eman ta zabal zazu
Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACION DE COCHE TELEDIRIGIDO MEDIANTE ARDUINO

Alumno/Alumna: <Peña,Uriarte,Dabi>

Director/Directora (1): <Oleagordia,Aguirre,Iñigo Javier>

Curso:<2018-2019>

Fecha:11/02/2019

- RESUMEN CASTELLANO

Como objetivo principal se presenta el diseño e implementación de un coche teledirigido por medio de una aplicación móvil, empleando un módulo de tecnología bluetooth para llevar a cabo la comunicación inalámbrica.

Como cerebro del automóvil se configura un microcontrolador cuya función será la de procesar la información que se le envíe, de forma que pueda enviar las órdenes pertinentes a los componentes hardware de control, encargados del correcto funcionamiento del vehículo.

Además del control directo del coche se implementan los componentes necesarios (sensor ultrasónico y servomotor) para poder desarrollar una función de conducción automática capaz de sortear los objetos que se encuentre en su trayectoria.

Desde el menú de control del teléfono móvil se podrá seleccionar el modo de operación correspondiente al control directo del automóvil y el correspondiente a la función esquiva obstáculos, además se tendrá el control de las luces de posición incorporadas.

- RESUMEN INGLES

The main goal of the project is to design and build a remote control car which can be controlled with a smartphone, using a bluetooth module in order to have a wireless connection. A microcontroller is used as the car's CPU and has the mission of processing all the information and data that will be sent from the smartphone to transform it into the commands that will be transferred to the hardware components.

Apart from the direct control of the car, necessary components are included (ultrasonic sensor and servomotor) to develop an automatic pilot function, so that it can detect objects in front of it and sidestep them. From a smartphone control menu both function modes are available: the direct control mode and the automatic pilot function mode. Furthermore, there is another option in this menu that provides the possibility to turn the car's lights ON or OFF.

- RESUMEN EUSKERA

Proiektuaren helburu nagusi gisatelefono mugikor baten aplikazio baten bitartez telegidatutako auto baten diseinua eta inplementazioa aurkezten da; horretarako, bluetooth teknologiako modulu baterabiltzen da, haririk gabeko konexioa bermatzeko.

Autoaren burmuin gisa mikrokontroladore bat konfiguratzen da; mikrokontroladore horrek igortzen zaion informazioa prozesatuko du, kontrol-hardware osagaiei −autoaren funtzionamendu egokiaren arduradun−, agindu egokiak bidali ahal izateko,Autoaren zuzeneko kontrolaz gain, haren ibilbidean zehar aurki ditzakeen objektuak saihesteko beharrezko osagaiak ere inplementatzen dira (ultrasoinu-sentsoreak eta serbomotoreak).

Telefono mugikorraren kontrolaren menutikbi funtzionamendu-modu aukeratu ahal izango dira: autoaren zuzeneko kontrolaren funtzionamendua eta oztopo-saiheste modua; horretaz gain, txertatutako posizio-argien kontrola ere izango da.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

INDICE DE DIAGRAMAS

1. INTRODUCCION

-Como introducción se presenta una pequeña vista general del proyecto, se exponen los objetivos a alcanzar y se analiza la estructura escogida para realizar la memoria de este.
En este proyecto se estudia el diseño, a nivel teórico, y la implementación, a nivel práctico, de un automóvil controlado de manera remota mediante un smartphone o dispositivo móvil. Para la implementación física de este se emplea el hardware necesario para construir la estructura del vehículo (chasis, llantas...) y los componentes de hardware de control necesarios (motores, puente H, módulo bluetooth...). Esto se puede reunir todo en una misma sección "hardware", mientras que por otro lado, se encuentra toda la parte correspondiente al "software" donde se analiza la programación, diagramas de flujo, simulaciones etc...
El vehículo estará controlado mediante la acción de un microcontrolador que se encargará de enviar los comandos pertinentes para llevar a cabo el control de este.

1.1.OBJETIVOS A ALCANZAR

-El objetivo principal del proyecto es la implementación y control de manera directa de un automóvil a escala desde un dispositivo móvil. El microcontrolador será el encargado de recibir y enviar las órdenes de control a los componentes, actuando como centro de control del automóvil. Para implementar el envío y recepción de datos se hace uso un módulo bluetooth como intermediario entre el usuario y el controlador.
El proyecto se puede reducir a la siguiente lista de objetivos:

- Conocimiento y control de los componentes hardware implementados.
- Entender y conocer a fondo la funcionalidad del microcontrolador Arduino, estudiando las ventajas y desventajas que ofrece con respecto a otros posibles controladores.
- Control directo del vehículo desde nuestro smartphone mediante tecnología bluetooth.
- Funciones extra del automóvil:
  -Modo automático esquiva obstáculos, mediante sensor de ultrasonidos.
  -Control de luces de posición.

1.2. ESTRUCTURA Y FASES DE LA MEMORIA DEL PROYECTO

-En cuanto a la estructuración de la memoria del proyecto, se comienza con una introducción resumiendo de lo que trata el proyecto y analizando los distintos objetivos de este. Una vez introducido se analiza el bloque de diseño, donde por una parte se estudia la arquitectura hardware empleada (sistema de procesamiento, componentes de control, cálculos...) y por la otra parte se centra también en el software necesario (entorno de programación, tipo de comunicación, simulaciones...).
El bloque de diseño abarca la mayor parte del contenido de la memoria, por lo que una vez estudiado se procede a analizar los resultados obtenidos, así como los problemas encontrados durante la realización de todo el proyecto y por último una sección de conclusiones a nivel personal.

2. DISEÑO

2.1. ESTUDIO DE LA ARQUITECTURA HARDWARE

-Cuando se habla del "hardware" se hace referencia a cualquier componente físico que forma parte del sistema que interactúa sirviendo como interfaz entre el usuario que envía las señales de control, y el programa (software) que ejecuta los comandos.
Se puede definir también como **[1]** *"conjunto de los componentes que integran la parte material de una computadora"*.
En esta sección se realiza un estudio de los componentes hardware necesarios y se analiza la funcionalidad de cada uno por separado. Primero se comienza por el montaje de la estructura del automóvil mediante el "kit robot-car 4WD". Una vez analizado el chasis del vehículo se procede al sistema de procesamiento, donde se expone el proceso de selección del controlador, la valoración de las características y las posibles ventajas y desventajas de este.
Analizada la estructura y el sistema de procesamiento se desglosa los distintos componentes y la función que desempeña cada uno en el proyecto.
Por último, se expone la sección de cálculos donde se estudia si las especificaciones del regulador de tensión (LM317) empleado se ajustan a las necesidades del proyecto, valorando la posibilidad de usar un radiador para facilitar la disipación de potencia.

2.1.1. IMPLEMENTACIÓN DE LA ESTRUCTURA DEL AUTOMÓVIL

-Para este apartado se exponen todos los materiales necesarios para la implementación a de la estructura del chasis del automóvil.

- KIT ROBOT-CAR ARDUINO 4WD

En una primera instancia, se valora la posibilidad de llevar a cabo el proyecto sin hacer uso de un kit comercial. Se estudia la opción de comprar motores y ruedas por separado, llevando a cabo a mano la implementación del chasis. Finalmente se llega a la conclusión de que en ambos casos el resultado iba a ser similar pero con una peor presentación, por este motivo se escoge hacer uso del "kit robot-car 4WD".

Figura 1: Kit robot-car Arduino 4WD

Este kit resulta en una mayor comodidad a la hora de colocar todos los componentes mediante una distribución que aproveche el espacio útil y además permita la manipulación de dichos componentes.

A continuación se expone una breve lista con los componentes del kit. En la sección de "*Anexos III"* se analizan las características y dimensiones de cada uno de estos componentes por separado.

Componentes del kit:

1. 2 x chasis del automóvil inteligente (metacrilato)
2. 4 x Ruedas de dirección
3. 1 x compartimento de las pilas 2 (18650 Batería de litio recargable 3,7 V)
4. 4 x encoders de velocidad
5. 4 x Motorreductores o motores DC 3 ~ 12V (120RPM con tensión de 3V)
6. Tornillería

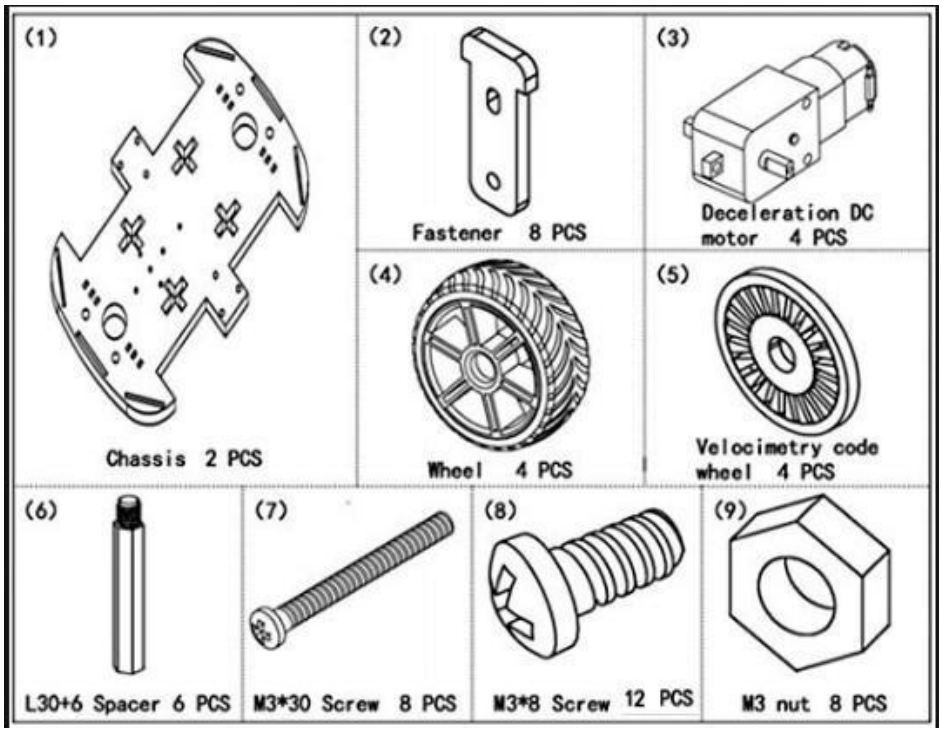


Figura 2: Componentes del Kit robot-car Arduino 4WD

2.1.2. ANÁLISIS DEL SISTEMA DE PROCESAMIENTO

2.1.2.1. ELECCIÓN DEL TIPO DE CONTROLADOR

-Para analizar el tipo de controlador que se ajuste más cómoda y eficientemente a este proyecto primero se procede a analizar las diferentes opciones de las que se disponen.

[2] Los **microprocesadores** se han desarrollado fundamentalmente orientados para el mercado de los ordenadores personales y las estaciones de trabajo (requerimientos de potencia y mucha memoria). Por otra parte, los **microcontroladores** están concebidos para ser utilizados en otro tipo de aplicaciones puntuales, es decir, aplicaciones donde el microcontrolador debe realizar un pequeño número de tareas, al menor costo posible.

Los **microcontroladores** son computadoras pequeñas que realizan tareas específicas. La diferencia principal entre un microcontrolador y una computadora es cuestión de escala. Usualmente un microcontrolador está programado para una tarea específica y suele hacerla sin intervención de nadie. Por el contrario, una computadora puede llevar a cabo una amplia variedad de trabajos.

Cabe destacar que los microcontroladores como también son computadoras, tendrán microprocesadores como parte de su sistema de hardware.

Un claro ejemplo de estas dos definiciones podría ser **Arduino**, basado en un microcontrolador y una **Raspberry Pi** se basada en un microprocesador.

Figura 3: Arduino

(µC)Figura 4: Rasberry Pi (µP)

A continuación se exponen las características propias junto con las ventajas y desventajas de cada uno de estos dos sistemas de procesamiento en la siguiente tabla:

	Microprocesadores (µP)	**Microcontroladores(µC)**
CPU	Mucha más potencia de cálculo, por lo cual solamente realiza sus funciones con los datos y su algoritmo o programa establecido.	Es una de sus partes principales, la cual se encarga de dirigir sus operaciones.
Memorias RAM y ROM	Son dispositivos externos que lo complementan para su óptimo funcionamiento	Incluidas en un solo circuito integrado
Velocidad de Operación	Bastante rápida	Relativamente lenta en comparación a µP
Interferencias	Más susceptibles a la interferencia electromagnética debido a su tamaño y a su cableado externo (propenso al ruido).	El alto nivel de integración reduce los niveles de interferencia electromagnética
Tiempo de desarrollo	El tiempo de desarrollo de un microprocesador es lento.	Por el contrario, el de un microcontrolador es rápido
Costos	Costo alto de producción.	Costo bajo de producción

Tabla 1: Microcontrolador / Microprocesador

-Tras analizar detenidamente las ventajas y desventajas de cada uno de estos sistemas de procesamiento se llega a la conclusión de que un microcontrolador (µC) es lo que más se ajusta a las necesidades de este proyecto.

En un proyecto grande lo ideal es usar ambos (µC y µP), cada uno en la tarea que más se le ajuste. Por ejemplo, la recolección de datos, supervisión del entorno, envío de alarmas, accionar motores…será responsabilidad para el Arduino, mientras que el tratamiento de los

datos recogidos, el interfaz gráfico de usuario, envío de correos, etc… será tarea para un ordenador o una Raspberry pi o similar.

2.1.2.2. MICROCONTROLADOR ARDUINO

-**Arduino** es una plataforma para prototipado de electrónica basada en hardware y software libre y sencillo de utilizar. Permite construir circuitos electrónicos y programarlos con esta placa. Existen varios modelos de sistemas Arduino que van cambiando de microcontrolador, siendo los primeros el **Atmega8** y el **Atmega168**. Se tratan de dos tipos de microcontroladores, que se diferencian en la capacidad de memoria interna de la que dispone cada uno para almacenar el programa. Tanto su diseño como su distribución son libres (Open-Hardware), es decir, puede utilizarse sin inconvenientes para desarrollar cualquier tipo de proyecto sin tener que adquirir ningún tipo de licencia.

Ventajas de este microcontrolador elegido:
Entradas y salidas disponibles. Dependiendo de las necesidades del proyecto, podremos elegir entre las distintas placas, que cuentan con multitud de entradas y salidas digitales, entradas analógicas, así como puertos de comunicaciones.
-**Sistema muy didáctico**. Gracias a su configuración de hardware y la simplicidad del lenguaje, unido a la cantidad de información y ejemplos existentes, es posible empezar con pequeños proyectos de forma rápida y segura.
-**Amplia gama de versiones y accesorios compatibles**
-**Precio**. Tenemos variedad de multitud de modelos de placas originales, todos ellos de un muy bajo coste, además de existir versiones de otros fabricantes.
-**Muy extendido y estandarizado**. Existen infinidad de librerías de libre distribución para poder comunicarse con hardware y software de terceros.

Entre los distintos modelos de este tipo de microcontrolador se exponen brevemente los más empleados y las especificaciones técnicas, incluidas en la sección de anexos (Anexos I: Microcontrolador Arduino)

- **Arduino Uno** (ATmega328, además de 14 pines de entrada/salida)

Figura 5: Arduino Uno

- **Arduino Mega** (ATmeg1280 (datasen). Tiene 54 entradas/salidas digitales)

Figura 6: Arduino Mega

- **Arduino Leonardo** (Atmega32u4 y producido por Atmel. Dispone de 20 pines digitales de entrada/salida)

Figura 7: Arduino Leonardo

Estos son los tipos de Arduino más utilizados para proyectos, sin embargo existe todavía más variedad de modelos menos utilizados actualmente:
-Arduino Ethernet.
-Arduino Mini.
-Arduino Duemilanove.
-Arduino Diecimila
Existen muchos más modelos distintos de Arduino, pero se ha considerado oportuno realizar el análisis de todos y cada uno de ellos.
Para la implementación del proyecto la idea original era utilizar el **Arduino Uno**. Sin embargo, una vez montados los componentes este modelo concreto, se aprecia que requiere de una gran superficie del espacio disponible, por esta razón se selecciona finalmente el **Arduino Nano** para emplear en el proyecto puesto que se ajusta mejor a las necesidades de este.

Las características del modelo de Arduino escogido son las siguientes:

Figura 8: Arduino Nano

Placa compacta diseñada para usar directamente en placas de desarrollo, no dispone de conector para alimentación externa y funciona mediante un cable USB MiniB en vez del cable estándar, tipo B. Basado en el ATmega328 (Arduino Nano 3.0, que es el empleado para el proyecto) o ATmega168 (Arduino Nano 2.x) que se usa conectándola a una protoboard. Es la nueva generación de placas que permite un rápido 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 conector externamente.

2.1.2 COMPONENTES HARDWARE DE CONTROL

Los componentes de control empleados para llevar a cabo y ejecutar las órdenes que se envíen desde nuestro microcontrolador Arduino son los siguientes:

1. Puente en H.
2. Módulo Bluetooth.
3. Fuente de tensión regulada.
4. Servomotor.
5. Sensor ultrasónico.

En este apartado se analizan cada uno de estos y su funcionalidad, con el fin de poder realizar pruebas y simulaciones con cada uno de estos componentes, antes de exponer el código final correspondiente, en el apartado de “Estudio de la arquitectura software” en la sección de “Programación: códigos Arduino” y en “Simulaciones”.

1. PUENTE H

1.1. INTRODUCCIÓN

-El puente en H es un circuito electrónico que permite a los motores eléctricos ser activados, en un sentido u otro y al mismo tiempo poder controlar variables como, velocidad y torque de los mismos. (El torque se define como la capacidad de desarrollar fuerza sobre un eje, es decir, es la fuerza que ejerce sobre algo que gire o sobre algo para hacerlo girar).

Un puente H se construye con 4 interruptores (mecánicos o mediante el uso de transistores). En el momento en que los interruptores S1 y S4 están cerrados (S2 y S3 abiertos) se aplica una tensión haciendo girar el motor en un sentido.

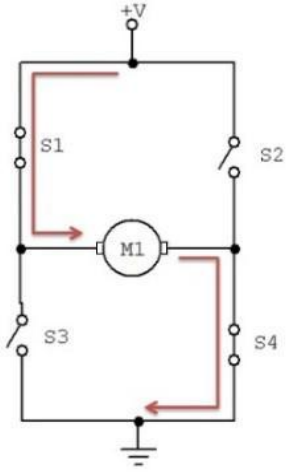


Figura 9: Puente H

En el caso contrario abriendo los interruptores S1 y S4 (cerrando S2 y S3), la polaridad del voltaje se invierte, permitiendo el giro en sentido inverso del motor.

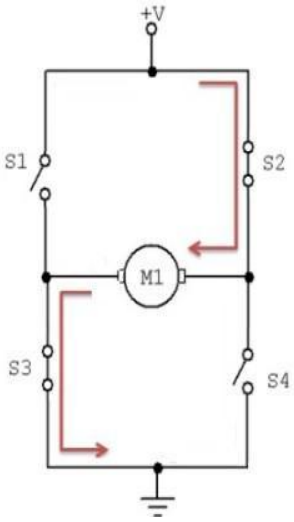


Figura 9: Puente H

A continuación se expone una tabla de la verdad explicativa de las diferentes situaciones que se pueden llevar a cabo mediante las distintas polarizaciones de los transistores (posición de los interruptores a nivel teórico)

S1	S2	S3	S4	**Resultado**
1	0	0	1	El motor gira en *avance*
0	1	1	0	El motor gira en *retroceso*
0	0	0	0	El motor se detiene bajo su inercia
1	0	1	0	El motor frena (*fast-stop*)
0	1	0	1	El motor frena (*fast-stop*)

Tabla 2: Tabla de la verdad del puente en H

Normalmente para el puente H se usan interruptores de estado sólido (transistores), puesto que sus tiempos de vida y frecuencias de conmutación son mucho más altos. Por ejemplo en los convertidores de potencia sería impensable usar interruptores mecánicos.

