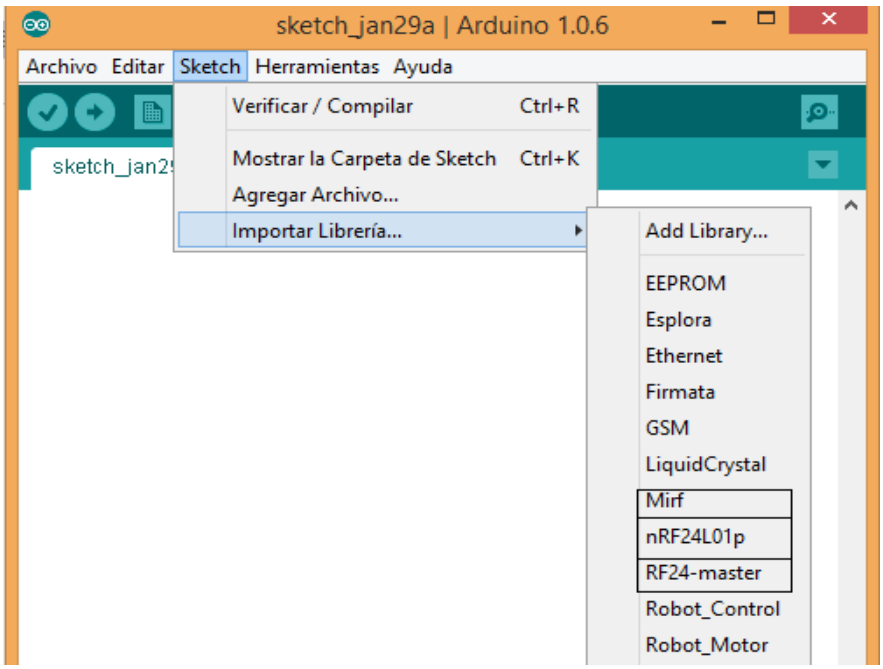


***Fig.5.3.** Menú sketch de la interfaz Arduino*

- **Herramientas**

En esta pestaña tenemos las opciones para la elección de nuestra tarjeta Arduino utilizada y el puerto serie al que está conectada; en este caso, la tarjeta será seleccionada como la Arduino Uno y, al conectarla con el USB al ordenador, éste reconoce el puerto como COM4, el cual deberemos de seleccionar desplegando la pestaña de Puerto Serial.

En el caso de tener dos o más tarjetas Arduino conectadas a la vez al ordenador, debemos abrir la opción de dispositivos en el Panel de Control para distinguir los diferentes puertos asignados a cada dispositivo ya que, en este caso, al abrir la pestaña de Puerto Serial, aparecerían varias opciones de conexión.

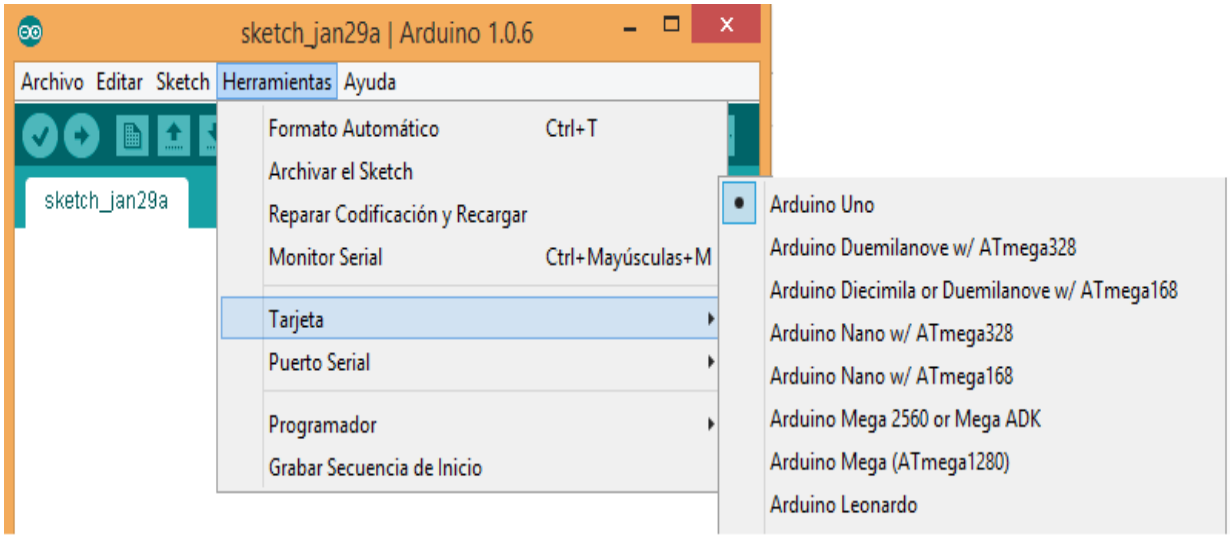


***Fig.5.4**.Menú "Tarjeta" en el menó Herramientas de la interfaz Arduino*

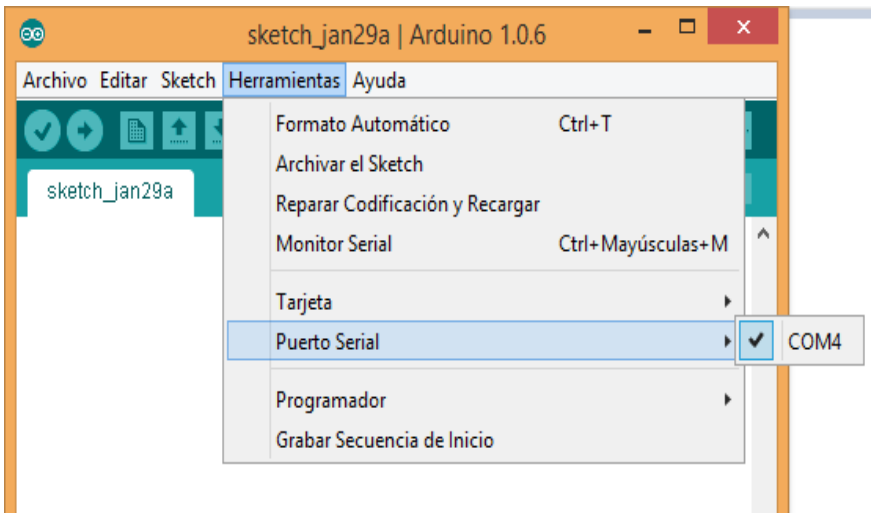


***Fig.5.5.** Menú "Puerto Serial" en el menú Herramientas de la interfaz Arduino*

5.2. LENGUAJE DE PROGRAMACIÓN

C es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.
Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.
Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

Es el lenguaje de programación de propósito general asociado al sistema operativo UNIX. Es un lenguaje de medio nivel que trata con objetos básicos como caracteres, números, también con bits y direcciones de memoria. Posee una gran portabilidad por ello se utiliza para la programación de sistemas: construcción de intérpretes, compiladores, editores de texto, etc.
El lenguaje C consta de C propiamente dicho: tipos de datos, expresiones y estructuras de control, extensiones en forma de macros y un amplio conjunto de librerías predefinidas.
Un programa C consta de uno o más módulos (ficheros fuentes) el cual puede contener las directivas del precompilador, p.e para “incluir” otros ficheros (#include) y “definir” constantes y macros (#define), declaraciones de variables y prototipos de funciones, una o más funciones y comentarios.

Cada función puede contener diferentes directivas del precompilador, declaraciones y uno o más bloques de comentarios. Cada sentencia debe estar terminada por ; cada bloque de sentencias se encierra entre llaves {. . . } y la función denominada main es la que primero se ejecuta. Los comentarios pueden aparecer en cualquier lugar del código y se insertan entre /* */ o // //.

El lenguaje de programación que utiliza la plataforma Arduino es un lenguaje nativo y derivado del C++. Este es un lenguaje muy extendido, que posee múltiples librerías y documentación. La IDE de Arduino con una librería de C/C++ llamada “Wiring”, la cual hace que las típicas operaciones de entrada/salida resulten más sencillas.

Una función es un bloque de código identificado por un nombre y que es ejecutado cuando la función es llamada. La declaración de una función incluye en primer lugar el tipo de datos que devuelve la función (e.j. int si lo que devuelve es un valor entero). Después del tipo de datos se especifica el nombre de la función y los parámetros de la misma.
A continuación, haremos un resumen de las funciones y sintaxis básicas que podremos usar a la hora de programar en nuestro sketch:

Sintaxis básica

- Delimitadores:;, {}
- Comentarios: //, /* */
- Cabeceras: #define, #include
- Operadores aritméticos: +, -, *, /, %
- Asignación: =
- Operadores de comparación: ==, !=, <, >, <=, >=
- Operadores Booleanos: &&, ||, !
- Operadores de acceso a punteros: *, &
- Operadores de bits: &, |, ^, ~, <<, >>
- Operadores compuestos:
  -Incremento y decremento de variables: ++, --
  -Asignación y operación: +=, -=, *=, /=, &=, |=

Estructuras de control

- Condicionales: if, if...else, switch case
- Bucles: for, while, do... while
- Bifurcaciones y saltos: break, continue, return, goto

Constantes

- HIGH/LOW: representan los niveles alto y bajo de las señales de entrada y salida. Los niveles altos son aquellos de 3 voltios o más.
- INPUT/OUTPUT: entrada o salida.
- false (falso): Señal que representa al cero lógico. A diferencia de las señales HIGH/LOW, su nombre se escribe en letra minúscula.
- true (verdadero): Señal cuya definición es más amplia que la de false. Cualquier número entero diferente de cero es "verdadero".