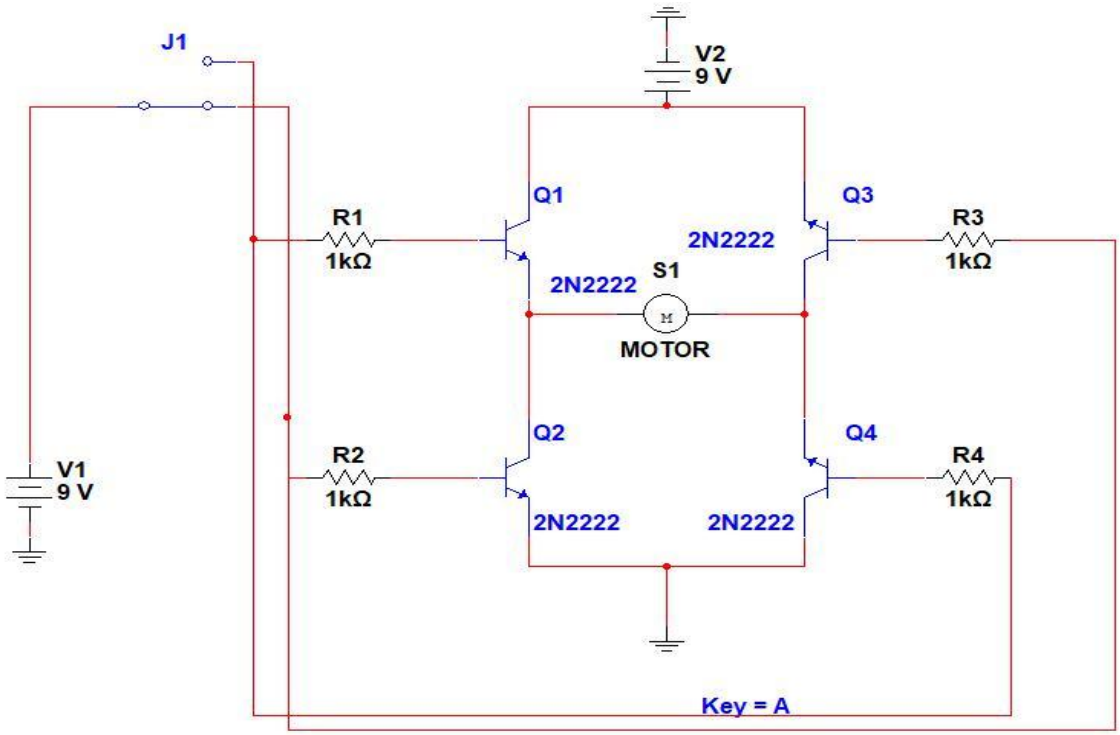


Figura 10: Circuito puente H

En el apartado de "Estudio de la arquitectura software: Simulaciones" se puede observar el funcionamiento del puente en H comprobando como fluyen las intensidades por medio de la simulación del circuito.

1.2. PUENTE H: Módulo L298N

-El módulo L298N será el componente real empleado para la implementación del control de los cuatro motores.

Este módulo dispone de dos canales de Puente H, pudiéndose utilizar para controlar dos motores DC o un motor pasó a paso, permitiendo variar el sentido de giro y velocidad.

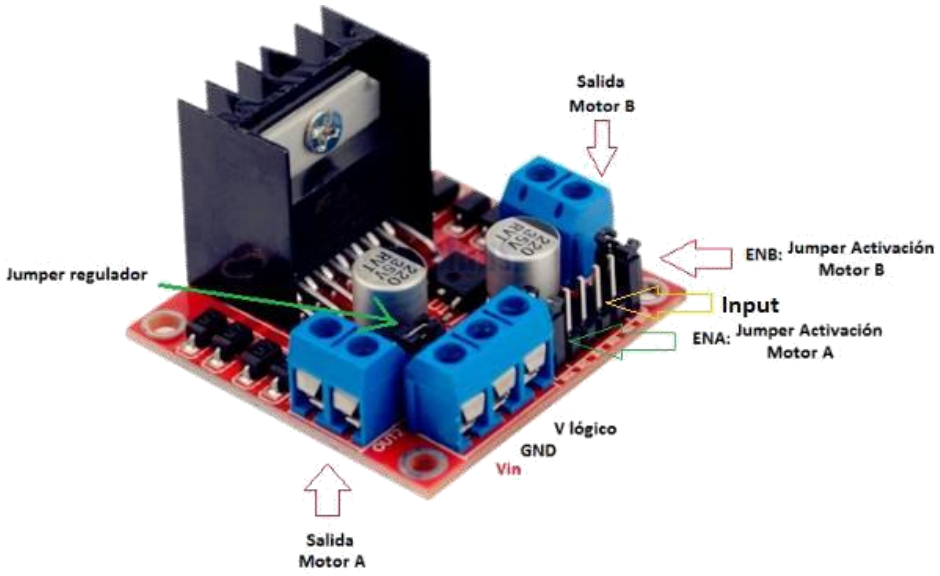


Figura 11: Módulo L298N

[3]Como ya se ha comentado el módulo posee dos pines de salida distintos para conectar a dos o más motores. A continuación se encuentra el pin **Vin** a través del cual se alimenta el módulo con una tensión de entre 6V-12V. Junto a **Vin** está el pin al que vincularemos la tierra (**GND**) y por último un pin **Vlógico** en el que se dispone de 5V siempre que esté el regulador de tensión funcionando (jumper regulador activo).

Cuenta con tres jumper distintos, que se podrán dejar activos o inactivos en función de las necesidades del proyecto.

El primer jumper está conectado al regulador de tensión que lleva incorporado el módulo. Si se remueve este jumper se desactiva la regulación de tensión interna del componente (la funcionalidad de este regulador se entenderá mejor a la hora de valorar las distintas formas de alimentar el L298N).

Entre los otros dos jumper (**ENA y ENB**), que están cada uno asociado a la salida de los motores, están los 4 pines de entrada **IN1, IN2** (motor A) **y IN3, IN4** (motor B) que sirven para el control de dichos motores. **ENA** y **ENB**, sirven para habilitar o deshabilitar sus respectivos motores, generalmente se utilizan para controlar la velocidad, ingresando una señal de PWM por estos pines. Si no se usan estos pines se deben de conectar los Jumper para que siempre estén habilitados.

1.3 ESQUEMA ELECTRONICO MODULO L298N

-El esquema electrónico es bastante sencillo. Se reduce al driver L298N junto con sus diodos de protección y un regulador **LM7805** que suministra 5V a la parte lógica (**Vlógico**) del integrado L298N.

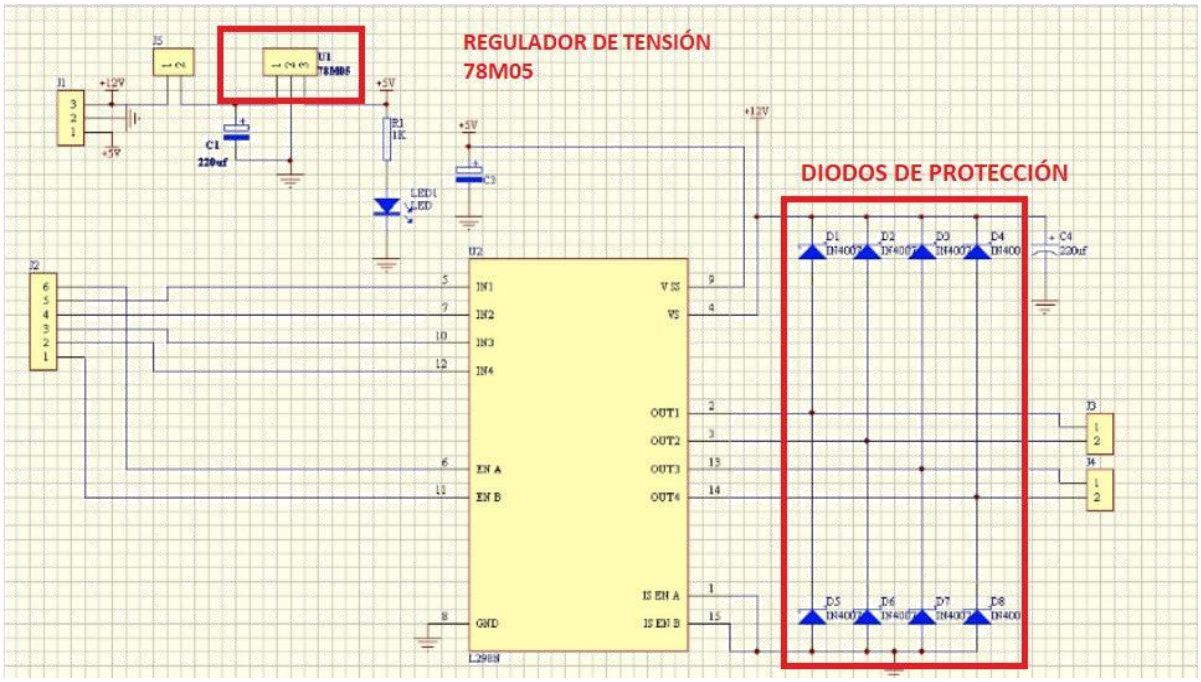


Figura 12: Esquema del módulo L298N

1.3 FORMAS DE ALIMENTACION

-La primera manera es utilizando una sola fuente, conectada a la entrada (Vin) de 12V y con el Jumper activo para que actúe el regulador tensión. Como resultado el voltaje de la fuente de alimentación será el que reciba el motor. De esta forma la entrada de 5V (Vlogic) no debe estar conectada a ninguna fuente de tensión, ya que en este pin están presentes 5V a través del regulador interno (78M05). Este pin nos queda libre pudiéndose utilizar como una salida de 5V, pero sin exceder los 500mA de consumo. Se recomienda hacer esta conexión para voltajes menores de 12V para no sobrecalentar el regulador.

-La segunda forma se realiza utilizando dos fuentes, una de 5V conectada a la entrada de 5V (Vlogic) y otra fuente de voltaje con el valor de tensión que trabaja el motor, conectada al pin de 12V. Para esto tenemos que desconectar el Jumper lo que desactivará al regulador (78M05).

1.4 CONEXIONADO DE MÓDULO L298N

-Lo primero destacar que se alimentará el módulo de la primera forma mencionada. De esta manera solo se precisa de una sola fuente de tensión (dos pilas de 3,7V) y como se encuentra el regulador 78M05 activado, se dispone de 5V (sin exceder los 500mA) en el pin Vlogic que se podrán emplear para alimentar al microcontrolador Arduino o el sensor de ultrasonidos por ejemplo.

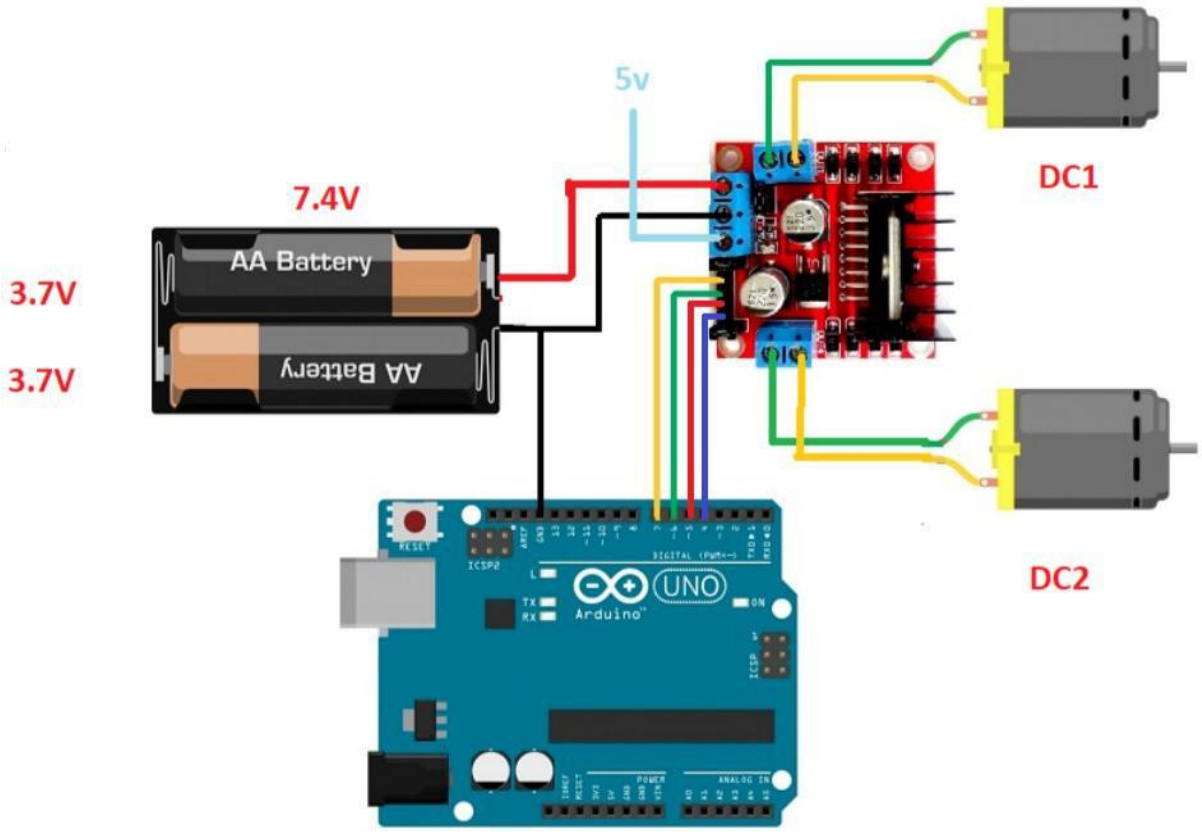


Figura 13: Conexionado módulo L298N

2. CONEXION INALAMBRICA: BLUETOOTH

2.1. INTRODUCCIÓN

-Bluetooth es una tecnología para redes inalámbricas de área personal (WPAN) que posibilita la transmisión de información entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz.

Los objetivos que se pretenden conseguir mediante esta tecnología son:

- •Facilitar las comunicaciones entre equipos móviles.
- •Eliminar la necesidad de cables y conectores entre estos.
- •Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Los aparatos que más utilizan esta tecnología pertenecen a sectores de las telecomunicaciones y la informática personal, como PDA, teléfonos móviles, computadoras portátiles, ordenadores personales, impresoras o cámaras digitales.

2.2. TIPOS DE BLUETOOTH

-Se trata de una tecnología de ondas de radio de corto alcance, 2.4GHz, que fue pensada para dispositivos de bajo consumo como dispositivos móviles, que requieren un corto alcance de emisión y basados en transceptores de bajo costo.

Clase	Potencia Máxima Permitida (mW)	Potencia Máxima Permitida (dBm)	Alcance (Aprox.)
Clase1	100 mW	20 dBm	100 m
Clase2	2.5 mW	4 dBm	10 m
Clase3	1 mW	0 dBm	1 m

Tabla 3: Tipos de bluetooth en función de la potencia

Existen diferentes versiones en función del ancho de banda del que se quiera disponer:

Versión	Ancho de banda
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
Versión 3.0 + HS	24 Mbit/s
Versión 4.0	24 Mbit/s

Tabla 4: Tipos de bluetooth en función del ancho de banda

2.3. MÓDULO BLUETOOTH: HC-06

-Entre los distintos módulos Bluetooth los más populares son HC-06 y el HC05.

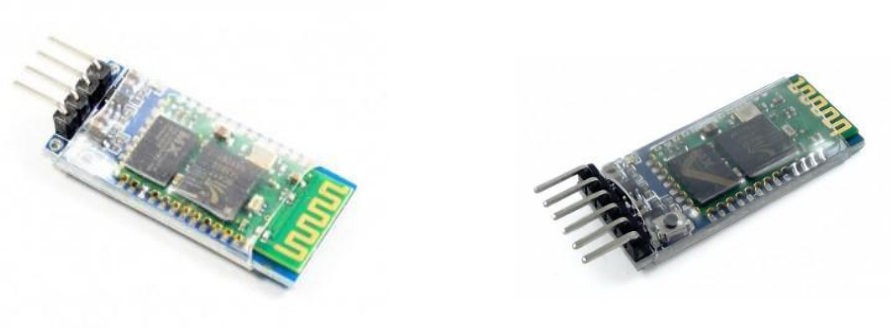


Figura 14: Módulo HC-06 y Módulo HC-05

El módulo HC-06 es muy similar con los demás módulos que existen en el mercado. Una simple diferencia es que el módulo HC-06 funciona como Slave y el HC -05 como Master y Slave.

HC- 06 se comporta como esclavo, esperando peticiones de conexión, si algún dispositivo se conecta el HC-06 transmite a este todos los datos que recibe del Arduino y viceversa. (Para este Proyecto trabajaremos un módulo HC06 pero también es válido para un módulo HC-05 en modo Esclavo)

Físicamente se diferencian por el número de pines. En el HC-06 tiene un conector de 4 pines mientras que el HC-05 trae uno de 6 pines

2.4. MODOS DE OPERACIÓN DEL MODULO HC-06

[4] Modo AT (Desconectado):

- Entra a este modo en cuanto se alimenta el modulo y cuando no se ha establecido una conexión bluetooth con ningún otro dispositivo.
- EL LED del módulo está parpadeando (frecuencia de parpadeo del LED es de 102ms).
- En este modo es cuando se debe enviar los comandos AT en caso se quiera configurar, si se envían otros datos diferentes a los comandos AT el HC-06 los ignorará.

Modo Conectado

- Entra a este modo cuando se establece una conexión con otro dispositivo bluetooth.
- El LED permanece encendido sin parpadear.
- Todos los datos que se ingresen al HC-06 por el Pin RX se trasmiten por bluetooth al dispositivo conectado, y los datos recibidos se devuelven por el pin TX. La comunicación es transparente.

-En este Modo el HC-06 no puede interpretar los comandos AT.

2.5 CONEXIONADO MÓDULO BLUETOOTH

Características módulo HC-06: 4 pines

Vcc: voltaje positivo de alimentación, se debe prestar especial atención puesto que existen módulos que solo soportan voltajes de 3.3V, pero en su mayoría ya vienen acondicionados para que trabajar en el rango de 3.3V a 6V, sin embargo, es recomendable revisar los datos técnicos del módulo escogido antes de hacer las conexiones.

-GND: voltaje negativo de alimentación, se conectará al GND correspondiente del Arduino o al GND de la placa que se esté usando.

-**TX**: pin de transmisión de datos, por este pin se transmiten los datos que le llegan desde un PC o un dispositivo móvil (como es en este caso). Este pin debe ir conectado al pin RX (recepción de datos) del Arduino.

-**RX**: pin de recepción, a través de este pin el HC -06 recibirá los datos del Arduino los cuales se transmitirán por Bluetooth, este pin va conectado al Pin TX del Arduino.

El conexionado del módulo junto con Arduino quedaría de la siguiente manera:

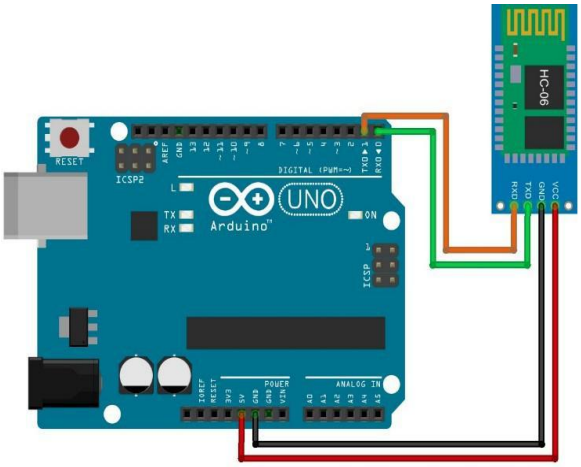


Figura 15: Conexionado módulo bluetooth: HC-06

Para cargar el programa al Arduino, se debe desconectar los pines RX0 y TX0 del microcontrolador, puesto que internamente el Arduino trabaja con los mismos pines para cargar el programa y si están conectados al módulo Bluetooth, no permitirá cargar (para evitar este inconveniente se puede usar el software serial y usar otros pines).

Más adelante en la sección de "estudio de la arquitectura software" se analiza el tipo de comunicación empleada y la manera de configurar este módulo para que sea funcional además del proceso de vinculación al dispositivo móvil.