Tipos de datos

void, boolean, char, unsigned char, byte, int, unsigned int, word, long, unsigned long, float, double, string, array.

Conversión entre tipos

Estas funciones reciben como argumento una variable de cualquier tipo y devuelven una variable convertida en el tipo deseado;

char(), byte(), int(), word(), long(), float()

Cualificadores y ámbito de las variables

- static, volatile, const

Funciones básicas

E/S digital
- pinMode(pin, modo)
- digitalWrite(pin, valor)
- int digitalRead(pin)

E/S analógica
- analogReference(tipo)
- int analogRead(pin)
- analogWrite(pin, valor)

Tiempo
- delay(ms)
- delayMicroseconds(microsegundos)

Bits y Bytes

- lowByte(), highByte(), bitRead(), bitWrite(), bitSet(), bitClear(), bit()

Interrupciones externas

- attachInterrupt(interrupción, función, modo)
- detachInterrupt(interrupción)

Comunicación por puerto serie

Las funciones de manejo del puerto serie deben ir precedidas de la palabra "Serial" aunque no necesitan ninguna declaración en la cabecera del programa. Por esto se consideran funciones base del lenguaje. Estas son las funciones para transmisión serial:

- begin(), available(), read(), flush(), print(), println(), write()

Vemos pues que la estructura básica de programación de Arduino es bastante simple de manera que divide la ejecución en dos partes; Setup y Loop. Setup() constituye la preparación del programa y Loop() la ejecución; en la función Setup() se incluye la declaración de variables y se trata de la primera función que se ejecuta en el programa.

Esta función se ejecuta una única vez y es utilizada para configurar en PinMode() (por ejemplo si un determinado pin digital es de entrada o de salida) e inicializar la comunicación serie.La función Loop() incluye el código a ser ejecutado continuamente (leyendo las entradas y salidas de la placa).

5.2.1. Ejemplos de programación con Arduino

Veremos algunos ejemplos incluidos ya en el programa Arduino así como las librerías asignadas, de manera que comprendamos mejor el funcionamiento de la programación, asignación de variables y tipos de entradas y salidas.

- **Blink**

En este ejemplo el LED conectado al pin 13 parpadea cada segundo, el LED se configura como salida digital;

```
void setup() {
  // El pinl3 será una salida digital
  pinMode(13, OUTPUT);
}

// el bucle se ejecutará n veces
void loop() {
  digitalWrite(13, HIGH);   // enciende el LED, voltaje a nivel ALTO
  delay(1000);              // Espera ls
  digitalWrite(13, LOW);    // apaga el LED, voltaje a nivel BAJO
  delay(1000);              // espera ls
```

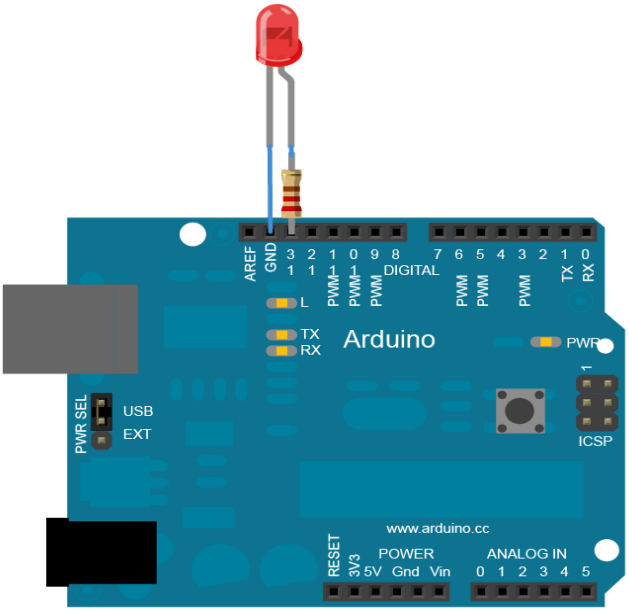


***Fig.5.6**. Montaje Blink*

- **Analog Input**

Un potenciómetro proporciona una resistencia variable, que se puede leer en la placa Arduino como un valor analógico. En este ejemplo, se conecta un potenciómetro a una de las entradas analógicas del Arduino para controlar la velocidad a la que el LED en el pin 13 parpadea.

```
int sensorPin = A0;    // Selecciona la entrada y la asigna al potenciómetro
int ledPin = 13;      // Selecciona el pin 13 para asignarlo al LED
int sensorValue = 0;  // variable para ir acumulando el valor proveniente sensor

void setup() {
  // declara como SALIDA el ledPin 13
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // lee el valor del sensor
  sensorValue = analogRead(sensorPin);
  // enciende el led
  digitalWrite(ledPin, HIGH);
  // para el programa durante <sensorValue> milisengundos
  delay(sensorValue);
  // apaga el led
  digitalWrite(ledPin, LOW);
  // para el programa durante <sensorValue> milliseconds:
  delay(sensorValue);
}
```

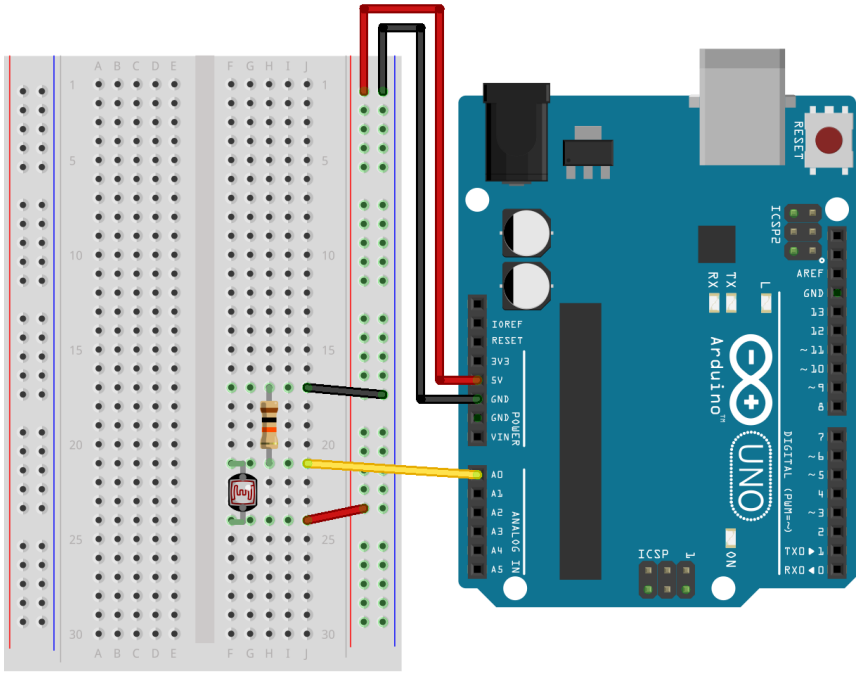


Fig.5.7. *Montaje Analog Input*

- **Physical pixel**

Este ejemplo utiliza la placa Arduino para recibir datos desde el ordenador. La placa Arduino enciende un LED cuando recibe el carácter 'H', y apaga el LED cuando recibe el carácter 'L'.

Los datos pueden ser enviados desde el puerto serial de, u otro programa como Processing , Flash , PD, o Max / MSP.

```
const int ledPin = 13; // declara el pin 13
int incomingByte;      // declara la variable para leer la entrada de datos

void setup() {
  // inicializa comunicación serial a una velocidad de 9600 baudios
  Serial.begin(9600);
  // pin13 como salida
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // ve si hay datos de llegada
  if (Serial.available() > 0) {
    // lee el último byte en el almacenamiento serial
    incomingByte = Serial.read();
    // si hay una H mayúscula(ASCII 72), enciende LED:
    if (incomingByte == 'H') {
      digitalWrite(ledPin, HIGH);
    }
    // si hay una L mayúscula (ASCII 76) apaga LED:
    if (incomingByte == 'L') {
      digitalWrite(ledPin, LOW);
    }
  }
```

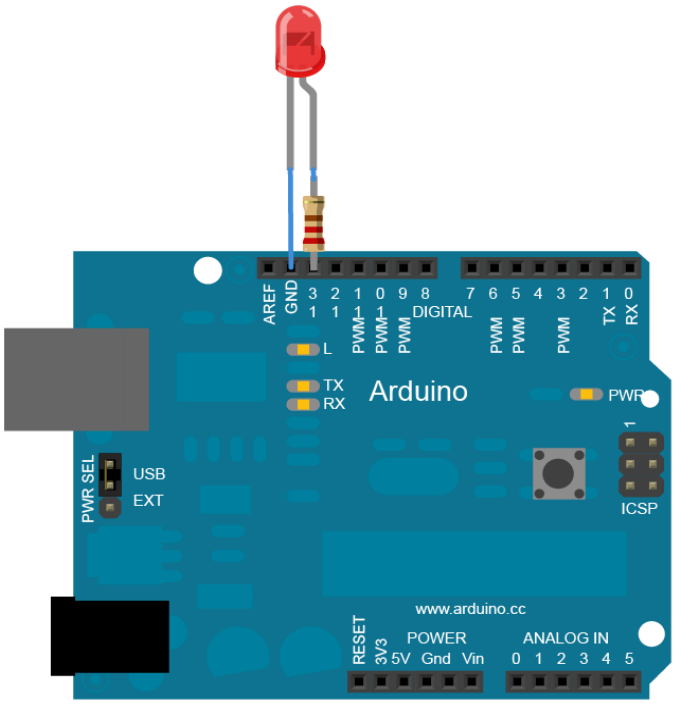


***Fig.5.8**. Montaje Phisycal Pixel*

5.3. DESARROLLO DE PROTOTIPOS

Por fin, en este apartado, se procederá a la proyección de lo anteriormente visto, esto es, se aplicarán los materiales, conexiones, montajes y programación explicados, de manera que el proyecto final se ejecute correctamente basado en los conocimientos explicados y todo lo anterior se haga finalmente tangible.

Se verá tanto la arquitectura de cada módulo del prototipo por separado, es decir, por un lado tendremos el montaje y arquitectura del módulo de radio emisor así como su correspondiente código de programación, todo ello correspondiente a fin de cuentas al mando de control, y por otro lado, la arquitectura, montaje y código del módulo receptor, lo que es el vehículo, junto a la pequeña tarjeta con el integrado L293D, controlado por radio desde el módulo emisor.

5.3.1. Módulo emisor

Pues bien, este apartado se centrará totalmente en la fabricación y puesta en marcha del módulo emisor de radio; el mando de control.

Empezando por la arquitectura, el montaje final tanto en imágenes como en montaje de conexiones electrónicas, pasando por el proceso de soldadura de las mismas, y acabando con

el resultado final del mando, se dará a conocer también el código utilizado para su implementación, tanto individualmente para el control de puerto serial y la posición del Joystick como, el global de comunicación entre módulos.

5.3.1.1. ARQUITECTURA

La arquitectura del mando será bastante más sencilla y simplificada que la del coche, ya que básicamente, constará del Arduino con el código emisor, del Joystick, que comandará los movimientos del vehículo, y del transceptor nRF24L01 que emitirá por radiofrecuencia dichos datos al módulo emisor, siendo recibidos por el transceptor del mismo.
Dichos elementos anteriormente citados irán encapsulados en una caja que hará las veces de mando donde, habiendo hecho una ranura para el cableado, colocaremos el joystick sobre la parte superior dejándolo en el exterior para el correcto uso del mismo.

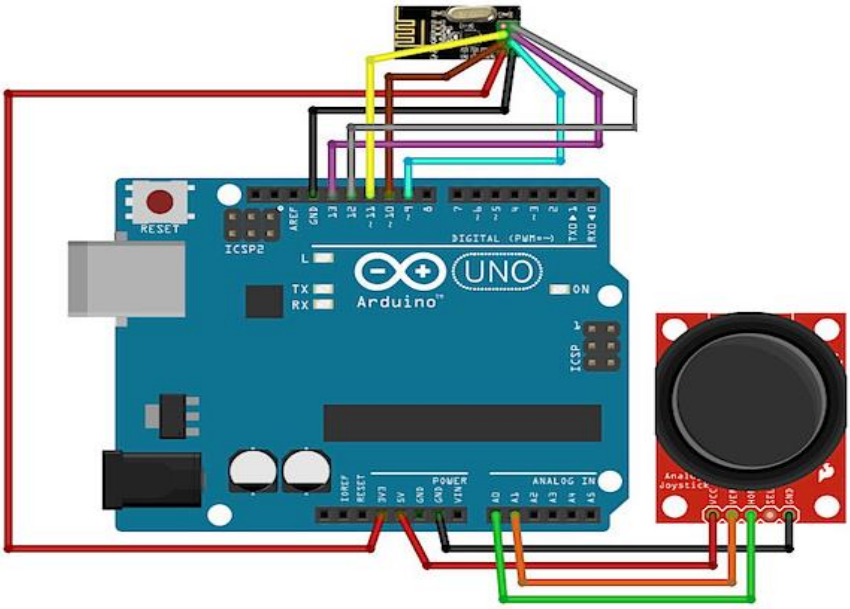


Fig. 5.9. *Montaje de conexiones del Joystick y del Nrf24L01 al Arduino Uno*

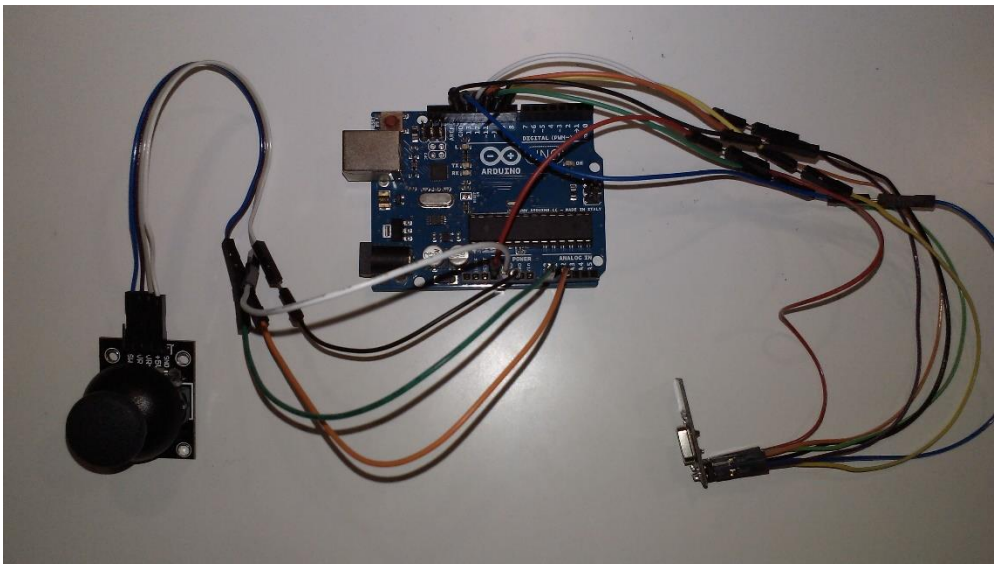


Fig.5.10. *Conexionado de elementos en el módulo emisor*

 Por causas físicas ajenas que ocurrieron tras el proceso de programación fue imposible la utilización del joystick como comanda de los movimientos del módulo receptor y como parte básica del módulo emisor, por ello se programará como emisor el ordenador al cual estará conectado el Arduino emisor junto con el transceptor de radio nRF24L01.

Así, en lugar de coordenadas emitidas por el joystick, tendremos caracteres asociados a cada uno de los estados del motor como mostraremos a continuación.

5.3.1.2. PROGRAMACIÓN

Para la programación tendremos un programa de prueba para la comprobación del correcto funcionamiento del módulo de radio en función de las posiciones del joystick, que harán que el módulo receptor de prueba reciba los datos y los muestre a través del Com Port de la interfaz Arduino.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#define CE_PIN 9
#define CSN_PIN 10
#define JOYSTICK_X A0
#define JOYSTICK_Y A1

const uint64_t pipe = 0xE8E8F0F0E1LL;

RF24 radio(CE_PIN, CSN_PIN);
int joystick[2];

void setup()
{
Serial.begin(9600);
radio.begin();
radio.openWritingPipe(pipe);
}
```