3. REGULADOR DE TENSIÓN

3.1 INTRODUCCIÓN

-Una fuente regulada, es la que puede mantener un voltaje estable en su salida, a pesar de las variaciones del voltaje en la entrada y la carga a la que es expuesta.

¿Necesidad de un regulador de tensión?

Para este proyecto la alimentación de los motores no será un problema (2 pilas de 3,7V) sin embargo para alimentar el resto de componentes (servomotor, sensor de ultrasonido, módulo bluetooth y luces de posición) precisan de 5V constantes. Alimentar todos los componentes de una sola fuente de tensión de dos pilas de 3,7V es una opción muy poco eficiente. Los cuatro motores DC para trabajar a pleno rendimiento requieren de una alta demanda de corriente. De esta misma manera los servomotores por lo general precisan de una fuente de alimentación externa a la de Arduino debido a que la corriente que demanda este tipo de motor es mayor que la que puede proporcionar el microcontrolador.

Por todo ello se emplea una fuente de alimentación para Arduino y otra distinta el resto de componentes. Arduino puede alimentarse directamente con una fuente de 9V, sin embargo para el resto de componentes se necesitará 5V constantes. Para conseguir estos 5V se ha expone a nivel teórico y práctico como podrían obtenerse de una batería de 9V mediante el uso del **LM317.**

3.2 FUENTE DE TENSIÓN REGULADA: LM317

-El LM317 se trata de un regulador de tensión lineal ajustable capaz de suministrar en su salida un rango que va desde 1,2 hasta 37 Voltios y una intensidad de 1,5 A. El posible complemento al LM317 pero en tensión negativa es el circuito integrado LM337. El LM317 es uno de los primeros reguladores ajustables de la historia; el primero que existió fue el más conocido.

Las patillas son: Entrada (IN), Salida (OUT), Ajuste (ADJ).

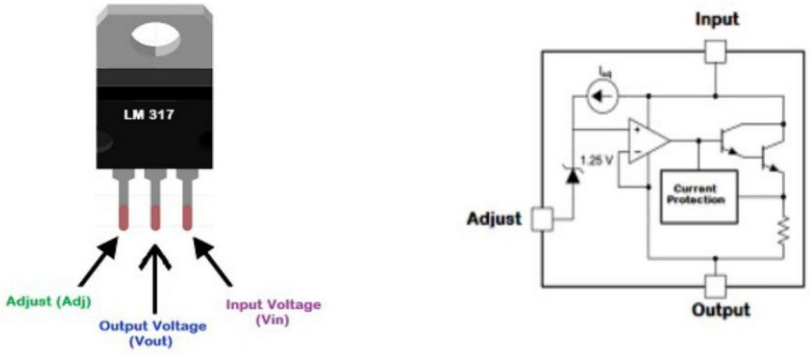


Figura 16: LM317

En la sección de "Anexos II" se disponen las especificaciones técnicas del componente. Por medio de estas se contrastará que el componente se adecua a nuestros parámetros de

operación del proyecto. También serán útiles de cara a valorar el posible uso de un radiador en el caso de requerir una disipación de potencia más eficiente.

3.3 CIRCUITO REGULADOR DE TENSIÓN

[5]Para esta fuente se ha utilizado el regulador variable **LM317**. Muy sencillo de montar y precisa de unos pocos componentes para acompañarlo. Solo requiere un potenciómetro y una resistencia, para ajustar el voltaje de salida, y en este caso un transistor, para aumentar el manejo de corriente.

La línea de carga y la regulación es de mejor calidad que la de los reguladores fijos, por ser un integrado más moderno. Está protegido contra las limitaciones de corriente, exceso de temperatura y por sobrecarga. A este tipo de regulador se las llama flotante. Quiere decir que el regulador solo ve la diferencial entre la entrada y la salida del voltaje. Esto permite utilizarlos para regular alto voltaje, siempre y cuando no supere más de 30 voltios la diferencia entre entrada/salida.

El voltaje de salida depende de la posición que tenga la patilla variable del potenciómetro, patilla que se conecta a la terminal de AJUSTE del integrado.

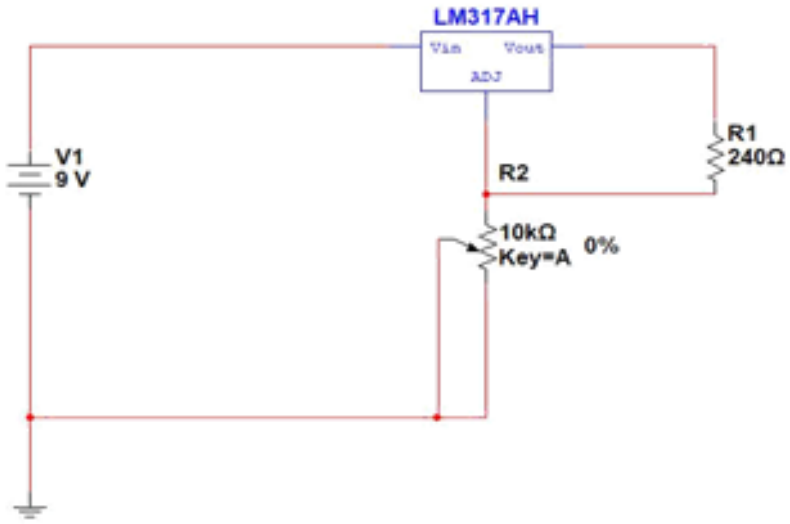


Figura 17: Regulador de tensión con salida variable

3.4 OPTIMIZACIÓN DEL FUNCIONAMIENTO DEL CIRCUITO REGULADOR DE VOLTAJE

-Con el propósito de optimizar el funcionamiento del regulador se pueden incorporar al diseño algunos elementos adicionales.

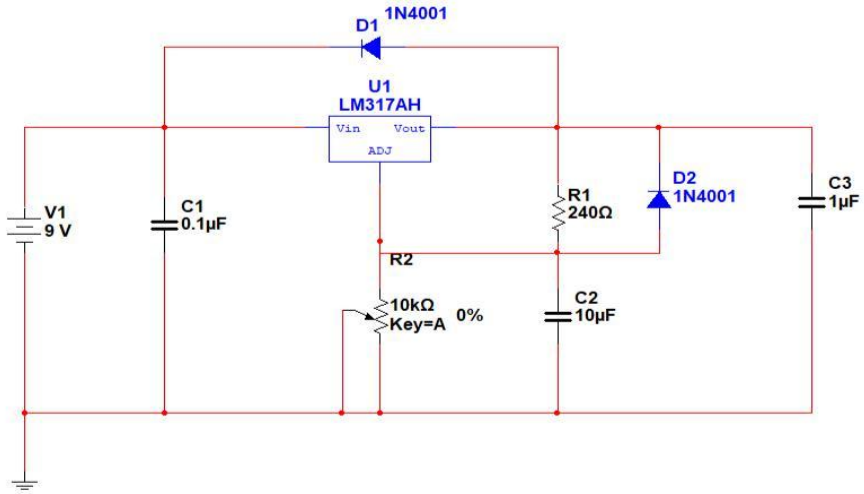


Figura 18: Regulador de tensión con salida variable con diodos de protección

- Condensador electrolítico C1 de 0.1uF en la patilla de entrada (IN) si el regulador se encuentra alejado del bloque que se encarga de la rectificación.
- Condensador electrolítico C3 de 25 uF en la patilla de salida (OUT) con el propósito de mejorar la respuesta a transitorios.
- Condensador electrolítico de 10 uF (C2) en paralelo con R2 con el propósito de mejorar el rechazo del rizado.
- Diodo D1 (1N4001) para proteger el regulador contra posibles cortos circuitos en la entrada del regulador.
- Diodo D2 (1N4001) para proteger al regulador contra posibles cortos circuitos en la salida al dar camino a la descarga de capacitores.

3.6 PRECAUCIONES A CONSIDERAR EN EL USO DEL LM317

-Las precauciones a considerar en el uso del LM317 están relacionadas principalmente en la disipación del calor. Toda energía disipada en este circuito se convierte en calor y este resulta en un aumento de la temperatura de los componentes, lo que puede generar el funcionamiento incorrecto o fallo de los mismos, por esto es muy importante tener controlada la disipación de energía y las temperaturas que se pueden alcanzar. Por ejemplo, si se tiene una tensión de entrada de 30v y queremos una salida de 5v para alimentar una carga que consume una corriente I=1A, entonces la potencia disipada será:

$$P = (Vin – Vout)*I = (30\ v – 5v) * 1A = 25w$$

Por tanto en el ejemplo anterior se convierte 25w de energía eléctrica en calor y se deberá prestar especial atención a toda esa energía disipada.

En la aplicación práctica para el proyecto (manteniendo el supuesto de que la corriente que alimente a la carga es de 1 A) se pasa de los 9V que aporta la batería a los 5V que se necesitan para alimentar el Arduino, por lo que la energía disipada quedaría:

$$P = (Vin – Vout)*I = (9\ v – 5v) * 1A = 4w$$

En la sección de cálculos se lleva a cabo un estudio sobre la necesidad real de un radiador para el proyecto, hallando la máxima potencia disipable por el componente sin radiador y comparándola con la que se necesita disipar para saber si será mayor o menor.

Como base previa a este estudio se expone en “Anexos II” una breve introducción sobre el análisis de la disipación de calor en general, para poder entender mejor los cálculos y el procedimiento seguido para el estudio de la disipación de calor.

En el apartado de “simulaciones” se analizará y comprobará el funcionamiento del circuito regulador de tensión.

4. SERVOMOTOR

4.1 INTRODUCCIÓN

-Los servomotores son dispositivos de accionamiento para el control de precisión de velocidad, posición y par motor. Se trata de un dispositivo electromecánico que consiste en un motor eléctrico, un juego de engranes y una tarjeta de control, todo dentro de una carcasa de plástico.

Establece una relación y convierte el movimiento mecánico (giros del eje) en pulsos digitales interpretados por un controlador de movimiento. También utilizan un driver, que en conjunto forman un circuito para comandar posición, torque y velocidad.

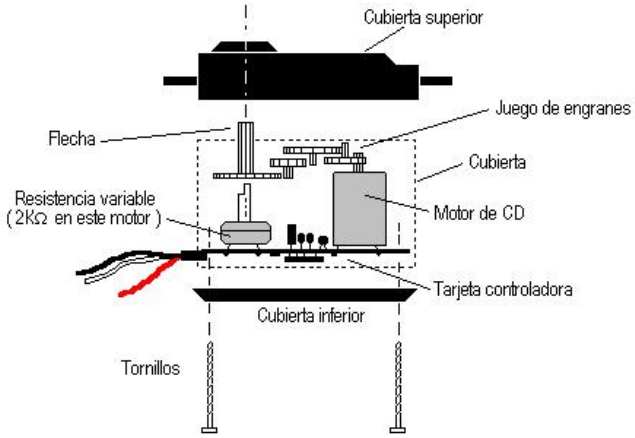


Figura 19: Despiece servomotor

Figura 20: Servomotor

4.2 FUNCIONAMIENTO: CONTROL PWM

-Uno de los sistemas más empleados para el control de los servos es la modulación por anchura de pulso, PWM (*Pulse WundtModulation*). Este sistema consiste en generar una onda cuadrada en la que se varía el tiempo que el pulso está a nivel alto, es decir el ciclo de trabajo, manteniendo el mismo período (normalmente), con el objetivo de modificar la posición del servo según se desee.

Todos los servos disponen de tres cables, dos para alimentación Vcc y Gnd (4.8 a 6 V) y un tercero para aplicar el tren de pulsos de control (señal PWM), que hace que el circuito de control diferencial interno modifique la posición del servo según los comandos que se introduzcan.

Los servos tienen sus márgenes de operación predeterminados, que se corresponden con el ancho del pulso máximo y mínimo con los que pueden trabajar. Los valores límite se corresponden con pulsos de entre 1 ms y 2 ms de anchura, que dejarían al motor en ambos extremos (0º y 180º). El valor 1.5 ms indicaría la posición central o neutra (90º), mientras que otros posibles valores de pulso lo resulten en posiciones intermedias.

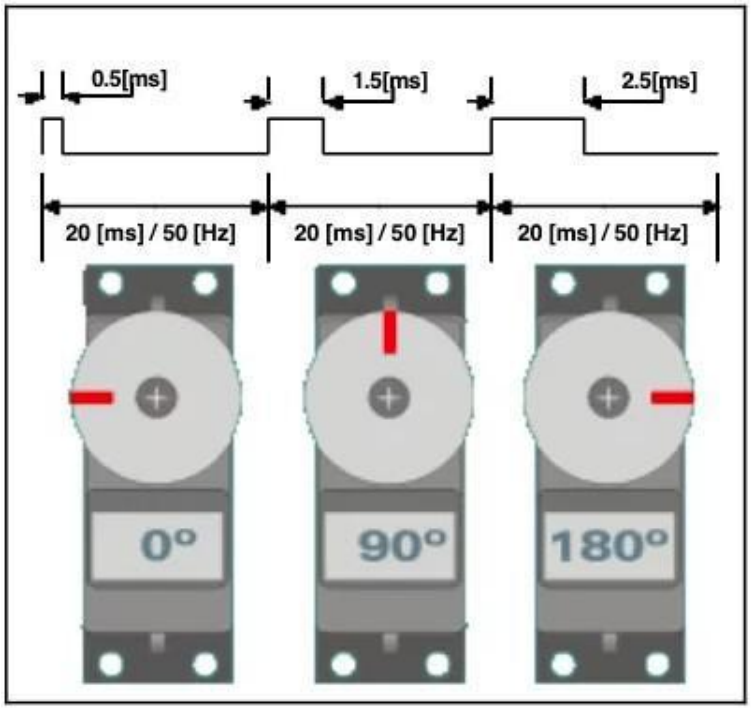


Figura 21: Relación entre el ciclo de trabajo y la posición del servomotor

Estos se tratan de los valores recomendados por los fabricantes, sin embargo, es posible emplear pulsos menores de 1 ms o mayores de 2 ms, pudiéndose alcanzar ángulos mayores de 180°. Si se sobrepasan los límites de movimiento del servo, éste comenzará a emitir un zumbido a modo de señal, indicando que se debe modificar la longitud del pulso introducido. Lo único que limita el recorrido es el tope del potenciómetro y los límites mecánicos físicos del componente en sí.

4.3 APLICACIONES MAS COMUNES

-El mundo del servomotor está muy presente en la rama industrial, tienen una gran cantidad de usos tales como: robótica, en las impresoras (control de avance y retroceso del papel), zoom de una cámara de fotos, las puertas de un ascensor, coches de radiocontrol, máquinas herramientas, sistemas de producción, en el timón de los aviones etc...

En sistemas de seguimiento solar se emplean para controlar el movimiento de los paneles solares en dirección del Sol. En máquinas como tornos, fresadoras, máquinas de troquelado (máquinas que se utilizan en la industria para hacer cortes) se utilizan los servomotores para controlar los cortes y poder hacerlos con la precisión adecuada. En este proyecto desempeñará la función de controlar hacia donde apunta el sensor de ultrasonidos a la hora de tomar mediciones.

4.4 SERVO MOTOR SG90

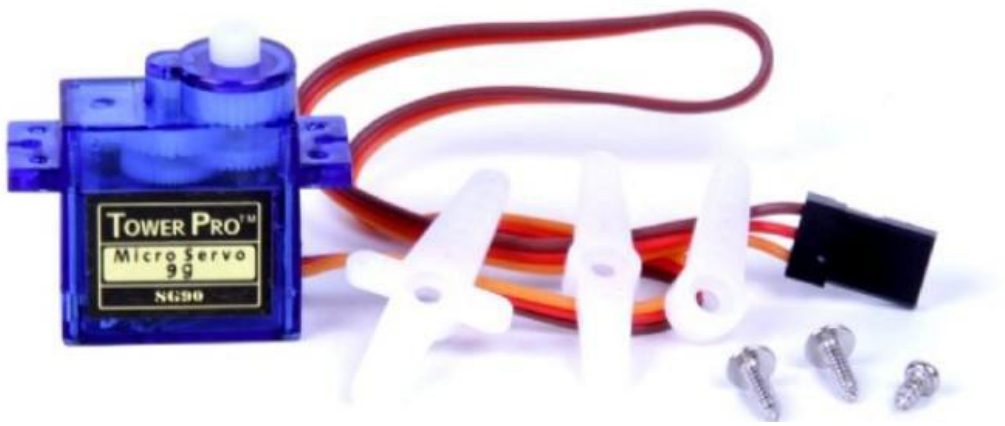


Figura 22: Servo Motor sG90

-Se ha escogido este componente debido a ser ideal para proyectos de bajo torque y donde se requiera poco peso. Muy usado en aeromodelismo, pequeños brazos robóticos y mini artrópodos. Su funcionalidad es compatible con la mayoría de tarjetas electrónicas de control, microcontroladores y también con la mayoría de los sistemas de radio control comerciales. Especialmente recomendado en aeronaves de aeromodelismo debido a sus características de torque, tamaño y peso.

4.4.1 CARACTERISTICAS Y DIMENSIONES

Las características y especificaciones del servo SG90 son las siguientes:

•Velocidad: 0.10 sec/60° (operando a 4.8V)

•Torque: 1.8 Kg-cm (operando a 4.8V)

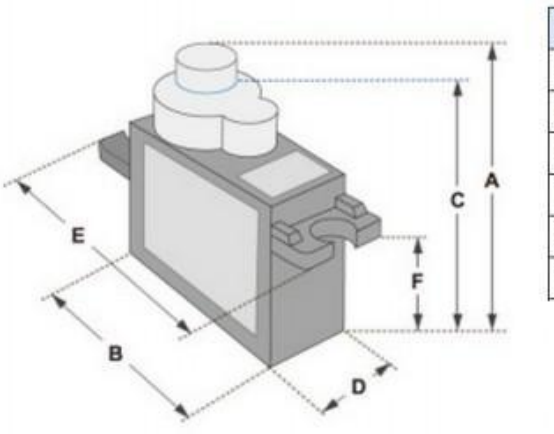
•Voltaje de funcionamiento: 4.8 V (~5V)

•Temperatura de funcionamiento: 0 ºC – 55 ºC
•Ángulo de rotación: 180°

•Ancho de pulso: 500-2400 µs

•Longitud de cable de conector: 24.5cm

Las dimensiones del componente son:



Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5

Figura 23: Dimensiones servomotor

4.4.2 CONEXIONADO SERVOMOTOR

[6]Como ya ha sido mencionado anteriormente los servos disponen de tres cables de conexión; uno para la alimentación, otro para masa y el tercero para introducir la señal PWM que controla el motor.

Figura 24: Conexionado servomotor SG90

En la mayoría de los casos la alimentación de los servos se realizará desde una fuente de tensión externa (una batería o fuente de alimentación) a una tensión de 5V-6.5V.

Esto se debe a la cantidad de corriente que puede necesitar el servomotor para su correcta alimentación. Arduino puede llegar a proporcionar suficiente corriente para encender un servo pequeño (SG90 en nuestro caso) para llevar a cabo unos cuantos proyectos de prueba. Sin embargo, no dispone de corriente suficiente para alimentar uno grande (MG996R). Incluso en el caso de alimentar varios servos pequeños, o hacer excesiva fuerza con ellos puede exceder la capacidad de corriente de Arduino, dañándolo o provocando su reinicio.

Por esta razón no es lo más adecuado emplear el microcontrolador para alimentar a este servomotor, de modo que se habilitará una fuente de alimentación independiente a la de Arduino.

En la sección de códigos de Arduino correspondiente a este servomotor se podrá apreciar el funcionamiento directo de este componente y la forma en la que se debe que programar.

5. SENSOR ULTRASÓNICO

5.1. INTRODUCCIÓN

-Se trata de un dispositivo de medición de distancia que se fundamenta mediante el uso de ondas ultrasónicas. El sensor cuenta con una lámina de material magnetoestrictivo o membrana tiene la propiedad de deformarse mecánicamente y generar ultrasonidos al ser excitada por una corriente eléctrica. Se produce también el mismo efecto a la inversa, es decir, que una vibración mecánica produce una corriente eléctrica. Estos dispositivos emiten una radiación ultrasónica que rebota en los obstáculos que tenga delante y captan los ecos recibidos.