```
void loop()
{
joystick[0] = analogRead(JOYSTICK_X);
joystick[1] = analogRead(JOYSTICK_Y);
radio.write( joystick, sizeof(joystick) );
}
```

Como en el código de prueba del transceptor del módulo receptor, la cabecera será la misma, es decir, tendremos declaradas las tres librerías necesarias para que se abra la comunicación por radio; *SPI*, para el protocolo de comunicación, y *nRF24L01* y *RF24* para el módulo de radiofrecuencia.

Definimos los pines 9 y 10 de nuestro Nrf24L01 emisor como los pines que transmiten los datos, CE y CSN, además también se declararán los ejes X e Y del Joystick de manera que se establezcan las entradas analógicas A0 y A1 del Arduino.

También se declara la dirección de memoria donde se almacenarán los datos, que deberá ser la misma tanto en este módulo emisor como en el receptor, esta es; *0xE8E8F0F0E1LL.*

Se inicializa la variable Joystick mediante un array y comenzamos con nuestro setup.
En el ***setup*** tendremos la velocidad de comunicación a 9600 baudios que deberá coincidir con la del módulo receptor, después se comenzará con la emisión de datos por radiofrecuencia, estos datos serán guradados en la dirección de memoria declarada anteriormente.

En el ***loop*** del programa se ejecutará la lectura de las entradas analógicas de los ejes X e Y del joystick, conectadas al Arduino, transmitiéndolas por radio una vez leídas, esto es, el programa leerá las posiciones del joystick interpretándolas como tamaño de bits, esto es lo que será enviado, de manera que el receptor recibirá señales de X en función de Y de un tamaño que variará de 0 a 1023 bits es decir, 10 bytes.

Ahora bien, el código anterior utilizaría el joystick como interfaz de control entre el coche y el usuario pero, como se ha comentado con anterioridad, existieron fallos físicos posteriores en el mismo y, se optó por optimizar tiempo empleando al propio ordenador como interfaz, creando caracteres asociados a los movimientos de los motores del coche.

Este nuevo programa nuevo hará las veces de joystick, es decir, tomará los valores que hubiera tomado el joystick según la dirección que tomara.

A continuación, el código final utilizado en el proyecto usando el módulo emisor compuesto por el nRF24L01 como transceptor de radio, en este caso, emisor, el Arduino Uno con el programa emisor cargado, el ordenador, el cual estará alimentando y, a su vez, enviando las órdenes deseadas por el usuario al Arduino.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#define CE_PIN 9
#define CSN_PIN 10
#define JOYSTICK_X A0
#define JOYSTICK_Y A1
const uint64_t pipe = 0xE8E8F0F0E1LL;
char letra;
RF24 radio(CE_PIN, CSN_PIN);
int joystick[2];

void setup()
{
Serial.begin(9600);
radio.begin();
radio.openWritingPipe(pipe);
letra='s';
}

void loop()
{
Serial.println(letra);
letra=Serial.read();
if(letra=='s'){joystick[0] =520; joystick[1] =520;}
else if(letra=='f'){joystick[0] =520; joystick[1] =0;}
else if(letra=='b'){joystick[0] =520; joystick[1] =1023;}
else if(letra=='l'){joystick[0] =0; joystick[1] =520;}
else if(letra=='r'){joystick[0] =1023; joystick[1] =520;}
radio.write( joystick, sizeof(joystick) );
}
```

Comenzando, vemos que la cabecera de las cuatro primeras líneas es la misma que la que existe en el programa de código receptor, asegurando así la correcta comunicación entre las partes; las bibliotecas *SPI, Nrf24L01* y *RF24* se ocupan de los protocolos de comunicación y de características de radiotransmisión específicas del transceptor Nrf24l01, también definiremos los pines 9 y 10 del mismo, CE y CSN, los cuales llevarán a cabo esta labor.

Se definirán dos constantes; JOYSTICK_X y JOYSTICK_Y que están asociadas a los pines A0 y A1 analógicos del Arduino.

La dirección de memoria a la cual se irán mandando los datos será la *0xE8E8F0F0E1LL.*
Ahora es cuando declaramos la variable *letra* de tipo *char* que será nuestra variable de movimiento sustituto del joystick.
Vemos que tenemos una variable entera, *joystick*, array de dos variables; *joystick[0]* hace referencia al eje X y *joystick[1]* al Y.

En el ***setup*** tenemos el inicio de la comunicación serial a 9600 baudios, al igual que también se inicializa la radio abriendo la escritura de línea o dirección de datos; en el inicio estaremos en el estado de paro ya que se está mandando, en primer lugar, la letra '*s'* (stop)

En el ***loop*** tenemos una impresión serial del carácter *letra* introducido manualmente mediante el teclado del ordenador, éste carácter será leído e interpretado por el programa y asociado tanto a las variables *joystick[0]* y *joystick[1]* como a los movimientos asociadas a dichos caracteres.
El joystick tiene dos entradas analógicas que van de 0 a 1023, por lo que los movimientos asociados tendrán valores dentro de este rango.

LETRA	MOVIMIENTO	JOYSTICK[0]_x	JOYSTICK[1]_y
s	Stop	520	520
f	Forward	520	0
b	Backward	520	1023
l	Left	0	520
r	Right	1023	520

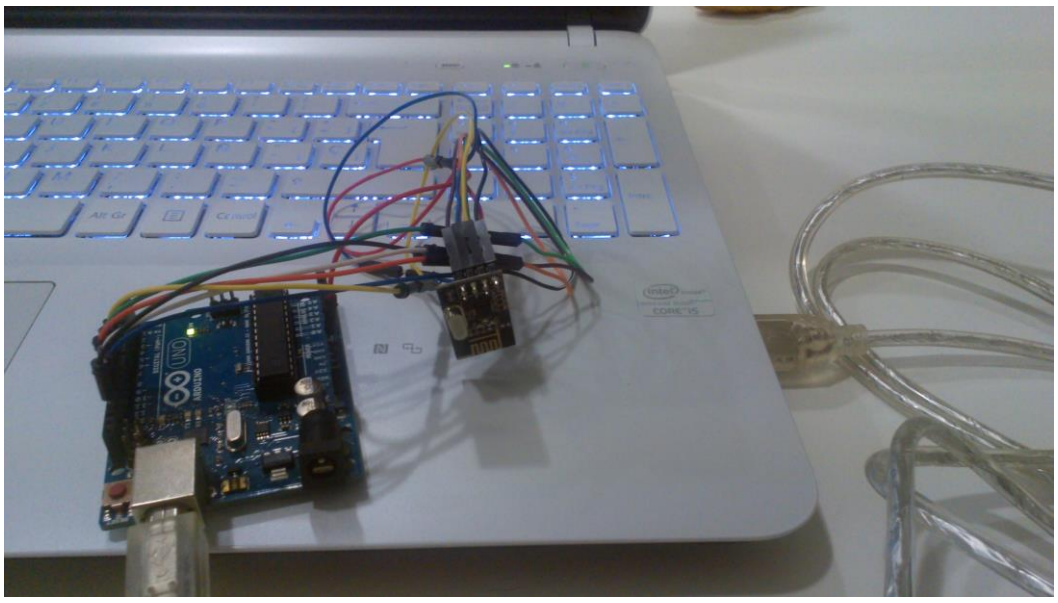


Fig.5.11*. Conexión del módulo emisor, Arduino+Nrf24L01, por USB al ordenador*

5.3.2. Módulo receptor

En este subapartado, se hablará del módulo receptor, en otras palabras, del vehículo radiocontrolado. Como en el anterior apartado, que hace referencia al módulo de mando emisor, en éste también hablaremos del proceso de fabricación, soldadura de componentes y puesta en marcha del coche.

De la arquitectura del kit-car ya se hizo alusión en capítulos anteriores, así que aquí explicaremos el proceso de montaje añadiendo las imágenes del mismo, soldadura de la tarjeta del integrado inversor de giro L293D y colocación de los diferentes elementos sobre la estructura vacía del kit-car.

También se dará a conocer el código de programación que se habrá cargado en el Arduino para que, con el movimiento del joystick del mando de control, interprete las señales que mandará a los motores para el movimiento requerido para el vehículo. Antes de este código final se probarán los motores para comprobar su correcto funcionamiento tras la soldadura al L293D.

5.3.2.1. ARQUITECTURA

Pues bien, la arquitectura del vehículo estará formada por el mismo montaje del coche, explicado con anterioridad, con la diferencia de que, sobre él, se colocará todo el sistema de comunicación y alimentación del proceso.

Sobre la placa de metacrilato transparente del coche, se situarán el Arduino Uno, la tarjeta con el integrado L293D, la pila de 9V 6LR61 que alimentará el pin 8 Vs del integrado y al que irán conectados también los pines de tierra 12 y 13. Los pines 1 y 9, Enable1 y Vss respectivamente, irán conectados a la salida de alimentación de 5V que tiene el Arduino, así, los pines 4 y 5 van a la salida GND del Arduino.
A su vez, los pines 2 y 7 del integrado irán a las entradas digitales 4 y 3 del Arduino, que harán referencia al motor izquierdo y los pines 10 y 15 del integrado a las entradas digitales 5 y 6 del Arduino, referentes ahora al motor derecho.

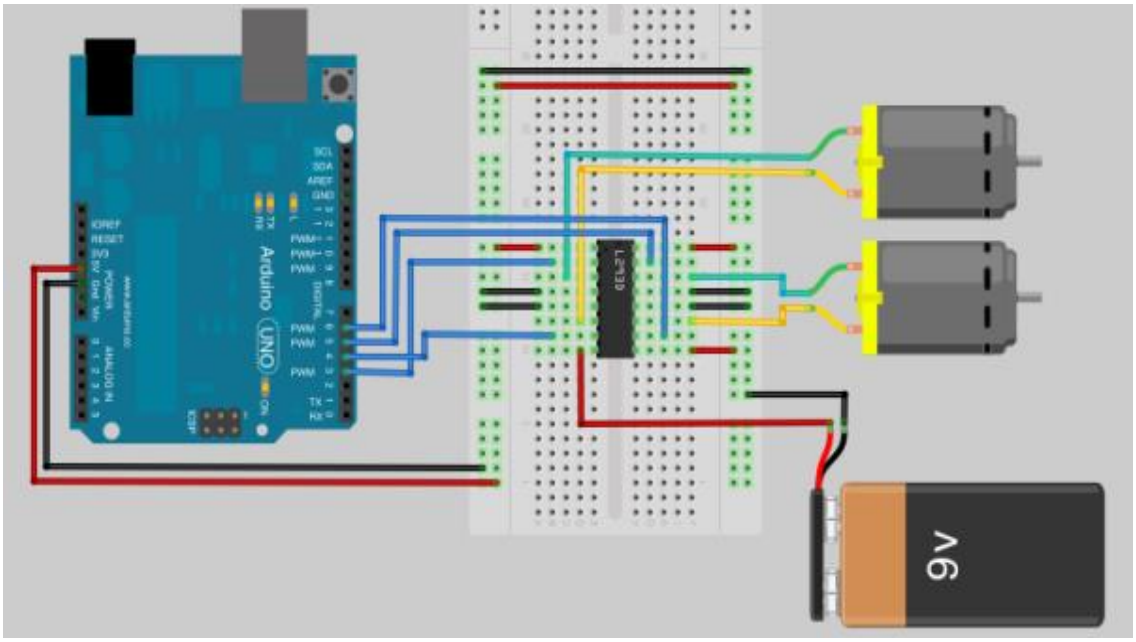


Fig.5.12. *Montaje de conexión del L293D al Arduino Uno y a ambos motores*

Durante el proceso de pruebas, se está conectando y alimentado al Arduino mediante USB con el ordenador, pero en el momento en el que esté todo a punto y acabado, éste será alimentado por cuatro pilas AAA de 1,5V, es decir, tendrá una alimentación de 6V; éstas pilas irán sobre un zócalo colocado también sobre el vehículo, en la parte delantera.

Decir que para la colocación de todos estos elementos sobre la placa y la posterior sujeción de los mismo a ella, se ha utilizado cinta adhesiva de doble cara ya que, atornillar los mismo al metacrilato podría acarrear grietas y fracturas en el material debido al calentamiento momentáneo del mismo a la hora de hacer el taladro o en el momento de apretar las tuercas.

La soldadura del integrado se ha hecho con un soldador de 30W junto con una aleación de estaño 60-40, 60% estaño y 20% plomo sobre una tarjeta presoldada para la introducción de los pines del integrado.

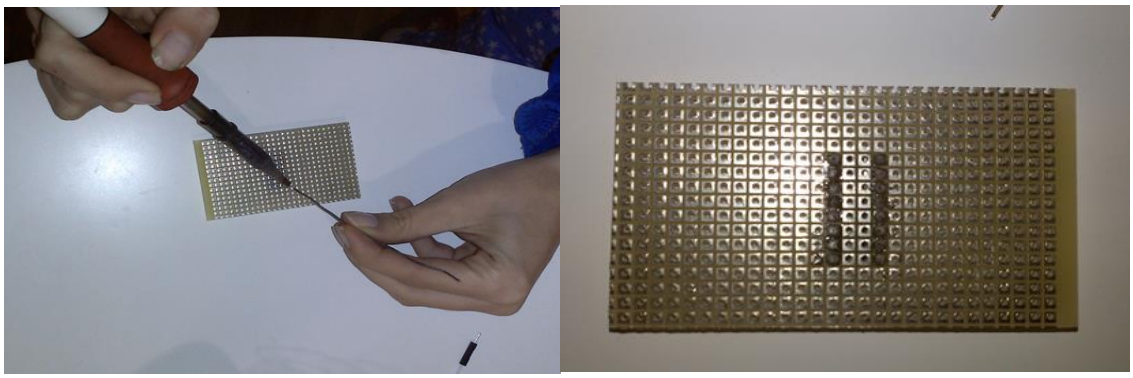


Fig. 5.13. *Detalle del proceso de soldadura (izda.) y vista del integrado en la parte posterior de la tarjeta*

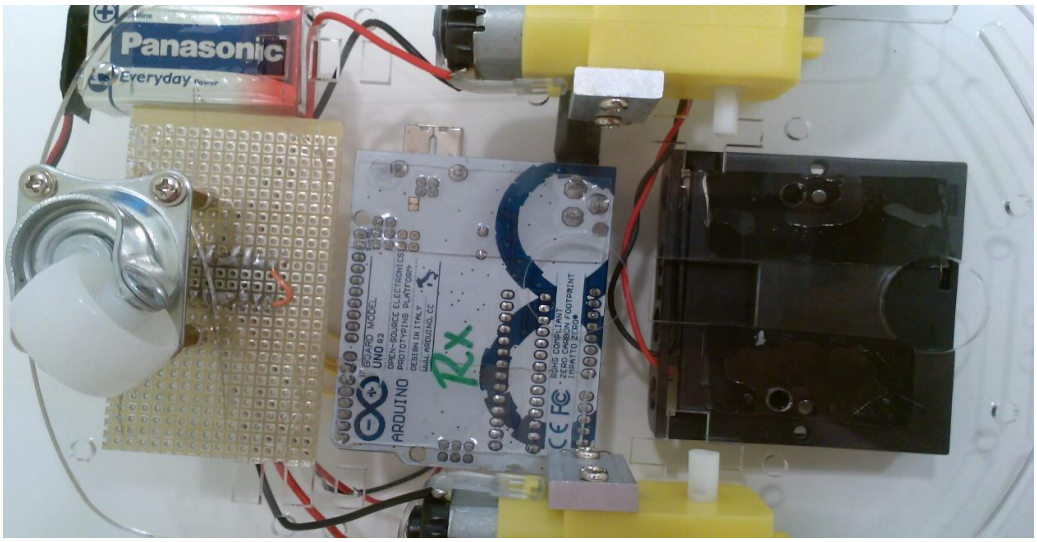


Fig. 5.14. *Vista de la planta del vehículo donde se pueden ver todos los elementos adheridos con adhesivo*

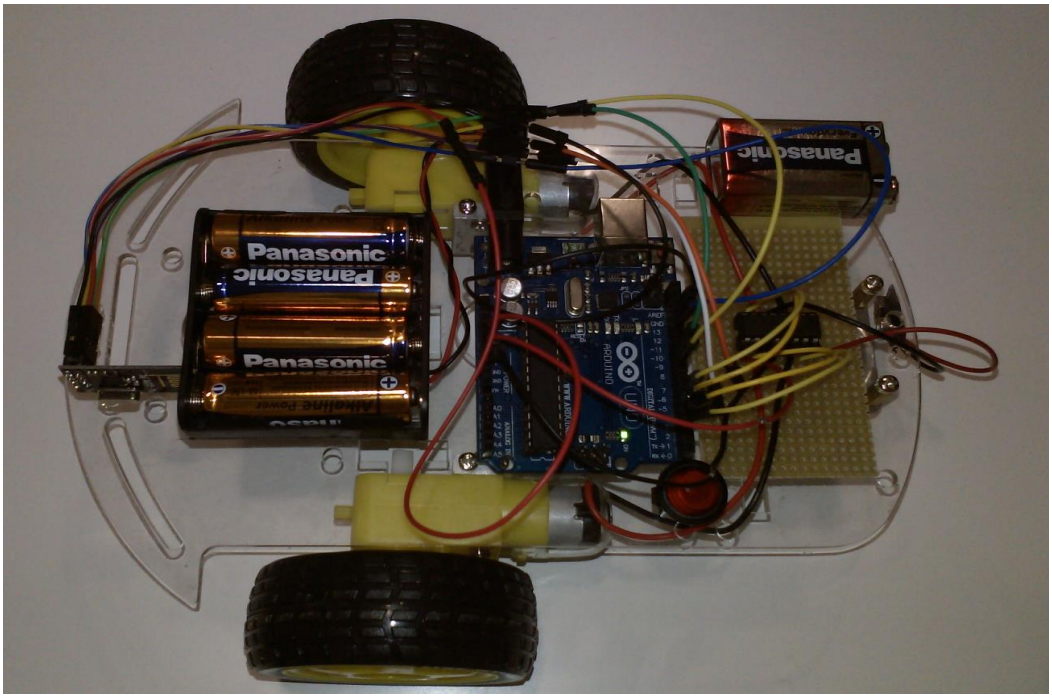


***Fig.5.15**. Resultado final del montaje físico del módulo receptor*

5.3.2.2. PROGRAMACIÓN

Aquí es donde se verá el código con el que se ha programado el Arduino para que, siguiendo las órdenes de mando del joystick del módulo emisor, el receptor las ejecute e interprete de manera deseada.