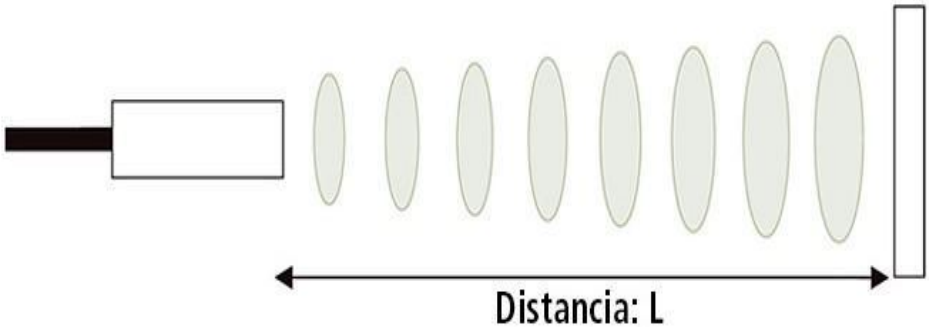


Figura 25: Ondas ultrasónicas

En resumidas palabras, miden la distancia al objeto contando el tiempo entre la emisión y la recepción del tren de ondas ultrasónicas. Para este proyecto se hará uso de este dispositivo para poder implementar una función que permita al automóvil avanzar en piloto automático esquivando objetos.

5.2. CONFIGURACIONES Y CARACTERÍSTICAS

-En cuanto a las posibles configuraciones de este tipo de sensores ultrasónicos tenemos dos opciones:

- **Sistema de emisor/receptor en el mismo transductor**: Las ondas ultrasónicas son generadas con una sola membrana que debe inmediatamente ser "bloqueada" tras emitir el tren de impulsos para poder recibir el rebote de las ondas. Este tipo de configuración requiere un tiempo de bloqueo de la membrana, por lo que en general pierde sensibilidad en distancias muy cortas.
- **Sistema de emisor/receptores separados**: Para esta configuración tenemos dos membranas, las ondas ultrasónicas se generar con una mientras que la otra membrana se utiliza para "escuchar" el rebote las ondas ultrasónicas. De esta manera la membrana receptora está preparada para escuchar desde el mismo instante en el que se acaban de emitir las ondas y por tanto no perdemos tanta sensibilidad para medir distancias cortas. Este tipo de configuración será la que utilice nuestro sensor escogido.

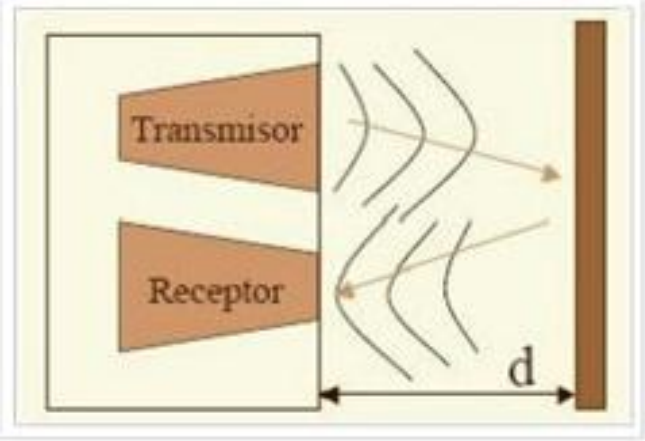


Figura 26: Envío y recepción de ondas ultrasónicas

Con respecto a las características destacar por ejemplo los objetos transparentes. Dado que las ondas ultrasónicas pueden reflejarse en una superficie de vidrio o líquido, y retornar al cabezal, incluso los objetos transparentes pueden ser detectados, por lo que no suponen problema.

Otra característica a destacar es el posible nivel de suciedad. La detección no se ve afectada por la acumulación de polvo o suciedad, así como tampoco es importante la forma o estructura del objeto a la hora de ser detectado

5.3. MEDICIÓN DE DISTANCIA

-Se basa sencillamente en **medir el tiempo entre el envío y la recepción de un pulso sonoro**. Conocemos que la velocidad del sonido es 343 m/s en condiciones de temperatura 20 ºC, 50% de humedad y presión atmosférica a nivel del mar. Con estos datos obtenemos la siguiente relación:

$$343\ \frac{\text{m}}{\text{s}} * 100\ \frac{\text{cm}}{\text{s}} * \frac{1}{1000000} * \frac{\text{s}}{\mu\text{s}} = \frac{1}{29.2}\frac{\text{cm}}{\mu\text{s}}$$

$$Distancia(cm) = \frac{Tiempo(\mu\text{s})}{29.2 * 2}$$

El sonido tarda 29,2 microsegundos en recorrer un centímetro. Por tanto, se puede obtener la distancia por regla de tres a partir del tiempo entre la emisión y recepción del pulso mediante la siguiente ecuación. Se divide el tiempo entre dos debido a que se ha medido el tiempo que tarda el pulso en ir y volver, por lo que la distancia recorrida por el pulso es el doble de la que se quiere medir.

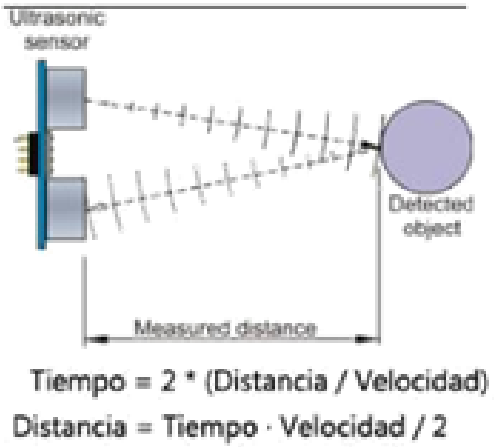


Figura 27: Medición de distancia

Se entenderá mucho mejor el funcionamiento, en cuanto a la forma de tomar medidas, visualizando la programación de Arduino correspondiente a este sensor en el apartado de "programación: códigos Arduino"

5.4 SENSOR ULTRASÓNICO: HC-SR04

-El sensor HC- SR04 está conformado de un transmisor que emite ondas sonoras y un receptor que las recibe cuando las ondas rebotan en un objeto.

Figura 28: Sensor ultrasónico HC-SR04

El HC-SR04 dispone de 4 pines, dos pines para la alimentación, un pin con la función de indicar cuando quieres que emita el pulso de sonido (Trig), y otro pin que se utiliza para enviar la señal de la recepción del sonido (Echo). Dispone de un margen de detección desde objetos a 2 centímetros hasta objetos a 400 cm, con una precisión de 3mm y un ángulo de cobertura de medición de 15 grados.

5.4.1 CARACTERISTICAS SENSOR ULTRASÓNICO

Dimensiones del circuito: 43 x 20 x 17 mm

•Tensión de alimentación: 5 Vcc

•Frecuencia de trabajo: 40 KHz

•Rango máximo: 4.5 m

•Rango mínimo: 1.7 cm

•Duración mínima del pulso de disparo (nivel TTL): 10 µS.

•Duración del pulso eco de salida (nivel TTL): 100-25000 µS.

•Tiempo mínimo de espera entre una medida y el inicio de otra: 20 mS.

5.4.2 CONEXIONES SENSOR ULTRASONICO

Se trata de un conexionado sencillo de los 4 pines. Se aprecia en el diagrama como se emplea el cable rojo y negro para la alimentación del dispositivo y el verde y el amarillo para la transferencia de datos. El pin 6 (Arduino) está conectado con el pin "Trig" del sensor ultrasonidos HC-SR04 y el pin 5 (Arduino) con el pin “Echo”.

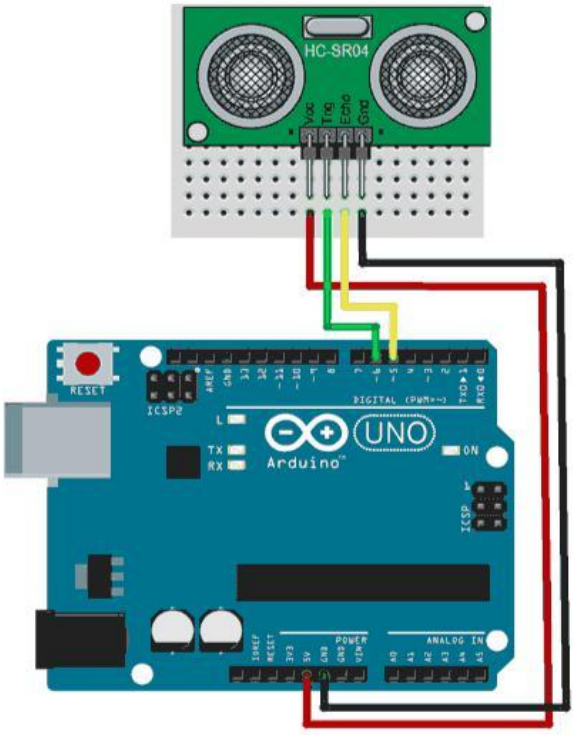


Figura 29: Conexionado módulo HC-SR04

2.1.4. CÁLCULOS HARDWARE

En esta sección se procede a estudiar los cálculos asociados a la máxima potencia que puede disipar el regulador de tensión incorporado en el proyecto,

2.1.4.1. POTENCIA MAXIMA DISIPABLE LM317

Para calcular la potencia que puede disipar un componente se recurre a una sencilla extrapolación de términos, haciendo referencia a las resistencias térmicas y las distintas temperaturas de unión, de forma que se analiza una revisión de la Ley de Ohm para parámetros térmicos. En este caso la similitud son los términos como temperaturas por tensiones, resistencias térmicas por resistencias óhmicas y flujo de calor por corriente eléctrica.

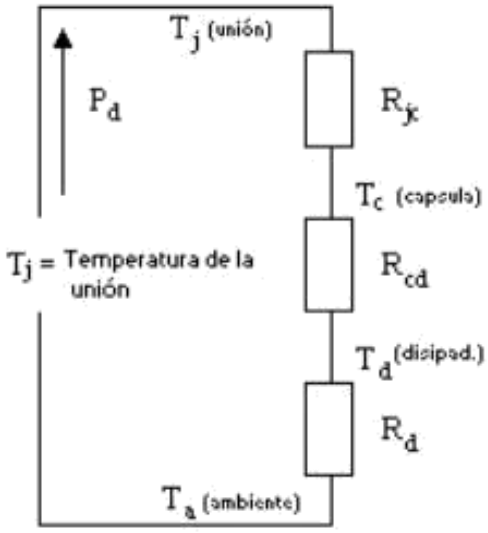


Figura 30: Esquema de resistencias térmicas

Rjc → Resistencia térmica de la junta a la carcasa del componente.

Rcd → Resistencia térmica de la carcasa (capsula) al disipador.

Rd → Resistencia térmica del disipador.

Ta → Temperatura ambiente.

Tj → Temperatura de la unión.

Tc→ Temperatura de la capsula o carcasa.

Td→ Temperatura del disipador.

Pd → Potencia disipada.

Aplicando la Ley de Ohm a nivel térmico quedaría como resultante la siguiente fórmula:

$$(1)\quad Tj - Ta = Pd \times Rja$$

Con esta relación se observa que la potencia disipada en el componente, multiplicada por la resistencia térmica entre la junta y el ambiente es igual a la diferencia entre la temperatura de la junta y la temperatura ambiente.

De esta manera se aprecia que la resistencia térmica entre la junta y el ambiente será la suma de las resistencias térmicas de la junta al capsulado, del capsulado al radiador y del radiador al ambiente, quedándonos de la siguiente forma:

$$Rja = Rjc + Rcd + Rda$$

Sustituyendo en la ecuación (1) se obtiene como resultado la siguiente expresión:

$$Tj - Ta = Pd \times (Rjc + Rcd + Rda)$$

Puesto que lo que interesa saber es la máxima potencia disipable por el dispositivo se despeja esta variable obteniendo la siguiente fórmula:

$$(2)\ Pd = (Tj - Ta) / Rja = (Tj - Ta) / (Rjc + Rcd + Rda)$$

Finalmente se observa que la potencia que puede disipar un dispositivo electrónico es función de la temperatura máxima en la junta (150ºC como temperatura máxima) y de la máxima temperatura ambiente e inversa de la resistencia térmica entre la junta y ambiente.

¿Necesidad de un disipador de calor?

Mediante la fórmula (2) se puede averiguar cuál es la potencia máxima Pd que puede disipar el dispositivo sin añadirle un disipador. Si la potencia que va disipar el dispositivo es igual o mayor a máxima, entonces se necesitará un disipador. Destacar que dependerá de la temperatura ambiente máxima que a la que vaya a estar expuesto.

Lo primero recordamos las características térmicas de nuestro LM317 (TO-263) de su hoja de datos:

THERMAL METRIC[1]		LM317 DCY (SOT-223) 4 PINS	LM317 KCS (TO-220) 3 PINS	LM317 KCT (TO-220) 3 PINS	LM317 KTT (TO-263) 3 PINS	UNIT
$R_{\theta(JA)}$	Junction-to-ambient thermal resistance	66.8	23.5	37.9	38.0	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	43.2	15.9	51.1	36.5	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	16.9	7.9	23.2	18.9	°C/W
Ψ_{JT}	Junction-to-top characterization parameter	3.6	3.0	13.0	6.9	°C/W
Ψ_{JB}	Junction-to-board characterization parameter	16.8	7.8	22.8	17.9	°C/W
$R_{\theta JC(bot)}$	Junction-to-case (bottom) thermal resistance	NA	0.1	4.2	1.1	°C/W

Tabla 5: Información térmica

Para analizar la máxima potencia que podría disipar se dará por supuesto una temperatura ambiente en torno a los 20ºC.

En cuanto a la temperatura de la junta o unión se usa su valor máximo que soporta de 150ºC.

$$Pd = (Tj - Ta) / Rja = (Tj - Ta) / (Rjc + Rcd + Rda) \rightarrow$$

$$\rightarrow Pd = (150ºC - 20ºC) / 38(ºC/W) = 3{,}42W$$

Una vez calculada la potencia máxima disipable sin el uso de un radiador se recuerda la expresión anteriormente calculada, donde se hace una valoración de la potencia real que se necesita disipar:

$$P = (Vin - Vout)*I = (9\ v - 5v) * I = (4*I)\ W$$

La corriente que se puede manejar con el LM317 como ya se ha comentado oscila entre 1mA y 1.5A

		MIN	MAX	UNIT
V_O	Output voltage	1.25	37	V
$V_I - V_O$	Input-to-output differential voltage	3	40	V
I_O	Output current	0.01	1.5	A
T_J	Operating virtual junction temperature	0	125	°C

Tabla 6: Valores máximos y mínimos

Por consiguiente, en función de la corriente que se necesite es posible que requiera un radiador o no.

$$4*I \leq 3{,}42\ W \rightarrow I \leq 0{,}855A$$

Se estima en base a los cálculos que por encima de una corriente de 855mA sería conveniente el uso de un radiador y por debajo de estos no sería necesario.
En la sección de anexos II se dispone de un apartado en el que se incluye ejemplos de distintos tipos de radiadores para usar en este tipo de componentes, indicando el tipo de materiales de cada uno y sus dimensiones.

2.2. ESTUDIO DE LA ARQUITECTURA SOFTWARE

[7] La arquitectura de software es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la implementación y desarrollo de cualquier proyecto. Es un concepto fácil de entender pero difícil de definir con precisión. La arquitectura es algo más que una estructura; el IEEE Working Group on Architecture define esta como **[8]** "the highest-level concept of a system in its environment" (el concepto de más alto nivel de un sistema en su entorno). Además de ser un modelo abstracto y reutilizable que puede transferirse de un sistema a otro, permitiendo de esta manera la interacción e intercambio entre los distintos desarrolladores con el fin de establecer un intercambio de conocimientos y puntos de vista entre ellos.

Software es un término informático que hace referencia a un programa o conjunto de programas de cómputo que incluye datos, procedimientos y pautas que permiten realizar distintas tareas en un sistema informático.

En este apartado se estudia el software de control del que se ha dotado al microcontrolador Arduino para desempeñar sus distintas funciones. En primera instancia se explica el entorno de programación en el que se trabaja, continuando por el tipo de comunicación serial empleada (explicando los procesos de configuración y vinculación del módulo bluetooth) y finalizando mediante la presentación de las distintas simulaciones llevadas a cabo y la programación correspondiente.

2.2.1 ENTORNO DE PROGRAMACIÓN

-El entorno de programación de Arduino (IDE, Integrated development environment) está constituido por un editor de texto para escribir el código (en C++), un área para mostrar mensajes, consola de texto, barra de herramientas con una serie de botones para las funciones más empleadas y una serie de distintos menús. Permite la conexión, por USB, con el hardware de Arduino para cargar los programas y comunicarse con ellos.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación. Esto consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Además incorpora las herramientas para cargar el programa ya compilado en la memoria flash del hardware a través del puerto serie.

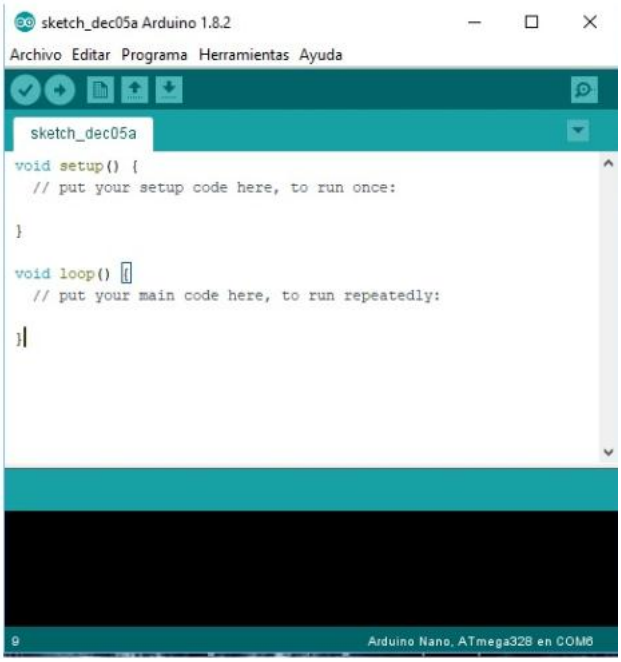


Figura 31: Arduino IDE, entorno de desarrollo, sketch.

[9]Focalizando en las funciones más importantes se destacan las siguientes opciones:

- **Nuevo y Abrir;** respectivamente permiten crear y abrir un sketch anteriormente guardado.
- **Proyecto**; permite abrir un conjunto de sketchs determinado que conforman un proyecto concreto.
- **Ejemplos**, se disponen ejemplos de sketch sobre el uso de la placa de Arduino. Desde ejemplos básicos, tales como hacer parpadear un led, hasta implementar un servidor web básico con Arduino y el shield ethernet.

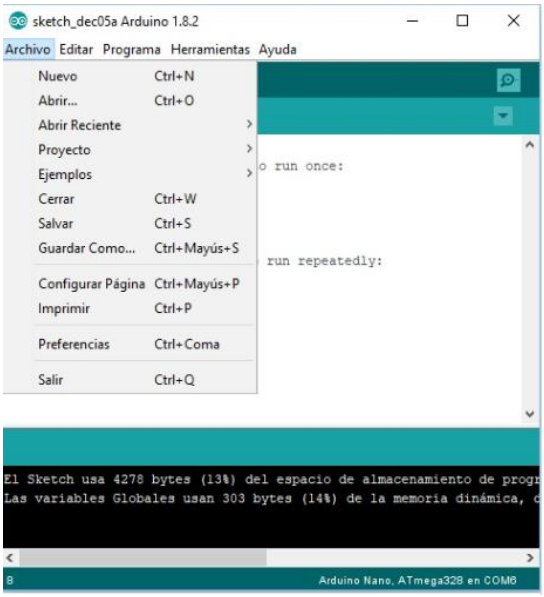


Figura 32: Arduino IDE, entorno de desarrollo, sketch.

En el menú "editar" se encuentran de las opciones básicas de edición (copiar, cortar y pegar), además de copiar el texto como html (necesario para publicar nuestro código).
El menú programa existen las opciones de **verificar y compilar el programa, incluir librerías** y **mostrar la carpeta del programa**, lo que resultará beneficioso a la hora de necesitar encontrar los sketchs sin tener que localizar la carpeta de los ficheros en la carpeta de usuario dentro de la carpeta Arduino.

- PROGRAMACIÓN

La estructura correspondiente a un código de Arduino atiende a la siguiente forma: en primer lugar el **void setup ()** donde se indica el modo de funcionamiento de los pines (entrada y salida), comunicación serie, declaración de variables etc…. **[10]** A continuación se encuentra el bloque del programa principal en el que se definen las acciones que se espera que realice el programa, **void loop ().** La función loop() realiza precisamente lo que sugiere su nombre, el código se ejecuta de forma cíclica, lo que posibilita que el programa este respondiendo continuamente ante los eventos que se produzcan.

2.2.2. COMUNICACIÓN BLUETOOTH

-Una vez estudiado el componente usado como módulo bluetooth en la sección de "componentes hardware de control", además de como conectarlo, se explica la forma de cómo conseguir configurarlo y vincularlo. De esta forma servirá de interfaz entre el microcontrolador y el dispositivo smartphone, desde el que serán enviados los comandos para que este ejecute las acciones correspondientes.

2.2.2.1. CONFIGURACIÓN DEL MÓDULO BLUETOOTH

-El hardware de Arduino ha incorporado el soporte para la comunicación serie en los pines 0 y 1 (que también va al ordenador través de la conexión USB). El soporte serie pasa a través de un componente de hardware (integrado en el chip) llamado UART. Este hardware permite al chip ATmega recibir la comunicación serie incluso mientras trabaja en otras tareas, siempre que haya espacio en la memoria intermedia serie de 64 bytes.
El modulo bluetooth es básicamente un nodo BT conectado a un interface serie, por lo que se podrá conectar los pines RX y Tx a los los pines 0 y 1 equivalentes de Arduino, teniendo en cuenta que se cruzan (Tx (BT) a Arduino Rx yRx (BT) a Arduino Tx). De esta forma se implementa la comunicación con el bluetooth mediante las instrucciones de Serial.print ().
La única pega de esta configuración es que se debe desconectar el módulo bluetooth para poder cargar código desde el PC (como ya se ha mencionado anteriormente). Esto se debe a que los pines 0 y 1 se utilizan en la comunicación serie de Arduino con el PC a través del USB y por tanto, si se usan para comunicar con el módulo BT, se pierde la conexión con el PC.

Para evitar esta situación una posible opción será destinar otro par de pines cualesquiera a la transmisión, donde para ello se tendría que importar una librería que habilite la comunicación serie con otros pines como es la "*librería SoftwareSerial"*.

2.2.2.2.CODIGO DE CONFIGURACIÓN DEL MÓDULO HC-SR06

-Una vez introducido la configuración del módulo se muestra el código correspondiente para definir el nombre y la contraseña personal escogida. Queda de la siguiente manera:

```
const int LED = 13; //Se comienza por asociar al pin13 a la variable LED que ayudará a observar visualmente el proceso (en el pin13 todos los Arduinos tiene un led asociado)

const int BTPWR = 12; // En este pin se establece la alimentación correspondiente al módulo de bluetooth. Esto se debe a que se evita tener que desconectar el módulo BT durante la carga del código, puesto que no se pondrá a 1 hasta haber cargado los comandos.

char nombreBT[5] = "EUITI";          //Se asocia el nombre que le hemos dado a nuestro módulo a una cadena de caracteres (5 en concreto para nuestro caso)

char velocidad ='4'; //9600bps          //Aquí se define la velocidad de comunicación en bps(baudios)
```

En cuanto a las distintas velocidades de comunicación posibles que se pueden aplicar, se aprecia en la siguiente tabla las 8 velocidades distintas disponibles:

1	1200bps
2	2400bps
3	4800bps
4	9600bps
5	19200bps
6	38400bps
7	57600 bps
8	115200 bps

Tabla 7: Velocidades de comunicación (bps)

```
char pin [5]= "0000"; //Por último se define la clave correspondiente que se necesitara para establecer la conexión.
```

Se procede con el código una vez definidas las variables que sirven para guardar el nombre, la contraseña y la velocidad de transmisión de datos.

```
void setup(){
  pinMode(LED, OUTPUT);            //Configuración  del pin 13 (LED) como pin de salida
  pinMode(BTPWR, OUTPUT);      // Configuración del pin 12 (BTPWR) como pin de salida
  digitalWrite(LED, LOW);          //Se pone a 0 el led 13
  digitalWrite(BTPWR, HIGH);      // Se introduce un uno lógico en el pin 12 alimentando el módulo BT
  Serial.begin(9600);       //Velocidad del módulo bluetooth, 9600 por defecto
  Serial.print("AT");  //Inicialización de comunicación con modulo bluetooth mediante comandos AT
  delay(1000);          //Espera de 1 segundo según datasheet entre envió de comandos AT

  Serial.print("AT+NAME");          //Cambio de nombre donde se envía AT+NAME y seguido el nombre elegido.
  Serial.print(nombreBT);     //Introducimos la variable que previamente se ha asociado con el nombre escogido .
  delay(1000);   //Espera de 1 segundo según datasheet entre envió de comandos AT

  Serial.print("AT+BAUD");           //Cambio de la velocidad del módulo en baudios
  Serial.print(velocidad);       //Se introduce la variable que previamente hemos asociado con la velocidad escogida
  delay(1000);          //Espera de 1 segundo según datasheet entre envió de comandos AT

  Serial.print ("AT+PIN");   //Configuración password, se envía AT+PIN y seguido password que permitirá vincularse al módulo
  Serial.print(pin);      //Se introduce la variable que previamente hemos asociado con nuestra clave de acceso
  delay(1000);          //Espera de 1 segundo según datasheet entre envió de comandos AT
  digitalWrite(LED, HIGH); //Se pone a 1 el led para indicar que se ha finalizado la configuración correctamente
}

void loop(){ /                    //Bucle principal (vacío)

  }
```

En este punto ya se ha configurado el módulo HC-06 con nuestro nombre y clave escogidas.

2.2.2.3. VINCULACIÓN MÓDULO BLUETOOTH CON SMARTPHONE

-En primera instancia se debe activar el bluetooth desde el teléfono y buscar los dispositivos de bluetooth disponibles con los que conectarse. Se observa cómo sale el dispositivo HC-06 nombrado "*EUITI*".

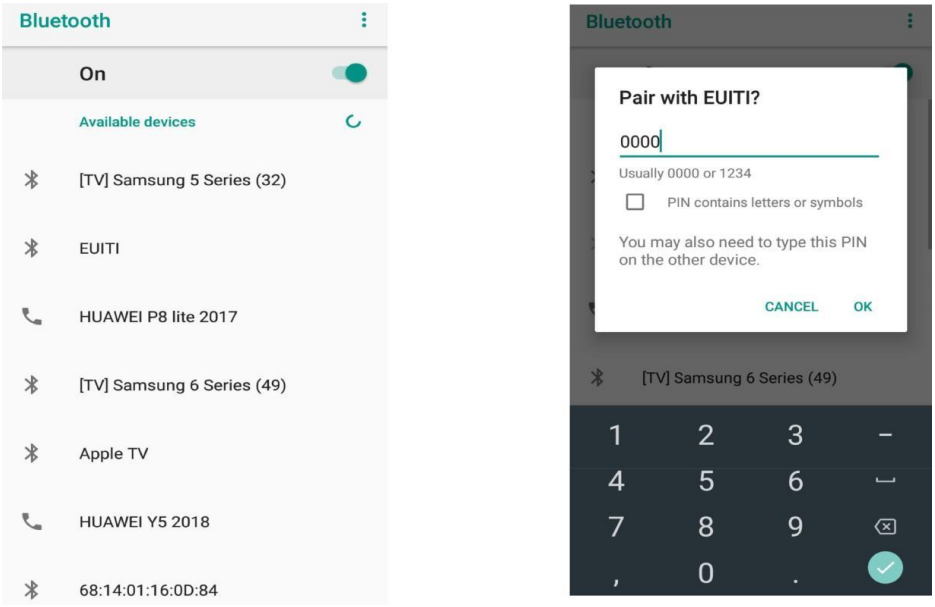


Figura 33: Vinculación mediante módulo bluetooth

Una vez localizado nuestro módulo se pincha sobre este para vincularse e introducir la clave previamente escogida (en nuestro caso es 0000). Después de introducir la clave en el smartphone ya se ha vinculado con el módulo HC-06 vía bluetooth correctamente, como se puede apreciar en la siguiente imagen:

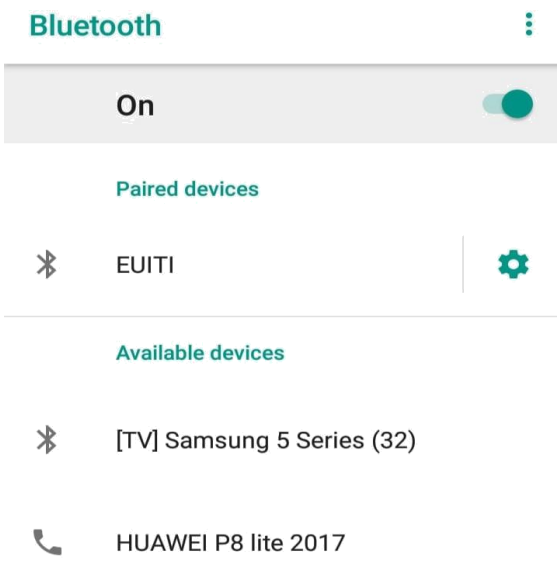


Figura 34: Dispositivos vinculados

2.2.2.4. ENVIO Y RECEPCION DE DATOS

-Previamente se ha conseguido realizar la vinculación del módulo HC-06 con el teléfono móvil, ahora se procede a conectarlo con el microcontrolador (Arduino). Esta conexión se lleva a cabo por medio de un puerto serial de nuestro Arduino, mediante el cual se podrán ver reflejados en el monitor serie de Arduino los datos enviados desde el dispositivo móvil.

Figura 35: Diagrama de envío y recepción de datos entre módulo bluetooth y Arduino

En esta ocasión se ha optado por incluir la librería "*SoftwareSerial.h"* y dejar libres los pines de comunicación serie 1 y 0 del microcontrolador para no interferir con la conexión USB durante la carga de códigos.

La biblioteca "*SoftwareSerial.h"* sirve para permitir la comunicación serie en otros pines digitales del Arduino, usando el software para replicar la funcionalidad (de ahí el nombre de "SoftwareSerial"). Es posible tener múltiples puertos serie de programas con velocidades de hasta 115200 bps.

2.2.2.5. CODIGO DEL ENVIO Y RECEPCION DE DATOS

Se expone y estudia el diagrama de flujo y los comandos correspondientes al envío y recepción de datos.

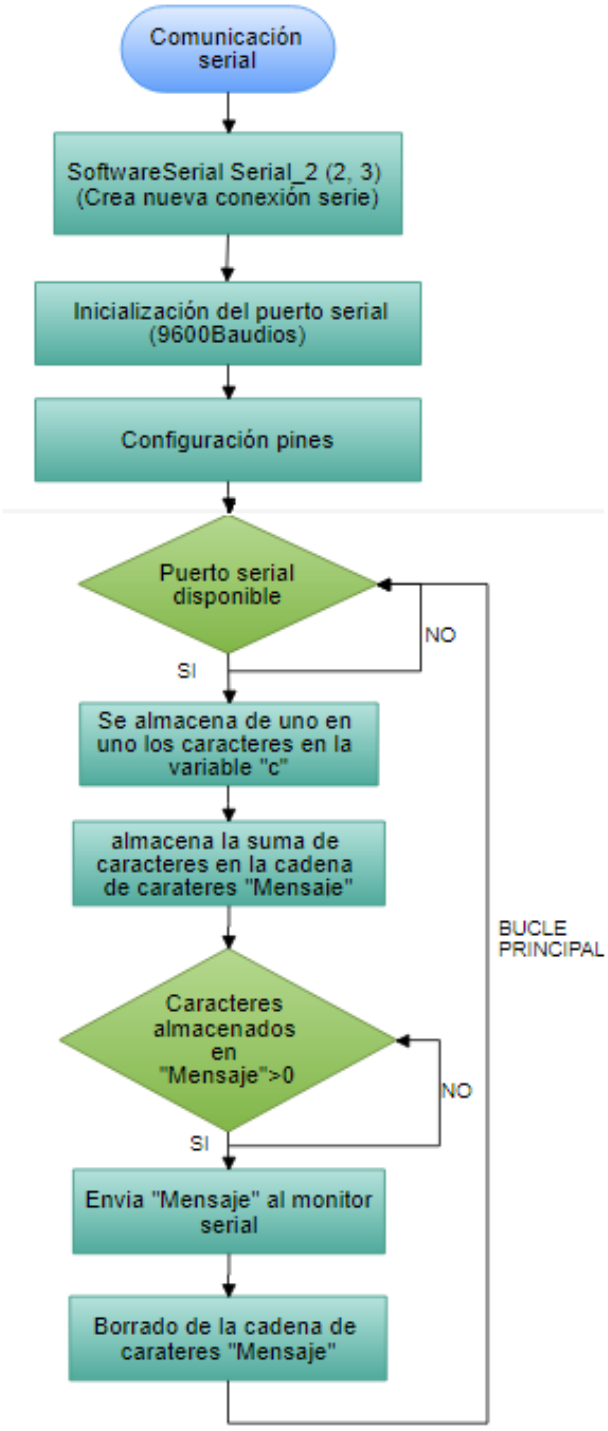


Diagrama 1: Comunicación serial

```
#include <SoftwareSerial.h>
SoftwareSerial Serial_2 (2, 3); // Crea nueva conexión serie- Pin2 (RX) a TX y Pin3 (TX) a RX
String Mensaje; // Variable de cadena de caracteres para almacenar el mensaje


void setup() {
   Serial_2.begin(9600); // Inicialización del nuevo puerto Serial_2 a 9600 Baudios


   pinMode(13,1);          // Configuración del pin 13 como Salida. Otra manera de indicarlo sería “pinMode(13,OUTPUT)”
}
void loop() {
  while (Serial_2.available()) { // Mientras que el puerto Serial_2 esté disponible, es decir, mientras que lleguen caracteres
  delay(5); // espera de 5 milisegundos
     char c = Serial_2.read(); // Lee los caracteres uno a uno y los almacena en la variable c
     Mensaje += c;              // Almacena la suma de caracteres en el mensaje
  }
    if (Mensaje.length()>0){
    Serial.println(Mensaje);    // envía mensaje al PC
    Mensaje="";                 // Borrado del mensaje ya enviado al PC
}
```

Introducido y cargado el código se pueden enviar datos desde el teléfono y recibirlos en el monitor serie de Arduino. Para ello se necesita lo primero un terminal bluetooth (aplicación) en el móvil para llevar a cabo la conexión. En este caso se ha optado por *"**Bluetooth SPP**"*.

Bluetooth SPP Manager es una aplicación que proporciona comunicación entre dispositivos habilitados para conexión Bluetooth. Utiliza el **protocolo RFCOMM**, también conocido como protocolo de puerto serie (SPP) para transmitir datos entre dispositivos.

Figura 36: Protocolo de puerto serie, SPP

Esta aplicación muestra varias propiedades de los dispositivos Bluetooth conectados, así como dispositivos alrededor y permite la comunicación con ellos. Un servidor incorporado puede aceptar conexiones entrantes desde cualquier otro dispositivo. Además esta aplicación tiene un Real Time Clock Manager integrado. Con esta función, alguien puede enviar el tiempo actual o un tiempo predefinido a un dispositivo remoto. Por ejemplo, esto permite configurar el RTC de un microcontrolador y mostrar la hora actual en un LCD.

Figura 37: Menú de aplicación bluetooth SPP

Este terminal bluetooth para el teléfono permite dos modos de comunicación: modo terminal y modo control. Para esta primera toma de contacto el modo terminal permitirá enviar caracteres (asociados a órdenes o comandos) al microcontrolador, quedando estos reflejados en el monitor serial. Lo primero que se debe hacer es asegurarse de tener habilitado el bluetooth en el dispositivo y entonces pulsar en "*buscar dispositivos"*.

Figura 38: Menú de dispositivos bluetooth emparejados

Una vez muestra la lista de todos los dispositivos bluetooth con los que se puede establecer una conexión se busca el módulo que se esté utilizando y se vincula mediante la aplicación (bluetooth SPP). En el momento en el que se realice la conexión se aprecia como el led rojo que parpadea en el módulo HC-06 deja de hacerlo para indicar que ha sido finalmente vinculado con un dispositivo.

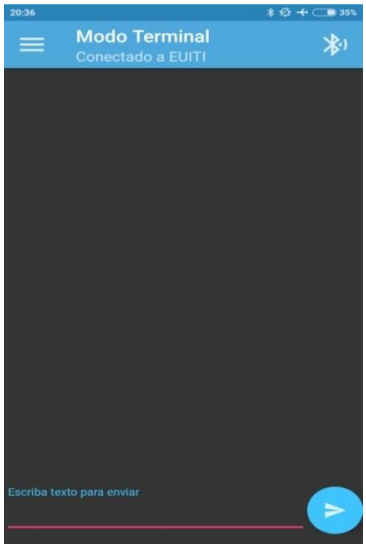


Figura 39: Modo terminal (bluetooth SPP)

En este punto en el que ya se ha establecido una conexión se puede acceder al modo terminal y realizar una prueba para comprobar el correcto funcionamiento del envío de datos.

Figura 40: Modo terminal (bluetooth SPP)

Se observa cómo se transmite correctamente la información introducida desde el dispositivo móvil al Arduino, quedando reflejada en el monitor serial.

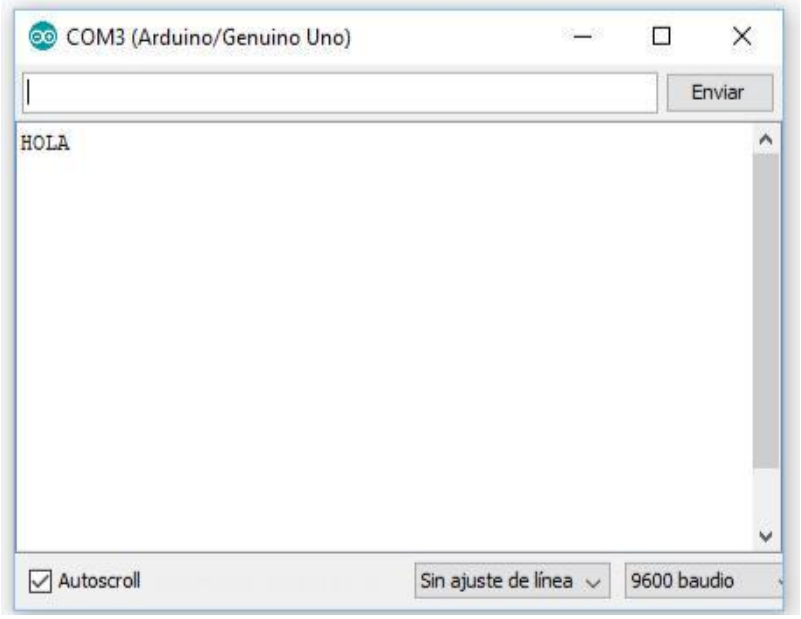


Figura 41: Monitor serie Arduino

Figura 42: Modo control

Figura 43: Configuración de botones

El modo control por ejemplo será útil de cara al control final del coche teledirigido, puesto que cuenta con una serie de botones configurables de forma que se pueda controlar y modificar la trayectoria del coche, es decir, a modo de joystick.

Además de lo ya mencionado cuenta con otros tres botones configurables para añadirle cualquier otro tipo de función (como podría ser apagar y encender las luces).

2.2.2.6.EJEMPLO PRACTICO

-A continuación se demuestra, mediante un sencillo ejemplo práctico, la funcionalidad de poder enviar datos desde un dispositivo (smartphone en este caso) por medio del bluetooth al microcontrolador.