Antes de programar el conjunto del código final, una vez montado el módulo receptor, probaremos que los motores actúan de manera correcta, esto es básicamente, que las ruedas se muevan en función de una señal dada, en este caso, programada, para que las mismas funcionen automáticamente.

// Control de los dos motores DC con el driver de puente en H, L293D H-Bridge driver

```
int leftMotor[]  = {3,4};
int rightMotor[] = {5,6};

void setup(){

Serial.begin(9600);
int motorLeg;
for(motorLeg = 0; motorLeg < 2; motorLeg++){
pinMode(leftMotor[motorLeg], OUTPUT);
pinMode(rightMotor[motorLeg], OUTPUT);
}

}
```

```
void loop(){

driveMotorsForward();
delay(1000);
stopMotorsForward();
delay(3000);

driveMotorsForwardRight();
delay(1000);
stopMotorsForward();
delay(3000);

driveMotorsForwardLeft();
delay(1000);
stopMotorsForward();
delay(3000);

driveMotorsBackward();
delay(1000);
stopMotorsBackward();
delay(3000);

driveMotorsBackwardRight();
delay(1000);
stopMotorsBackward();
delay(3000);

driveMotorsBackwardLeft();
delay(1000);
stopMotorsBackward();
delay(3000);

}


void driveMotorsForward(){
digitalWrite(leftMotor[0], HIGH);
digitalWrite(leftMotor[1], LOW);
digitalWrite (rightMotor[0], LOW);
digitalWrite(rightMotor[1], HIGH);
}
void driveMotorsForwardRight(){
digitalWrite(leftMotor[0], LOW);
digitalWrite(leftMotor[1], LOW);
digitalWrite(rightMotor[0], LOW);
digitalWrite(rightMotor[1], HIGH);
}
void driveMotorsForwardLeft(){
digitalWrite(leftMotor[0], HIGH);
digitalWrite(leftMotor[1], LOW);
digitalWrite(rightMotor[0], LOW);
digitalWrite(rightMotor[1], LOW);
}
```

```
void stopMotorsForward(){
digitalWrite(leftMotor[0], LOW);
digitalWrite(leftMotor[0], LOW);
digitalWrite(rightMotor[1], LOW);
digitalWrite(rightMotor[1], LOW);
}
void driveMotorsBackward(){
digitalWrite(leftMotor[0], LOW);
digitalWrite(leftMotor[1], HIGH);
digitalWrite(rightMotor[0], HIGH);
digitalWrite(rightMotor[1], LOW);
}
void driveMotorsBackwardRight(){
digitalWrite(leftMotor[0], HIGH);
digitalWrite(leftMotor[1], HIGH);
digitalWrite(rightMotor[0], HIGH);
digitalWrite(rightMotor[1], LOW);
}
void driveMotorsBackwardLeft(){
digitalWrite(leftMotor[0], LOW);
digitalWrite(leftMotor[1], HIGH);
digitalWrite(rightMotor[0], HIGH);
digitalWrite(rightMotor[1], HIGH);
}
void stopMotorsBackward(){
digitalWrite(leftMotor[0], HIGH);
digitalWrite(leftMotor[0], HIGH);
digitalWrite(rightMotor[1], HIGH);
digitalWrite(rightMotor[1], HIGH);
}
```

El sistema debe empezar con unos valores iniciales que debemos de establecer nosotros, es decir, asignamos los valores correspondientes a los pines conectados del Arduino al motor derecho y al motor izquierdo, es decir; el motor izquierdo, *LeftMotor*, se asociarán a los pines 3 y 4 y el motor derecho, *RightMotor*, a 5 y 6.

El ***setup*** de nuestro programa receptor contiene los pines que conforman el módulo receptor y que son de tipo salida, OUTPUT, el *LeftMotor* y el *RightMotor*. Además, se inicializa la variable *MotorLeg* donde, dentro de la estructura *for*, se inicializa a 0. La velocidad de comunicación serial será de 9600 baudios.

Ahora, en el ***loop***, se hace precisamente lo que su nombre sugiere; bucles de forma consecutiva, permitiendo al programa cambiar y responder de manera que se utiliza para controlar activamente la placa Arduino. Aquí, se controlará el movimiento del coche, esto es, hacia adelante, *MotorsForward*, a la derecha y hacia adelante *MotorsForwardRight*, hacia la izquierda y hacia adelante, *MotorsForwardLeft*, parada de movimientos hacia adelante *stopMotorsForward,* hacia atrás, *MotorsBackward*, hacia la derecha a la vez que atrás, *MotorsBackwardRight*, a la izquierda a la vez que atrás, *MotorsBackwardLeft,* y parada de los movimientos hacia atrás, *stopMotorsBackward*.

A continuación, se declararán estas funciones anteriormente descritas. Para resumirlas, se pondrá la tabla de valores lógicos empleados para la activación-desactivación de las mismas;

MotorLeft			***MotorRight***			
P3	**P4**	**Acción**	**P5**	**P6**	**Acción**	**Variable asignada**
1	0	ADELANTE	0	1	ADELANTE	*driveMotorsForward*
0	0	PARO	0	1	ADELANTE	*driveMotorsForwardRight*
1	0	ADELANTE	0	0	PARO	*driveMotorsForwardLeft*
0	0	PARO	0	0	PARO	*drivestopMotorsForward*
0	1	ATRÁS	1	0	ATRÁS	*driveMotorsBackward*
1	1	PARO	1	0	ATRÁS	*driveMotorsBackwardRight*
0	1	ATRÁS	1	1	PARO	*driveMotorsBackwardLeft*
1	1	PARO	1	1	PARO	*drivestopMotorsBackward*

Los pines 3 y 4 pertenecen al motor izquierdo y los pines 5 y 6 al derecho. Teniendo en cuenta que la entrada Enable está siempre a nivel alto, 1, ya que está conectada directamente a la alimentación de 5V del Arduino, asignaremos valores a cada pin, es decir, cada motor tendrá un estado de 2 bits:

 Aclarar que en el motor izquierdo, hemos invertido los valores de manera que, donde debiera haber un 1 hay un 0 y viceversa ya que, en el proceso de soldadura, los cables que van hacia el motor se han conectado de manera contraria y se consideró más fácil la opción de invertir los valores a la hora de programar.

- Cuando los 4 pines están todos a 0 o a 1, el estado será el de paro.
- En el motor izquierdo, los valores “10” (invertidos a los valores del motor derecho, por la razón explicada en el punto anterior) para sus pines P4P5 harán que se mueva hacia adelante, “01” hacia atrás y “00” paro.
- En el motor derecho, los valores “01” harán que éste se mueve hacia adelante, “10” hacia atrás y “11” paro.

Tenemos pues, 8 estados bien diferenciados, que ya son conocidos y han sido declarados en nuestro programa.

 Ahora bien, no se debe confundir el motor que está activo, con la dirección que toma el vehículo, es decir, por ejemplo; *MotorsForwardRight* hace referencia al motor que se mueve y en qué dirección lo hace y no en qué dirección se mueve el vehículo, esto es, se moverá el motor derecho hacia adelante, pero el vehículo en su totalidad girará hacia la izquierda ya que el motor izquierda permanecerá parado.
Y lo mismo ocurre con el resto de estados excepto con las paradas y los movimientos hacia adelante y hacia atrás simultáneos; *MotorsForward, stopMotorsForward, MotorsBackward y stopMotorsBackward.*

Como se ha visto al final del apartado del módulo emisor, por problemas del joystick,éste no puede ser usado como emisor de señal o datos para el movimiento del coche, por lo que se tendrá que adaptar el códio del programa receptor, como se ha hecho con el emisor, de manera que, los caracteres enviados por el ordenador sean recibidos e interpretados por el mismo corresctamente, haciendo que el vehículo se mueva en la dirección asociada y demandada.

A continuación, se mostrará el programa final utilizado para su ejecución.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#define CE_PIN 9
#define CSN_PIN 10
#define JOYSTICK_X A0
#define JOYSTICK_Y A1

const uint64_t pipe = 0xE8E8F0F0E1LL;
RF24 radio(CE_PIN, CSN_PIN);

int leftMotor[]  = {3,4};
int rightMotor[] = {5,6};

int joystick[2];
int movimiento;
int motorLeg;


void setup(){

Serial.begin(9600);
delay(1000);
Serial.println("Nrf24L01 Receiver Starting");
radio.begin();
radio.openReadingPipe(1,pipe);
radio.startListening();


movimiento=0;

for(motorLeg = 0; motorLeg < 2; motorLeg++){

pinMode(leftMotor[motorLeg], OUTPUT);
pinMode(joystick[0], OUTPUT);

pinMode(rightMotor[motorLeg], OUTPUT);
pinMode(joystick[1], OUTPUT);
}

}
```

```
void loop(){


if ( radio.available() )
{
bool done = false;
while (!done)
{
done = radio.read( joystick, sizeof(joystick) );

Serial.print("X = ");
Serial.print(joystick[0]);
Serial.print( "Y =" );
Serial.println(joystick[1]);
Mover();
}
}

else
{
Serial.println("No radio available");
}
}


void Mover(){

detectarmovimiento();



switch(movimiento){


   case 0: stopMotorsBackward();
    Serial.println("parado");
              break;

   case 1: driveMotorsForward();
    Serial.println("adelante");
              break;

   case 2: driveMotorsBackward();
    Serial.println("atras");
             break;

   case 3: driveMotorsForwardRight();
    Serial.println("izquierda");
             break;

   case 4: driveMotorsForwardLeft();
    Serial.println("derecha");
             break;
```

```
   default: stopMotorsBackward();
    Serial.println("defecto");
            break;
 };
}

void driveMotorsForward(){
digitalWrite(leftMotor[0], HIGH);
digitalWrite(leftMotor[1], LOW);
digitalWrite (rightMotor[0], LOW);
digitalWrite(rightMotor[1], HIGH);
}

void driveMotorsForwardRight(){
digitalWrite(leftMotor[0], LOW);
digitalWrite(leftMotor[1], LOW);
digitalWrite(rightMotor[0], LOW);
digitalWrite(rightMotor[1], HIGH);
}

void driveMotorsForwardLeft(){
digitalWrite(leftMotor[0], HIGH);
digitalWrite(leftMotor[1], LOW);
digitalWrite(rightMotor[0], LOW);
digitalWrite(rightMotor[1], LOW);
}

void stopMotorsForward(){
digitalWrite(leftMotor[0], LOW);
digitalWrite(leftMotor[0], LOW);
digitalWrite(rightMotor[1], LOW);
digitalWrite(rightMotor[1], LOW);
}

void driveMotorsBackward(){
digitalWrite(leftMotor[0], LOW);
digitalWrite(leftMotor[1], HIGH);
digitalWrite(rightMotor[0], HIGH);
digitalWrite(rightMotor[1], LOW);
}

void driveMotorsBackwardRight(){
digitalWrite(leftMotor[0], HIGH);
digitalWrite(leftMotor[1], HIGH);
digitalWrite(rightMotor[0], HIGH);
digitalWrite(rightMotor[1], LOW);
}

void driveMotorsBackwardLeft(){
digitalWrite(leftMotor[0], LOW);
digitalWrite(leftMotor[1], HIGH);
digitalWrite(rightMotor[0], HIGH);
digitalWrite(rightMotor[1], HIGH);
}
```

```
void stopMotorsBackward(){
digitalWrite(leftMotor[0], HIGH);
digitalWrite(leftMotor[0], HIGH);
digitalWrite(rightMotor[1], HIGH);
digitalWrite(rightMotor[1], HIGH);
}



void detectarmovimiento(){
if(joystick[0]==1023) movimiento=4;
else if(joystick[0]==0) movimiento=3;
else if(joystick[1] >800) movimiento=2;
else if(joystick[1] <300) movimiento=1;
else movimiento=0;