Esta prueba consiste en encender y apagar un led poniendo a nivel alto y a nivel bajo una de las salidas digitales de Arduino en función de los datos que enviados. El código y el algoritmo correspondiente quedan de la siguiente forma:

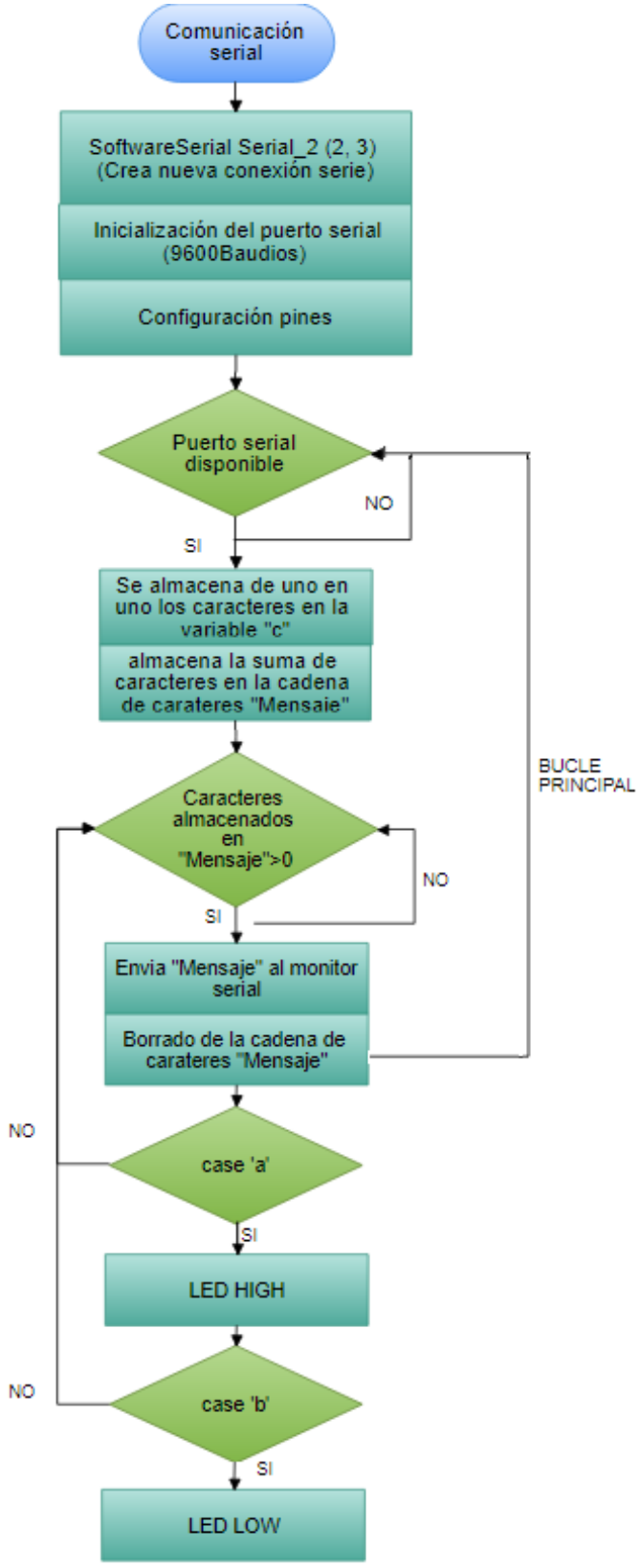


Diagrama 2: Ejemplo práctico (comunicación serial)

```
#include <SoftwareSerial.h>
SoftwareSerial Serial_2 (2, 3);          // Pin3 (TX) a RX
String Mensaje;
int LED = 12;
void setup() {
    Serial_2.begin(9600);
    Serial.begin(9600);        // inicialización a 9600 baudios
    pinMode(LED,OUTPUT);
}
void loop() {
  while (Serial_2.available()) {
     delay(5);
     char c = Serial_2.read();
     Mensaje += c;
 switch(c){
       case 'a':
       digitalWrite(LED,HIGH);
       break;
       case 'b':
       digitalWrite(LED,LOW);
       break;
      }
}
if (Mensaje.length()>0){
Serial.println (Mensaje);        // envía mensaje al PC
}
Mensaje="";              // Borra el mensaje ya enviado al PC
}
```

Se aprecia como lo único que cambia del código es la en la que se especifica que se debe poner a 1 la salida digital del pin 12 (encender led) en el caso de recibir el carácter 'a'.

Para la operación contraria, poniendo a 0 el pin (apagar led), se debe enviar desde el smartphone y recibir por el puerto serial del microcontrolador el carácter 'b'. Este se trata de un ejemplo muy sencillo a nivel práctico, sin embargo sirve para extrapolar sobre cómo se envían las órdenes a la hora de manejar los motores correspondientes al vehículo teledirigido.

Destacar que de este último código, aparte de controlar uno de los pines digitales, se proyectan los datos enviados en nuestro monitor serial. Si no interesa reflejar los datos sobre el monitor serial de Arduino el código se simplifica bastante:

```
#include <SoftwareSerial.h>
SoftwareSerial Serial_2 (2, 3); // Crea nueva conexión serie- Pin2 (RX) a TX y Pin3 (TX) a RX
int Mensaje=0;                  // Variable de cadena de caracteres para almacenar el mensaje
int LED = 12;
void setup() {
Serial_2.begin(9600);
Serial.begin(9600);
pinMode(LED,OUTPUT);
}
void loop() {
if (Serial_2.available()>0) {
Mensaje = Serial_2.read();          }
switch(Mensaje){
case 'a':
digitalWrite(LED,HIGH);
break;
case 'b':
digitalWrite(LED,LOW);
      break;
    }
}
```

2.2.3. SIMULACIONES

En este apartado se muestran las simulaciones realizadas para estudiar y comprender el funcionamiento de varios de los componentes que empleados en el proyecto.

2.2.3.1 REGULADOR DE TENSION

Recordar el circuito que sirve de regulador de tensión utilizando el LM317 como regulador. Se procede a analizar por medio de la simulación su funcionamiento.

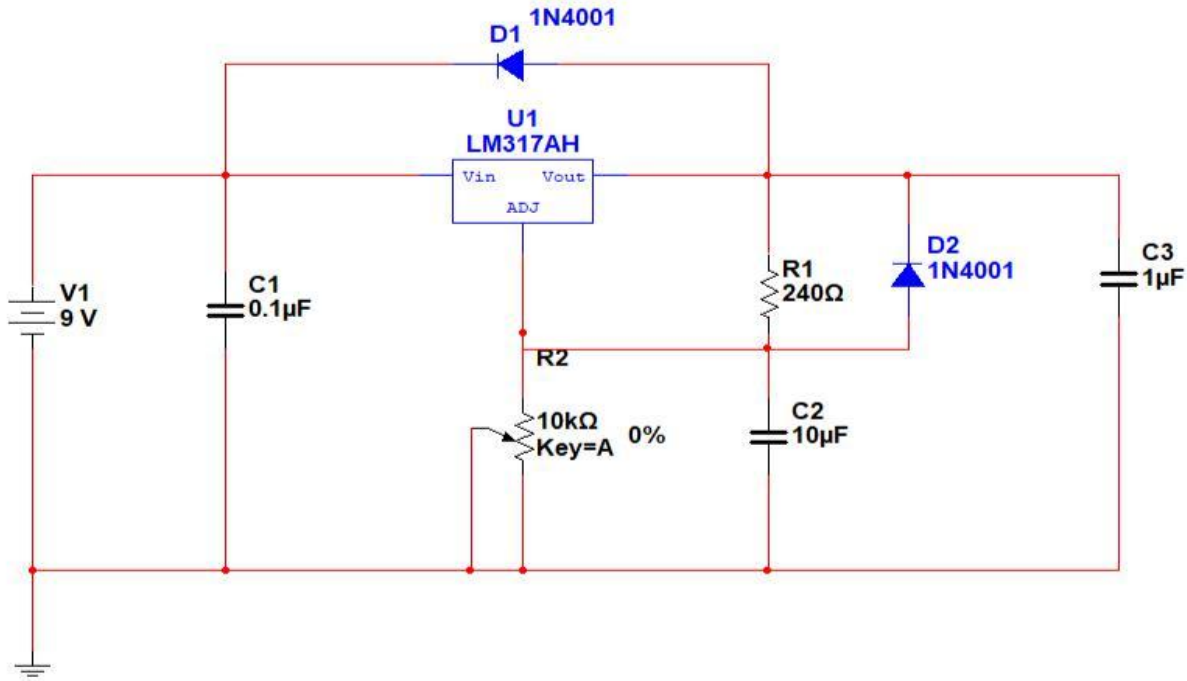


Figura 44: Regulador de tensión con salida variable con diodos de protección

La tensión entre la terminal ADJ y VOUT del LM317 es siempre de 1.25 V (tensión establecida internamente por el regulador) y en consecuencia la corriente que circula por la resistencia R1 es:

$$I_{R1} = V / R1 = 1.25\ V / R1$$

Esta misma corriente circula por la resistencia R2 más la corriente de ajuste, que comúnmente tiene un valor máximo de 100µA. Entonces la tensión en R2 es:

$$V_{R2} = R_2 * Iadj + R2 * I_{R1}$$

Si se sustituye IR1 en la última fórmula se obtiene:

$$V_{R2} = R2 * Iadj + R2 * (1.25/R1)$$

Como la tensión de salida es:

$$Vout = VR1 + VR2$$

$$Vout = 1.25 + R2 * Iadj + (1.25 * R2/R1)$$

Vout = 1.25 + (1+R2/R1)+ Iadj*R2

En la práctica como la corriente de ajuste **Iadj** será como mucho de 100µA se puede despreciar quedando la fórmula final simplificada de la siguiente manera:

Vout = 1.25 + (1+R2/R1)

Para este caso se busca un voltaje de salida de 5V, entonces la resistencia del potenciómetro responderá a la siguiente ecuación:

R2 = ((Vout/1,25)-1)*240Ω

R2 = 720Ω

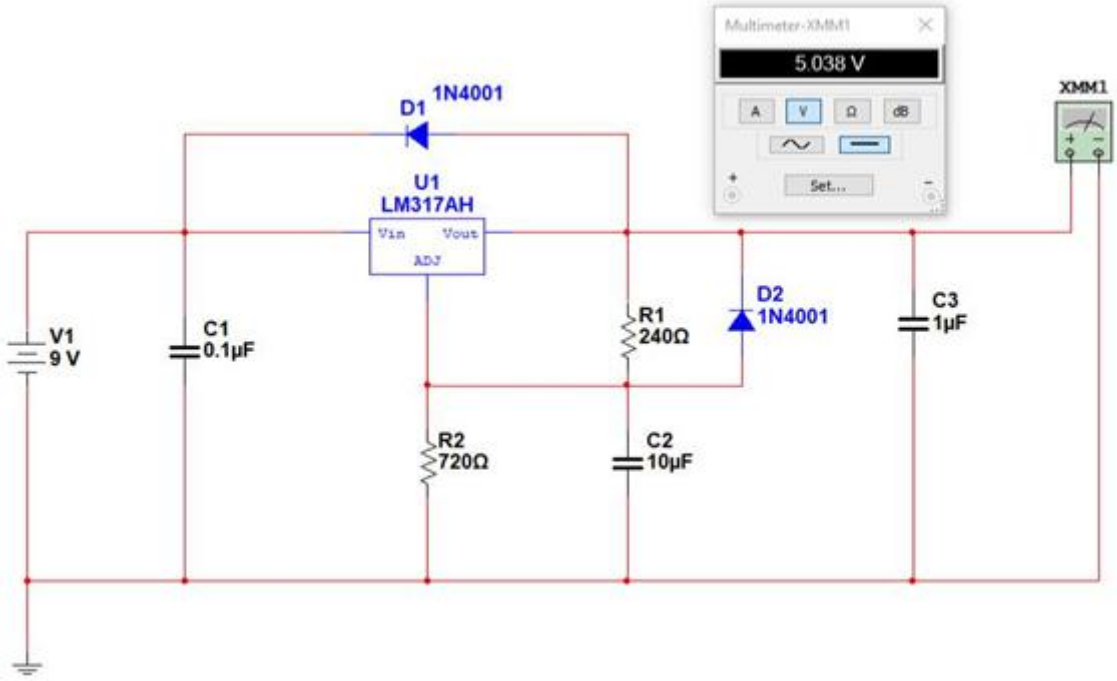


Figura 45: Regulador de tensión con salida variable con diodos de protección (sin potenciómetro)

A nivel práctico no se usa una resistencia de valor fijo sino un potenciómetro. De esta manera basta con ir observando la caída de tensión que obtenemos a la salida y girar gradualmente el potenciómetro hasta obtener el valor exacto de voltaje deseado.

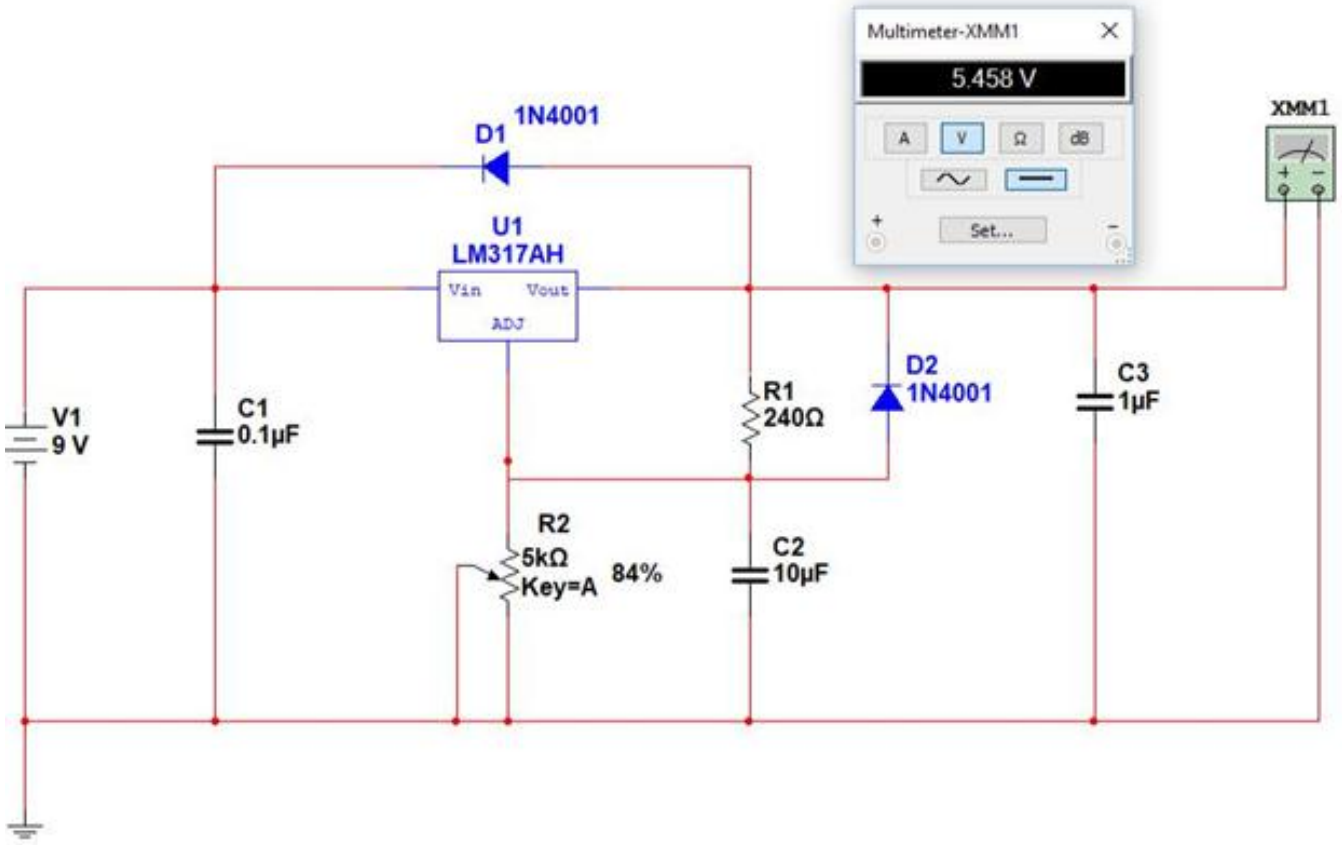


Figura 46: Regulador de tensión con salida variable con diodos de protección (con potenciómetro)

2.2.3.2. PUENTE H

Se procede a comprobar el cambio de la polaridad de la tensión que recibe el motor por medio del estudio de las corrientes del circuito. En referencia a la tabla de la verdad mencionada se simulan los dos primeros casos:

- **Motor gira en avance (S1=1||S2=0||S3=0||S4=1)**

En esta situación se encuentran los transistores S1 y S4 conduciendo en saturación aplicándose al motor tensión positiva al motor.

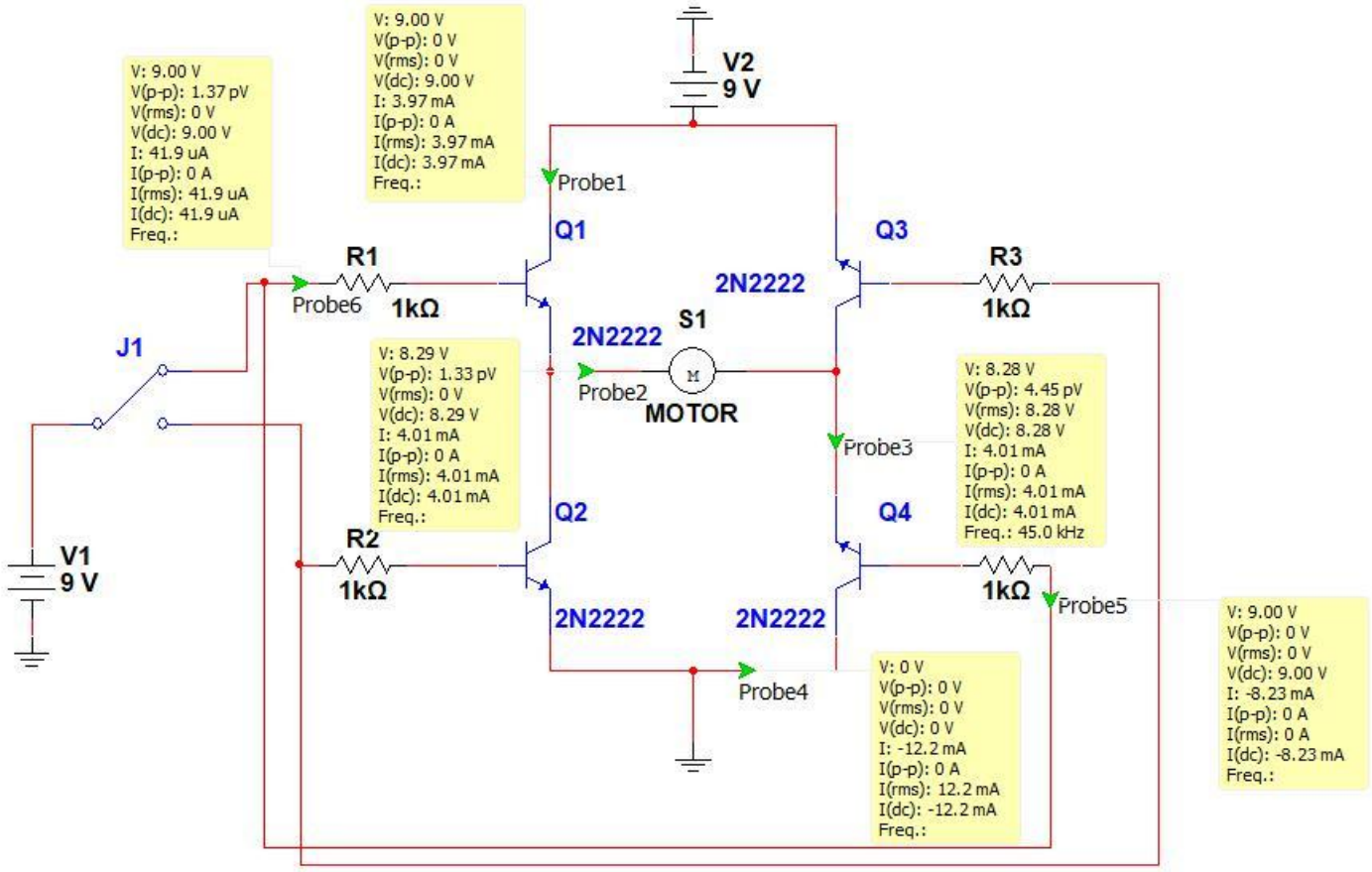


Figura 47: Circuito puente H

- **Motor gira en retroceso (S1=0||S2=1||S3=1||S4=0)**

Para este segundo caso se encuentra la situación contraria con los transistores S2 y S3 conduciendo en saturación, obteniendo en el motor una tensión con polaridad opuesta al ejemplo anterior.

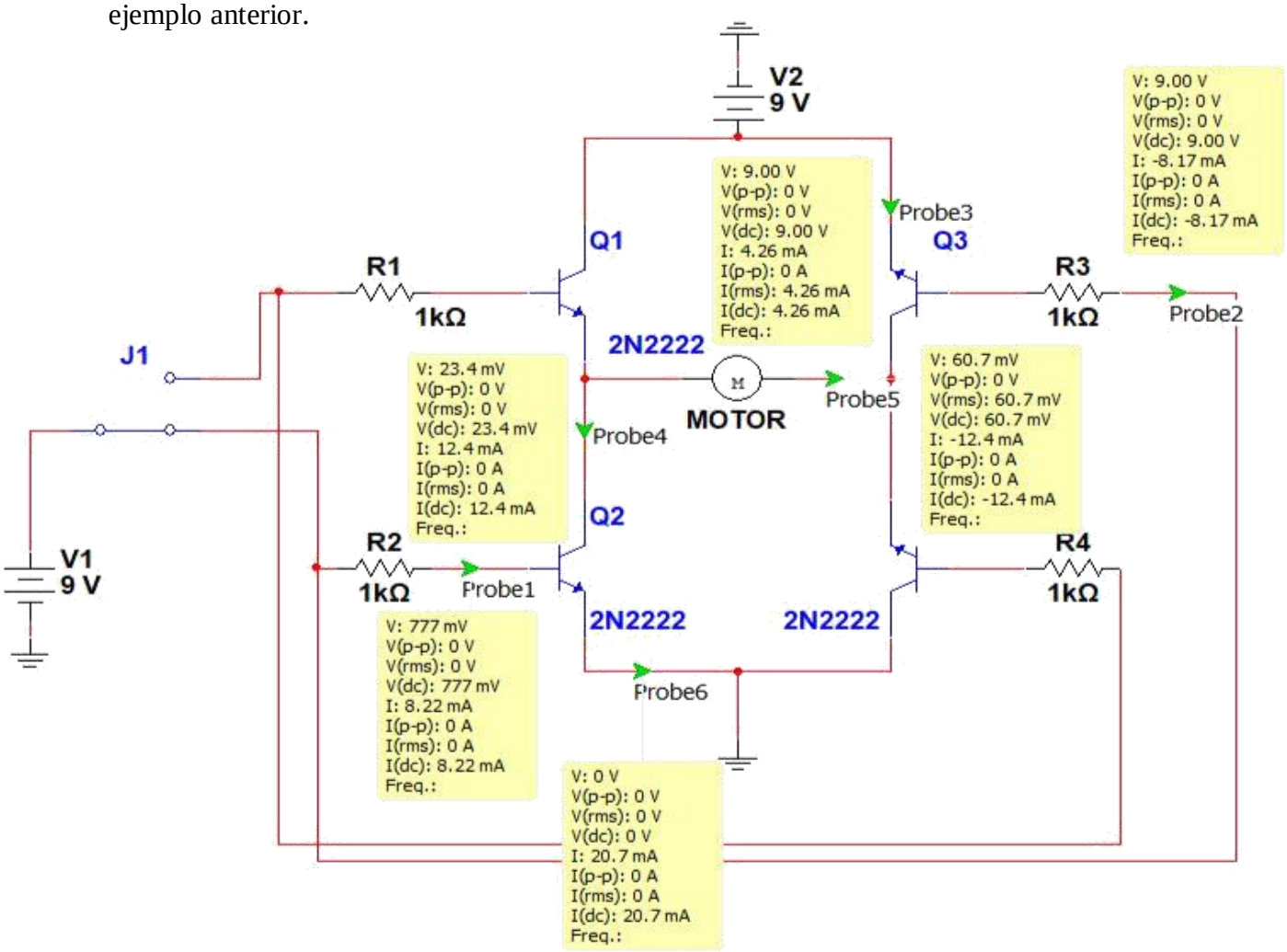


Figura 47: Circuito puente H

2.2.4 PROGRAMACIÓN: CÓDIGOS ARDUINO

- **Módulo L298N (Puente H)**: Modos de operación

Se emplean 4 pines del microcontrolador para el control de los dos motores. La distribución de pines queda de la siguiente forma:

Módulo L298N	Arduino
IN1 (DC1-Motor derecha)	pin7
IN2(DC1-Motor derecha)	pin6
IN3(DC2-Motor izquierda)	pin5
IN4(DC2-Motor izquierda)	pin4

Tabla 8: Modos de operación

Si se envía un 1 lógico por la entrada IN1 del driver, saldrán 6V por la salida OUT1 del módulo, en el caso de enviar un 0 lógico por IN1, saldrá GND (0V) por OUT1.

Entonces para poder manejar con soltura un vehículo mediante estos necesitaremos 4 operaciones concretas:

1. Operación de avance

2. Operación de retroceso.

3. Operación de giro sentido horario.

4. Operación de giro sentido antihorario.

A continuación antes de empezar con el código se formula la siguiente tabla de modos de operación:

	DC1 (MOTOR-DERECHA)		DC2 (MOTOR-IZQUIERDA)	
INPUT	IN1	IN2	IN3	IN4
AVANCE	1	0	1	0
RETROCESO	0	1	0	1
GIRO ANTIHORARIO	1	0	0	1
GIRO HORARIO	0	1	1	0
PARAR	0	0	0	0

Tabla 9: Tabla de verdad del módulo L298N

Para simplificar la nomenclatura, con respecto al código, se va a denominar los pines de entrada de la siguiente forma:

DC1-IN1 → motorDer1

DC1-IN2 → motorDer2

DC2-IN1 → motorIzq1

DC2-IN2 → motorIzq2

Una vez establecidos los diferentes modos de operación y la nomenclatura se pasan a estudiar directamente el código de Arduino junto a su diagrama de flujo correspondiente.

- **Diagrama de flujo:** Control de motores DC

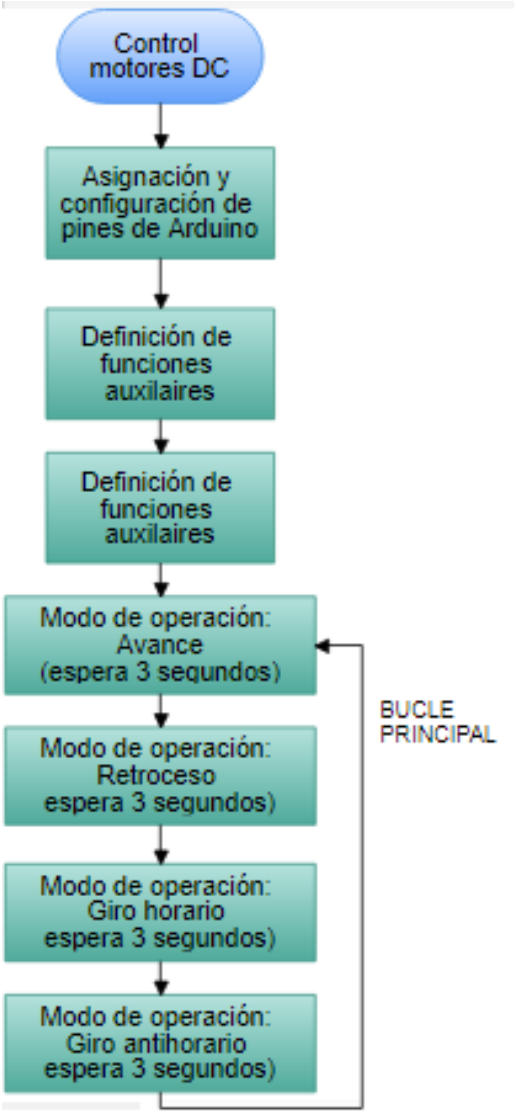


Diagrama 3: Control motores DC

- **Código:** Control de motores DC

```
// Lo primero se definen los pines a usar del Arduino asociándolos a su pin correspondiente de entrada del módulo del puente en H
int motorDer1=7;      //El pin 7 a In1 del L298N
int motorDer2=6;      //El pin 6 a In2 del L298N
int motorIzq1=5;      //El pin 5 a In3 del L298N
int motorIzq2=4;      //El pin 4 a In4 del L298N
void setup(){         //Configuración de los pines que se acaban de definir como pines de salida(de forma que se pueda introducir un 1 o 0 lógicos a través de estos)
pinMode(motorDer1, OUTPUT);
pinMode(motorDer2, OUTPUT);
pinMode(motorIzq1, OUTPUT);
pinMode(motorIzq2, OUTPUT);
}
//Se comienzan a definir las funciones que serán llamadas desde el bucle principal (aquí es donde se aprecia implementada la tabla de la verdad del módulo L298N anteriormente mencionada). Modo de operación: Retroceso
void atras(){
digitalWrite(motorDer1,HIGH);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,HIGH);
digitalWrite(motorIzq2,LOW);}
/Modo de operación: Avance
void adelante(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,HIGH);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);}

//Modo de operación: Giro en sentido horario
void giraDerecha(){
```

```
digitalWrite(motorDer1,HIGH);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);
}
//Modo de operación: Giro en sentido antihorario
void giraIzquierda(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,HIGH);
digitalWrite(motorIzq1,HIGH);
digitalWrite(motorIzq2,LOW);
}
//Modo de operación: Pararse
void parar(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,LOW);
}
//  Ya definidas todas las funciones auxiliares se pasa al bucle principal del código
void loop() {
adelante();
delay(3000);
atras();
delay(3000);
giraDerecha();
delay(3000);
giraIzquierda();
delay(3000);}
```

Se trata de un programa que realiza en bucle todos los modos de operación definidos, primero avanza, retrocede, gira en sentido horario y por último gira en sentido antihorario.

En el bucle principal se puede ver como el programa va llamando una a una a todas las funciones o subrutinas definidas y deja un margen de 3 segundos para cada una de ellas. Cuando el programa llega a la última función deja otros 3 segundos de margen y vuelve a comenzar el bucle. Para esta primera toma de contacto con el puente en H y su funcionalidad se ha comprobado cómo responden ambos motores a nuestro programa, sin embargo no se ha tenido en cuenta ni valorado la velocidad a la que se requiere que giren los motores.

- **Módulo L298N (Puente H)**: Control de velocidad

-El objetivo del control de la velocidad es buscar un manejo más cómodo y sobretodo preciso del vehículo teledirigido. Para poder controlar la velocidad es necesario conectar en los pines ENA y ENB del módulo dos salidas PWM de nuestro Arduino. Como ya se ha mencionado anteriormente en el estudio del microcontrolador, las salidas PWM son pines digitales capaces de generar una salida analógica.

ENA (módulo L298N) → pin10-PWM (Arduino)

ENB (módulo L298N) → pin11-PWM (Arduino)

Estas serían las únicas conexiones que se tendrían que añadir al diagrama del conexionado estudiado previamente. En cuanto a las velocidades que se van a escoger, para facilitar el manejo del vehículo, se debe tener en cuenta que la PWM que puede generar Arduino es de 8 bits, por lo que se elegirá un valor entre 0 (parado) y 255 (motores al máximo).

MODO DE OPERACION	VELOCIDAD(0-255)
AVANCE	200
AVANCE-TURBO	255
RETROCESO	200
GIRO HORARIO	170
GIRO ANTIHORARIO	170

Tabla 10: Velocidades

Para las operaciones de avanzar hacia adelante o dar marcha atrás se ha elegido un valor medio alto de velocidad, sin embargo para realizar los giros con más precisión se introduce un valor de velocidad bastante más reducido. También podemos observar cómo se añade un quinto modo de avance-turbo a velocidad máxima.

Se procede a estudiar cómo quedaría el nuevo código:

```
//Como primer paso se definen los nuevos pines que se van a usar del microcontrolador, asegurándose de elegir dos pines del microcontrolador que sean capaces de generar una salida PWM
int motorDer1=7;      //El pin 7 a In1 del L298N
int motorDer2=6;      //El pin 6 a In2 del L298N
int motorIzq1=5;      //El pin 5 a In3 del L298N
int motorIzq2=4;      //El pin 4 a In4 del L298
int PWM_Derecho=10;       //El pin 10 de Arduino se conecta con el pin EnA del L298N
 int PWM_Izquierdo=11;      //El pin 11 de Arduino se conecta con el pin EnB del L298N
//Se declaran los pines como pines de salidas
void setup() {
pinMode(motorDer1, OUTPUT);
pinMode(motorDer2, OUTPUT);
pinMode(motorIzq1, OUTPUT);
pinMode(motorIzq2, OUTPUT);
pinMode(PWM_Derecho, OUTPUT);
pinMode(PWM_Izquierdo, OUTPUT);
}
//En este punto tan solo falta añadir las órdenes pertinentes en cada una de las subrutinas que se habían definido. Los valores de la velocidad pueden oscilar entre 0 y 255 (8 bits)
void atras()
{
digitalWrite(motorDer1,HIGH);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,HIGH);
digitalWrite(motorIzq2,LOW);
analogWrite(PWM_Derecho,200);       //Velocidad motor derecho 200
analogWrite(PWM_Izquierdo,200);     // Velocidad motor izquierdo 200
}
```

```
  void adelante(){
  digitalWrite(motorDer1,LOW);
  digitalWrite(motorDer2,HIGH);
  digitalWrite(motorIzq1,LOW);
  digitalWrite(motorIzq2,HIGH);
  analogWrite(PWM_Derecho,200);          //Velocidad motor derecho 200
  analogWrite(PWM_Izquierdo,200);        //Velocidad motor izquierdo 200
}
void giraDerecha(){
digitalWrite(motorDer1,HIGH);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);
analogWrite(PWM_Derecho,170);          //Velocidad motor derecho170
analogWrite(PWM_Izquierdo,170);        //Velocidad motor izquierdo 170
}
void giraIzquierda(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,HIGH);
digitalWrite(motorIzq1,HIGH);
digitalWrite(motorIzq2,LOW);
analogWrite(PWM_Derecho,170);          //Velocidad motor derecho 170
analogWrite(PWM_Izquierdo,170);        //Velocidad motor izquierdo 170
}
void parar(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,LOW);
}
```

```
void adelanteTurbo(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,HIGH);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);
analogWrite(PWM_Derecho,255);        //Velocidad motor derecho 255
analogWrite(PWM_Izquierdo,255);     //Velocidad motor izquierdo 255
}
//  Ya definidas todas las funciones auxiliares se pasa al bucle principal del código.
void loop() {
adelante();
delay(3000);
atras();
delay(3000);
giraDerecha();
delay(3000);
giraIzquierda();
delay(3000);
adelanteTurbo();
}
```

- **Módulo HCSR-04 (Sensor ultrasónico)**

-Antes de meterse de lleno con el código se profundiza algo más en la manera de funcionar del componente.

En el momento que se aplica tensión al pin 6 desde Arduino el sensor de ultrasonidos emitirá un sonido de 8 pulsos con una frecuencia 40 KHz, se mantendrá activo durante 10 µS, que es el tiempo de disparo para activar el pin "Trig". Esta onda de sonido debe ser recibida por el receptor del sensor, después de calcular el tiempo que ha tardado la onda se emite una señal a través de Echo hacia al pin 5 en Arduino.

Mediante "pulseIn(EchoPin,HIGH)" se activa el pin 5 en Arduino. Esta función realiza la medida y devuelve un valor en microsegundos en relación a la señal emitida por el pin "Echo" del sensor. Se procede a estudiar el código correspondiente:

```
// Configuración de los pines del sensor Trigger y Echo
const int PinTrig = 6;
const int PinEcho = 5;

const float VelSon = 34000.0; // Constante velocidad sonido en cm/s

float distancia;

void setup()

{

Serial.begin(9600); // Inicialización del monitor serie para mostrar los datos
pinMode(PinTrig, OUTPUT); // Configuración el pin Trig en como salida

pinMode(PinEcho, INPUT); // Configuración el pin Echo en modo entrada

}

void loop()

{

iniciarTrigger();

   unsigned long tiempo = pulseIn(PinEcho, HIGH); // La función pulseIn obtiene el tiempo que tarda en cambiar entre estados, en este caso a HIGH

distancia = tiempo * 0.000001 * VelSon / 2.0; // Se calcula la distancia en centímetros. Se convierte el tiempo en segundos (se trabaja en microsegundos) y por eso se multiplica por 0.000001

Serial.print("Distancia: ");

Serial.print("cm");

Serial.println();
```

```
delay(1000);
}
void iniciarTrigger() // Función que inicia la secuencia del Trigger para comenzar a medir
{
digitalWrite(PinTrig, LOW); // Se introduce un cero lógico en el Trigger (estado bajo) y se espera 2 ms
delayMicroseconds(2);
digitalWrite(PinTrig, HIGH); Se introduce un uno lógico en el Trigger (estado alto) y se espera 10 ms
delayMicroseconds(10);
digitalWrite(PinTrig, LOW); // Se vuelve a poner el Trigger en estado bajo hasta la siguiente medida.
}
```

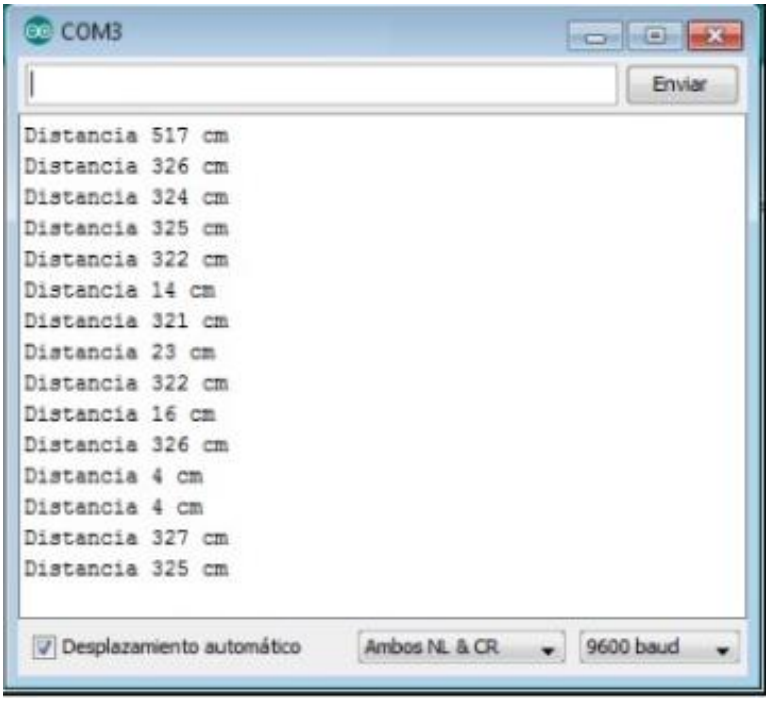


Figura 48: Monitor serie Arduino

Introducido el código se pueden ver reflejados los valores de distancia medidos en el monitor serie de Arduino.

- **Modo de operación esquiva obstáculos**

Uno de los objetivos secundarios del proyecto trata sobre conseguir una función esquiva obstáculos, capaz de dirigir el vehículo en piloto automático esquivando los objetos que se encuentre en su trayectoria. Para esta función se ha hecho uso del sensor ultrasónico ya mencionado.