}
```

En este código final se solaparán las funciones de comunicación por radio de los datos enviados de manera inalámbrica por el ordenador con las funciones de activación de los motores ya declaradas, además, se habrán asociado las medidas del joystick en función del ordenador a una variable encargada del arranque del movimiento del coche llama *movimiento.*

La cabecera siempre será igual, las librerías del transceptor y del protocolo de comunicación son básicas para la correcta transmisión de los datos módulo a módulo; *SPI*, *nRF24L01* y *RF24.*

Se definen los pines del transceptor encargados de la comunicación, el *CE* y el *CSN* asociados al pin 9 y 10 digitales del Arduino, al igual que también se definen las variables *JOYSTICK_X* en el pin analógico A0 y *JOYSTICK_Y* en el A1, que serán las variables representativas de los ejes X e Y del movimiento.

Se declara como *const* la líne a de memoria donde se registrarán los datos de llegada que coincidirá con la de los datos de salida del módulo emisor, se declaran también las variables enteras *int* que hacen alusión a los motores; leftMotorr[] y rightMotor[] cuyos pines de salida serán los (3, 4) y (5, 6) digitales en el Arduino.
Tendremos que declarar tres variables más; *joystick[2]*, array de dos caracteres asociado con los movimientos X e Y, *movimiento*, que hará la puesta en marcha de los motores a partir de la transferencia de radio desde el ordenador, y *motorLeg,* contador encargado de pasar de un movimiento a otro del motor según esté programado, todos serán enteros *int*.

En el ***setup*** tenemos el comiendo de la comunicación por radio a 9600 baudios con una pausa o *delay* de 1s donde veremos, en el Com Port del Arduino que dicha comunicación se ha establecido*, "Nrf24L01 Receiver Starting".* Además, comenzaremos habilitando la dirección de memoria haciendo que el receptor sea capaz de interpretar y guardar los datos del emisor que le están llegando, *startListening().*
Establecemos la variable *movimiento* a 0 y, dentro de una estructura *for*, inicializamos el contador *motorLeg*.
Estableceremos también los pines de los motores como salidas, *OUTPUT*, así como los del joystick.

En el ***loop*** tendremos que la comunicación será disponible si la radio también lo está, leyendo los datos de entrada como tamaño de movimiento del joystick, en su defecto, los datos establecidos como dicho tamaño en el programa emisor del ordenador, de manera que, podremos ver las coordenadas X e Y de nuestras variables *joystick[0]* y *joystick[1]* en el Com Port del programa, al igual, que comenzará el movimiento del coche con *Mover().*
Si no existiera buena comunicación entre los módulo, se recibirá un mensaje en el Com Port de *"No radio available".*

A continuación, en el *void Mover()*, se tendrá una estructurá *switch* encargada de ir escogiendo el movimiento asociado a las entradas que le van llegando; estas entradas diferentes estarán definidas como *case* , siendo 0 para *stopMotorsBackward,* 1 para *driveMotorsForward*, 2 para *driveMotorsBackward*, 3 para *driveMotorsForwardRight* y 4 para *driveMotorsForwardLeft*.
Para cada case tendremos en el Com Port un *"parado"*, *"adelante"*, *"atras"* y *"derecha"* respectivamente, además del *default*, en el que nos transmitirá *"defecto".*

Seguidamente, aparecen los ocho bloques de movimientos iniciales que se utilizaron en la prueba del movimientos del coche para el integrado L293D de los cuales, usaremos cinco; parada, adelante, atrás, derecha e izquierda.

En el último void es donde vemos la asociación de nuestros *case* del *loop* con los datos del joystick (ordenador) de manera que, teniendo un rango de valor de entrada de 0 a 1023, 10 bits, que imitarán los valores de tensión que se tendría con el joystick en cada eje, se verá para qué valor o par de valores de dicho rango se tendrá un movimiento u otro, así, para un valor igual a 1023 en el eje X el coche se moverá hacia la *"derecha"* ya que se mueve la rueda izquierda hacia adelante (*driveMotorsForwardLeft*), para un valor de 0 en el eje X el coche se moverá a la *"izquierda*" ya que se mueve la rueda derecha hacia adelante (*driveMotorsForwardRight*), para un valor en el eje Y mayor de 800, tendremos el movimiento hacia *"atrás*" (*driveMotorsBackward*), para Y menor de 300, hacia "*adelante" (driveMotorsForward*) si no tenemos valor alguno o no hay señal de radio, tendremos el movimiento puesto a 0,es decir*, "parado".*

CAPÍTULO 6

CONCLUSIONES, TRABAJOS FUTUROS Y PRESUPUESTO

En este último capítulo se expondrán las conclusiones que se obtienen tras la elaboración del presente proyecto. Se explicarán los problemas más relevantes que han ido surgiendo en la elaboración del control por radiofrecuencia inalámbrico entre ambos módulos y la forma de solucionarlos.

A continuación se realizará una valoración a título personal sobre lo que supone al autor el desarrollo y elaboración del proyecto. Por último se trazarán aquellas posibles líneas en las que se puede orientar la elaboración de futuros trabajos y proyectos relacionados con el mundo de la comunicación a larga y corta distancia a través de transceptores o módulo de radio y, en particular, con el proyecto expuesto en esta memoria.

Además, se hará un presupuesto de materiales y herramientas utilizadas para la construcción y desarrollo del proyecto.

6.1. CONCLUSIONES

Como se ha visto, el mundo del radiocontrol o de los vehículos controlados por radio a pequeña y gran escala, ha ido desarrollándose de forma exponencial hasta nuestros días.
En sus orígenes se partía de un coche conectado a su mando de control por un cable por el que pasaba la tensión de referencia que hacía mover los motores del vehículo pero, poco a poco, y gracias a la tenacidad y vocación de las personas interesadas en el desarrollo y avance tecnológico de las comunicaciones inalámbricas ya sea por hobby, trabajo, o por mera curiosidad de investigación, se pasó de la interacción entre módulos emisor y receptor a través de ese cable a poder controlar esta comunicación incluso desde nuestro teléfono móvil.

En nuestro caso, se ha utilizado la radiofrecuencia, una banda de frecuencia común y muy práctica que se ha utilizado increíblemente a lo largo de los años para multitud de fines y con diversas variantes.
La innovación ha sido la utilización de dicha banda de radio como comunicación y transferencia de datos desde un ordenador, establecido con funciones de valores de las coordenadas de un joystick de control, hasta la plataforma receptora, el módulo del vehículo, haciendo que éste responda de manera óptima a cada una de las señales enviadas, interpretaras todas por el código cargado perfectamente en el Arduino.

Pues bien, decir que, en este proyecto, han sido necesarios la gran parte de conocimientos aprendidos a lo largo de toda la formación universitaria, y más concretamente de electrónica digital, analógica y mucha programación, de hecho, desde los comienzos del proyecto, ha sido importantísima la documentación de manera que se ha debido profundizar bastante en aspectos concretos ampliando sobremanera la información y aprendizaje en ciertos ámbitos como los citados anteriormente.

Además, se han tenido que hacer frente a inconvenientes tanto físicos, temporales o de programación, que han retado la capacidad de improvisación o salidas alternativas, las cuales se han visto igual de válidas y óptimas que la idea inicial, es decir, la competencia no se ha rebajado en ningún momento.

Aunque los resultados obtenidos al acabar definitivamente la realización del proyecto puedan reflejar la dificultad del diverso proceso de investigación llevado a cabo, posiblemente no se aproximen a mostrar el verdadero esfuerzo a lo largo de toda la elaboración del mismo ya que fueron numerosos los problemas y contratiempos que iban surgiendo que hacía, por momentos, que la frustración imperara, pero con constancia y sacrificio se fueron solucionando para alcanzar la meta final.

La comprobación real del funcionamiento del vehículo, en función de cómo se había programado, todo bajo la tutela de nuestra opinión, y de la correcta construcción de todos los elementos sobre el mismo, fue un momento increíble pues ,aunque es incuestionable el peso de la parte teórica en el proyecto, se sabe que la práctica no siempre se cumple y de qué manera esto es cierto, surgen obstáculos que la diferencian del aspecto teórico o incluso del primer prototipo de resultado final, por lo que el hecho de que el sistema desarrollado funcione y se pueda comprobar en la realidad con rendimiento óptimo, supone una importante valía a la hora de analizar los resultados obtenidos.

6.2. TRABAJOS FUTUROS

Con todo lo expuesto anteriormente, se podría decir que, partiendo del proyecto realizado y desarrollado en esta memoria, se podrían tomar muchas vías diferentes de desarrollo alternativo a partir del mismo, es decir, podría haber muchas variantes según las modificaciones, cambios o implementaciones aplicada al proyecto original aquí desarrollado.

Se ha realizado un proyecto que ha cubierto las expectativas de comunicación inalámbrica entre diferentes módulos programados, a distancia, con la utilización de transceptores nRF24L01.
El término transceptor fue acuñado a principios de la década de 1920; es un dispositivo que cuenta con un transmisor y un receptor que comparten parte de la circuitería o se encuentran dentro de la misma caja aunque, cuando el transmisor y el receptor no tienen en común partes del circuito electrónico se conoce como transmisor-receptor.

Dado que determinados elementos del circuito se utilizan tanto para la transmisión como para la recepción, la comunicación que provee un transceptor solo puede ser semidúplex, lo que significa que pueden enviarse señales en ambos sentidos, pero no simultáneamente.

Por otra parte, no todos los elementos señalados son estrictamente necesarios para emitir una onda de radio, ya que cualquier generador de corriente alterna conectado a un conductor (antena) radiará una señal por lo que, entre otras razones, para optimizar el rendimiento del dispositivo emisor se prefiere el empleo de determinadas frecuencias del espectro electromagnético denominadas radiofrecuencias.

Nuevas salidas o variantes de este presente proyecto podrían ser las que utilicen, en la comunicación inalámbrica, un shield Xbee con protocolo de comunicación IEEE 802.15.4 mejor conocido como ZigBee, comunicación bluetooth o wifi desde el ordenador o dispositivo móvil, o incluso con un módulo con protocolo de comunicación Ethernet programado pudiendo controlar el recepto incluso a distancias mucho más largas y con rangio más amplio que el visto hasta ahora ya que, podríamos conectarnos a cualquier lugar de la red y tener acceso a las entradas y salidas del módulo receptor.

6.3. PRESUPUESTO FINAL

El presupuesto relativo al material y herramientas empleadas para la implementación del proyecto al completo será presentado de forma individual para el módulo emisor y el módulo receptor.

6.3.1. Módulo emisor

REFERENCIA	PRECIO/UNIDAD(€)	Nº UNIDADES	COSTE(€)
Arduino Uno Rev 3	20	1	20
Jumpers M-H	0.20	7	1.40
Transceptor Nrf24L01	5.50	1	5.50
Conector USB	4.75	1	4.75

COSTE TOTAL(€)
31.65

6.3.2. Módulo receptor

REFERENCIA	PRECIO UNIDAD(€)	Nº UNIDADES	COSTE(€)
Arduino Uno Rev.3	20	1	20
Kit Car Arduino	17.80	1	17.80
Jumper M-H	0.20	21	4.20
Antena ABS	3.20	1	3.20
Transceptor Nrf24L01	5.50	1	5.50
Conector USB	4.75	1	4.75
Pilas alcalinas AA 1.5V	0.55	4	2.20
Pila 9V	3.80	1	3.80
Zócalo pilas 1.5V	2.10	1	2.10
Zócalo pila 9V	1.80	1	1.80
Adhesivo doble cara	5.00	1	5.00
Adhesivo líquido transparente	2.50	1	2.50
Interruptor alimentación	0.95	1	0.95
Conector Pila 9V	0.40	1	0.40
Conector alimentación DCM	1.25	1	1.25

COSTE TOTAL(€)
75.15

6.3.3. Presupuesto total

Módulo emisor	31.65 €
Módulo receptor	75.15 €

COSTE TOTAL
106.80 €

BIBLIOGRAFÍA Y REFERENCIAS

Listado de referencias bibliográficas:

[1] "Manual básico de Arduino", Jorge Pomares Baeza, 2009, Universidad de Alicante.

[2] "Manual de Programación Arduino", José Manuel Ruiz Gutiérrez, 2012.

[3] "Arduino Práctico (Manuales Imprescindibles)", Joan Ribas Lequerica, 2013.

[4] "Comunicación por RF entre microcontroladores PIC18 mediante el módulo NRF24L01", Pablo Sanz Fernández, 2010.

[5] "Programación del microcontrolador ATMega328P", Alberto J. Molina Cantero, 2013.

[6] "Invertir el giro de un motor de DC con el L293B o L293D", Pedro Sánchez Ramírez, 2010.

[7] "Automatización de un sistema de riego por goteo mediante plataforma Arduino", Gorka Echarte []Vidaurre, 2012, Universidad de Navarra.

Listado de referencia a artículos:

[8] James Bruce, "What Is Arduino & What Can You Do With It? [Technology Explained]", 2011.

[9] Jeroen Doggen, "Arduino-based Wireless Sensor Network", Ambient 2012, Barcelona.

[10] Ibrahim Develi Volkan Onursoy, "High-quality image transmission over 2.4 GHz short-range wireless communication system", Information Science and Engineering (ICISE), 2009 1st International Conference.

Listado de referencias a direcciones URL:

[11] Página web de Panamahitek con información del registro Port.
http://panamahitek.com/registro-port-puerto/

[12] Página web de Bipedolandia
http://www.bipedolandia.es/t1685-diferencia-entre-servo-analogico-y-digital

[13] Página web de Electrónica y ciencia.
http://electronicayciencia.blogspot.com.es/2010/09/receptor-coche-rc-de-dos-_canales.html

[14] Página web de Ardumanía:
http://www.ardumania.es/apendice-del-ejercicio-3-senales-de-control-rc/

[15] Página web de Automodellismo
http://www.automodelismo.com/partespo.htm

[16] Página web de Wikipedia.
http://es.wikipedia.org/wiki/Servomotor_de_modelismo

[17] Página web de uControl.
http://www.ucontrol.com.ar

[18] Página web de Cochesrc.
http://www.cochesrc.com/foros/1-8-tt-gas/1793071-marca-espanola.html

[19] Página web de Cooking Hacks con información del Nrf24L01:
http://www.cooking-hacks.com/transceiver-nrf24l01-module-with-rp-sma?___store=es&___from_store=en

[20] Página web de Bricogeek.
http://blog.bricogeek.com/noticias/tutoriales/tutorial-robot-4x4-con-arduino/

[21] Página web de Robotic-controls.
http://robotic-controls.com/book/export/html/51

[22] Página web de Trasjeant.
http://www.trastejant.es/circuitos/servoJostickArduino.html