Diagrama de flujo: Modo de operación esquiva obstáculos

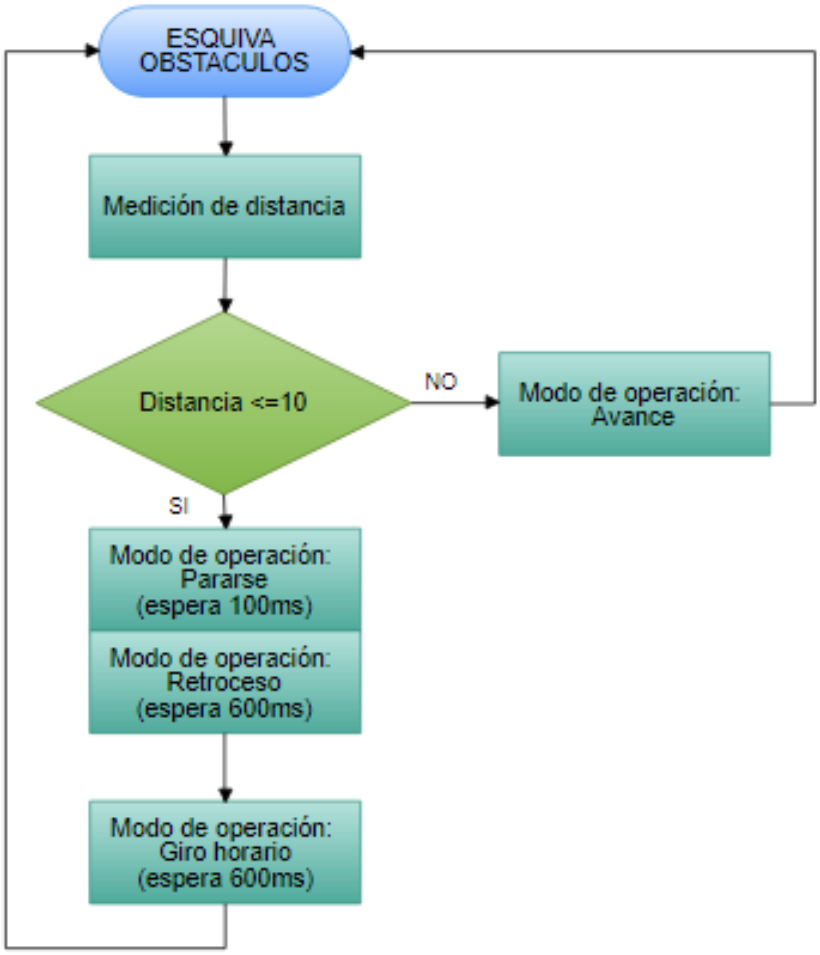


Diagrama 4: Modo de operación "Esquiva obstáculos"

Código Arduino: Modo de operación esquiva obstáculo

```
#include <SoftwareSerial.h>
int duracion, distancia;// Para calcular distancia
int motorDer1=7;            //El pin 7 a In1 del L298N
int motorDer2=6;            //El pin 6 a In2 del L298N
int motorIzq1=5;            //El pin 5 a In3 del L298N
int motorIzq2=4;            //El pin 4 a In4 del L298N
int PWM_Derecho=10;    //El pin 10 de Arduino se conecta con el pin EnA del L298N
int PWM_Izquierdo=11;  //El pin 11 de Arduino se conecta con el pin EnB del L298N
int pecho = 9; // Definición del pin 9 como (pecho) para el ultrasonido
int ptrig = 12; // Definición del pin 12 como (ptrig) para el ultrasonido

void setup(){  //Configuración de los pines como pines de entrada/salida
pinMode(motorDer1, OUTPUT);
pinMode(motorDer2, OUTPUT);
pinMode(motorIzq1, OUTPUT);
pinMode(motorIzq2, OUTPUT);
pinMode(PWM_Derecho, OUTPUT);
pinMode(PWM_Izquierdo, OUTPUT);
pinMode(pecho, INPUT);
pinMode(ptrig,OUTPUT);
}

/Comenzamos a definir las funciones que llamaremos desde nuestro bucle principal.
void EsquivaObstaculos(){
digitalWrite(ptrig, HIGH);  // Genera el pulso de Trigger por 10us
delay(0.01);
digitalWrite(ptrig, LOW);
duracion = pulseIn(pecho, HIGH);   // Lee el tiempo del Echo
distancia = (duracion/2) / 29; // Calcula la distancia en centímetros
delay(10);

if (distancia <= 10){     // Si la distancia es menor o igual a 10cm
digitalWrite(13,HIGH); // Enciende LED
digitalWrite(motorDer1,LOW);//Modo de operación: Pararse
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,LOW);
analogWrite(PWM_Derecho,170);          //Velocidad motor derecho 170
analogWrite(PWM_Izquierdo,170);        //Velocidad motor izquierdo 170
delay (200);
```

```
digitalWrite(motorDer1,HIGH); //Modo de operación: Retroceso
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,HIGH);
digitalWrite(motorIzq2,LOW);
analogWrite(PWM_Derecho,170);   //Velocidad motor derecho 170
analogWrite(PWM_Izquierdo,170); //Velocidad motor izquierdo 170
delay(600);

digitalWrite(motorDer1,HIGH);       //Modo de operación: Giro en sentido horario
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);
analogWrite(PWM_Derecho,170);    //Velocidad motor derecho 170
analogWrite(PWM_Izquierdo,170); //Velocidad motor izquierdo 170
delay(600);
digitalWrite(13,LOW);
}
    else{ // Si no hay obstáculos se desplaza al frente
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,HIGH);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);
analogWrite(PWM_Derecho,170);   //Velocidad motor derecho 170
analogWrite(PWM_Izquierdo,170); //Velocidad motor izquierdo 170
}
}
//Bucle principal
void loop() {
EsquivaObstaculos();
 }
```

Mediante este algoritmo se consigue una ejecución relativamente aceptable a la hora de esquivar cualquier tipo de obstáculo. El problema de este método es que tiene mucho margen de error, debido en parte a que cualquier tipo de objeto que se salga del margen de detección del sensor ultrasónico no será detectado.

Este primer código sirve como toma de contacto a la hora de desarrollar un algoritmo funcional capaz de detectar y esquivar todo tipo de objeto, de forma que se dé una fluidez de movimiento del vehículo, sin quedarse atascado o chocando con objetos situados en posibles puntos muertos del sensor de ultrasonidos.

Para solventar este problema se dispone del servo motor, el cual permitirá variar el ángulo de trayectoria de detección del sensor, consiguiendo de esta manera eliminar gran parte de los puntos muertos que se tuvieran en el ejemplo previo.

Diagrama de flujo: Modo de operación esquiva obstáculos (servomotor)

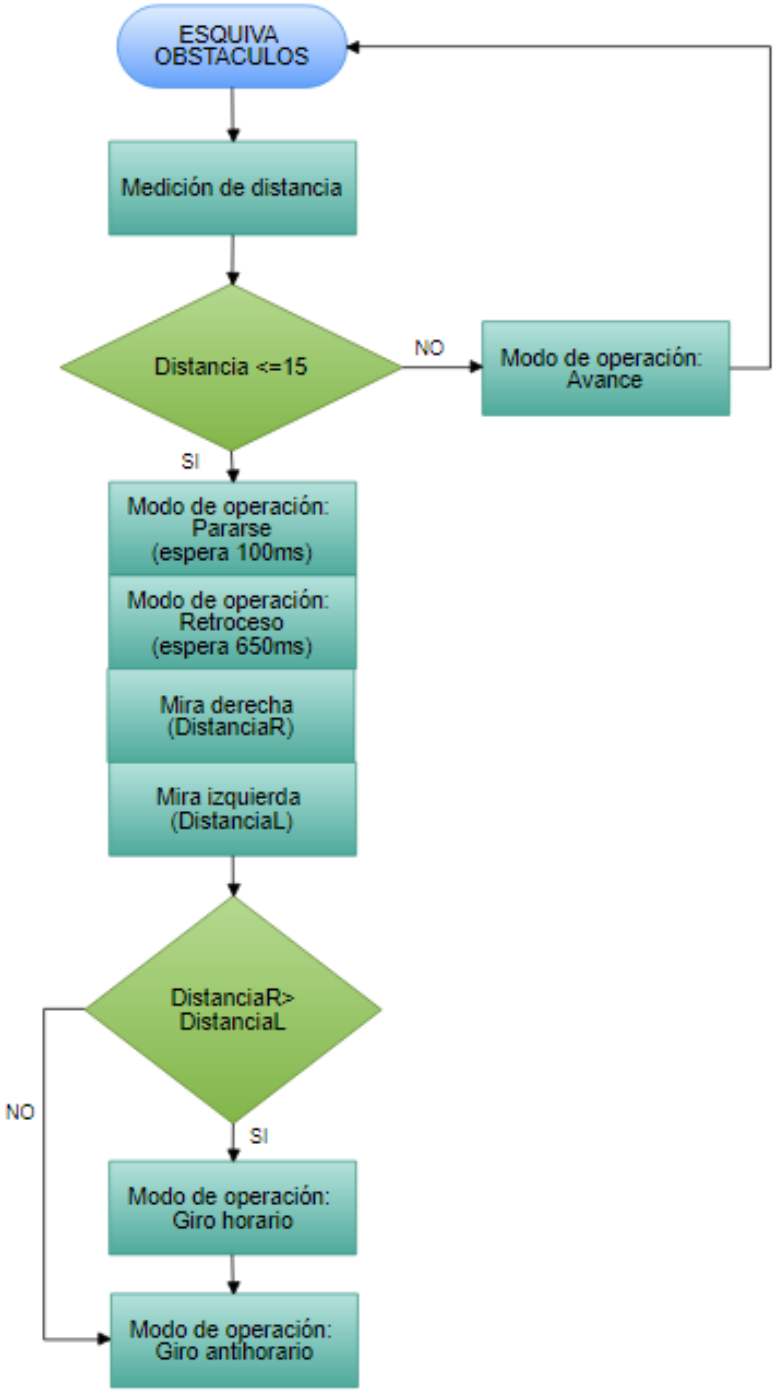


Diagrama 5: Modo de operación "Esquiva obstáculos (servomotor)"

- **Código Arduino**: Modo de operación esquiva obstáculo con servomotor

```
#include <SoftwareSerial.h>
#include <Servo.h>
SoftwareSerial Serial_2 (2, 4); // Nueva conexión serie- Pin2 (RX) a TX y Pin3(TX) a RX
Servo myservo;
int duracion,duracion1,duracion2, distancia; // Para Calcular distancia
int dist = 0;
int leftdist = 0;
int rightdist = 0;
int object = 14;     //Distancia de referencia
int motorDer1=7;       //El pin 7 a In1 del L298N
int motorDer2=6;       //El pin 6 a In2 del L298N
int motorIzq1=5;       //El pin 5 a In3 del L298N
int motorIzq2=10;      //El pin 10 a In4 del L298N
int PWM_Derecho=11;         //El pin 11 de Arduino se conecta con el pin EnA del L298N
int PWM_Izquierdo=3;         //El pin 3 de Arduino se conecta con el pin EnB del L298N
int pecho = 9; // Definición del pin 9 como (pecho) para el ultrasonido
int ptrig = 12; // Definición del pin 12 como (ptrig) para el ultrasonido
int servopin = 8; // Definición del pin 8 como para el servomotor

void setup(){   //Configuración de los pines como pines de entrada/salida
pinMode(motorDer1, OUTPUT);
pinMode(motorDer2, OUTPUT);
pinMode(motorIzq1, OUTPUT);
pinMode(motorIzq2, OUTPUT);
myservo.attach(servopin);
pinMode(PWM_Derecho, OUTPUT);
pinMode(PWM_Izquierdo, OUTPUT);
pinMode(pecho, INPUT);
pinMode(ptrig,OUTPUT);
delay(700);
}
```

```
void loop()  //Bucle principal del código
{
digitalWrite(ptrig, HIGH);   // Genera el pulso de Trigger por 10us
delay(0.01);
digitalWrite(ptrig, LOW);
duracion = pulseIn(pecho, HIGH);     // Lee el tiempo del Echo
dist = (duracion/2) / 29;        // calcula la distancia en centímetros
delay(100);
if(dist < object) {              //Si la distancia es menor a 14cm
BuscarRuta();                   //Buscar ruta
  }
 if(dist >= object) {           //Si la distancia es mayor o igual a 14cm
   forward();
  }
}
//Modo de operación: Avance
void forward(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,HIGH);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);
analogWrite(PWM_Derecho,160);   //Velocidad motor derecho 160
analogWrite(PWM_Izquierdo,160); //Velocidad motor izquierdo 160
return;
}
```

```
void BuscarRuta() {
  halt();                    //Modo de operación: Pararse
  backward();              //Modo de operación: Retroceso
  lookleft();//Modo de operación: Mira izquierda
  lookright();             //Modo de operación: Mira derecha

  if ( leftdist < rightdist )
  {
  turnright();    //Modo de operación: Gira derecha
  }
 else
 {  turnleft ();  //Modo de operación: Gira izquierda
 }
}
void backward(){       //Modo de operación: Retroceso
digitalWrite(motorDer1,HIGH);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,HIGH);
digitalWrite(motorIzq2,LOW);
analogWrite(PWM_Derecho,160);   //Velocidad motor derecho 160
analogWrite(PWM_Izquierdo,160); //Velocidad motor izquierdo 160
delay(600);
halt();
return;
}
void halt(){               //Modo de operación: Pararse
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,LOW);
delay(13);
return;
```

```
}
void lookleft() {         //Modo de operación: Mira izquierda
  myservo.write(180);
  delay(700);
  digitalWrite(ptrig, HIGH);   // Genera el pulso de Trigger por 10us
  delay(0.01);
  digitalWrite(ptrig, LOW);
  duracion1 = pulseIn(pecho, HIGH);    // Lee el tiempo del Echo
  leftdist = (duracion1/2) / 29;      // Calcula la distancia en centímetros
  myservo.write(90);
  delay(1000);
  return;
}
void lookright() {           //Modo de operación: Mira derecha
  myservo.write(0);
  delay(700);
  digitalWrite(ptrig, HIGH);  // Genera el pulso de Trigger por 10us
  delay(0.01);
 digitalWrite(ptrig, LOW);
 duracion2 = pulseIn(pecho, HIGH);     // Lee el tiempo del Echo
 rightdist = (duracion2/2) / 29;       // Calcula la distancia en centímetros
 myservo.write(90);
 delay(700);
 return;
}
```

```
//Modo de operación: Giro en sentido antihorario (giro izquierda)
void turnleft(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,HIGH);
digitalWrite(motorIzq1,HIGH);
digitalWrite(motorIzq2,LOW);
analogWrite(PWM_Derecho,250);   //Velocidad motor derecho 250
analogWrite(PWM_Izquierdo,250); //Velocidad motor izquierdo 250
delay(150);
return;
}
//Modo de operación: Giro en sentido horario  (giro derecha)
void turnright(){
digitalWrite(motorDer1,HIGH);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);
analogWrite(PWM_Derecho,250);   //Velocidad motor derecho 250
analogWrite(PWM_Izquierdo,250); //Velocidad motor izquierdo 250
delay(150);
return;
 }
```

Como se aprecia en el diagrama de flujo en el bucle principal se realiza una medida de distancia, donde en función de si es menor o mayor que la distancia fijada (14cm en este caso) como referencia el coche continuará avanzando o recurrirá a la función auxiliar de buscar ruta.

Dicha función detiene el vehículo, retrocede y mira a izquierda y derecha, comprobando que ruta tiene más margen de distancia para continuar y realiza un giro en esa dirección.

Una vez finalizada la función de buscar ruta se reinicia el bucle principal y vuelve a comenzar el proceso.

- **Código final: Control motores DC**

-Por último se expone el diagrama de flujo seguido de la programación correspondiente al control directo del vehículo teledirigido, es decir, se aplica el código correspondiente al control de los motores por medio del puente H, relacionándolo a su vez con los comandos pertinentes al envío y recepción de datos por medio del módulo bluetooth. De esta manera se controla fácilmente los diferentes modos de operación en función de los datos enviados.
Además se incluye el control directo del apagado o encendido de las luces de posición.
Diagrama de flujo: control motor DC

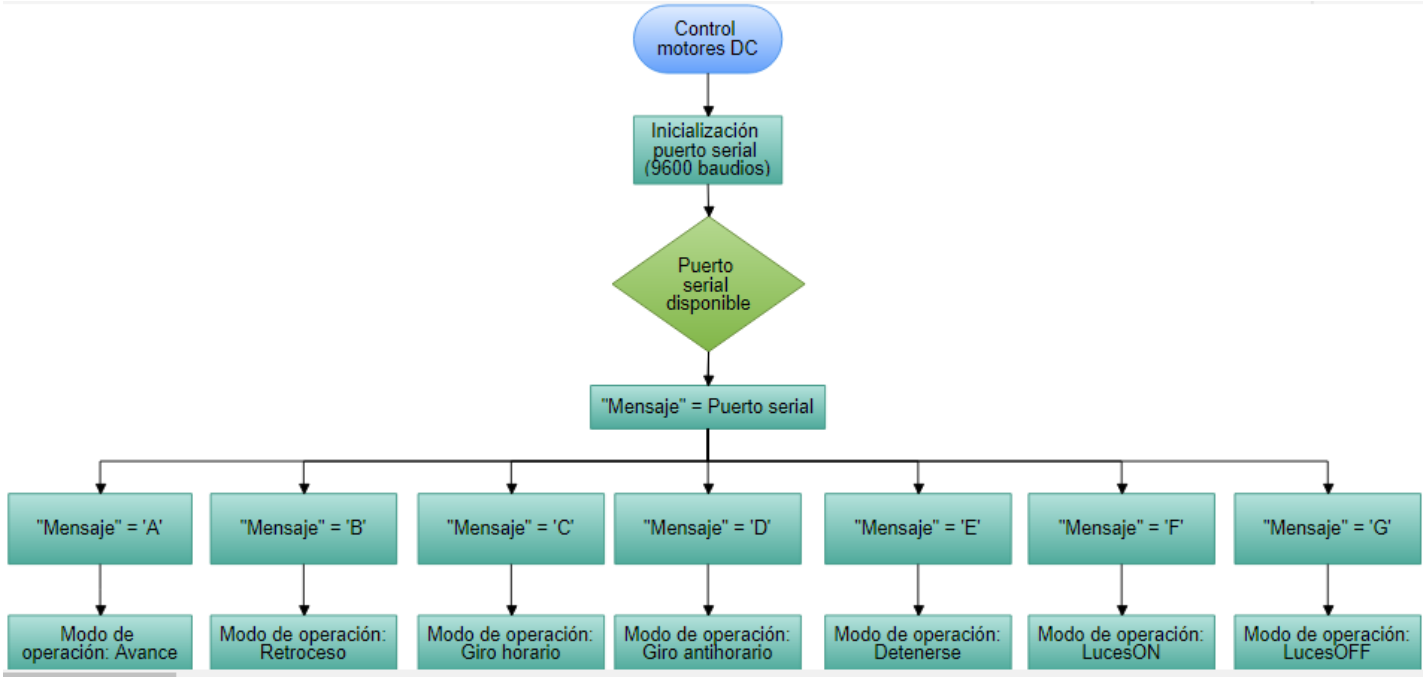


Diagrama de flujo 6: Control motores DC

Cada modo de operación es controlado desde el siguiente menú con las distintas opciones de modos de operación:

Figura 49: Menú de control

Código Arduino: Control motores DC

```
#include <SoftwareSerial.h>
SoftwareSerial Serial_2 (2, 3); // Crea una nueva conexión serie- Pin2(RX) a  TX y Pin3(TX) a RX
int  Mensaje=0;// Variable de cadena de caracteres para almacenar el mensaje
int motorDer1=7;                //El pin 7 a In1 del L298N
int motorDer2=6;                //El pin 6 a In2 del L298N
int motorIzq1=5;                //El pin 5 a In3 del L298N
int motorIzq2=10;               //El pin 10 a In10 del L298N
int PWM_Derecho=11;         //El pin 11 de Arduino se conecta con el pin EnA el L298N
int PWM_Izquierdo=3;        //El pin 3 de Arduino se conecta con el pin EnB el L298N
int luces=13;               //El pin13 se conecta a la alimentación de las luces de posición

void setup(){
Serial_2.begin(9600);
Serial.begin(9600);
pinMode(motorDer1, OUTPUT);     //Configuración de los como pines de salida
pinMode(motorDer2, OUTPUT);
pinMode(motorIzq1, OUTPUT);
pinMode(motorIzq2, OUTPUT);
 pinMode(PWM_Derecho, OUTPUT);
pinMode(PWM_Izquierdo, OUTPUT);
pinMode(luces,OUTPUT);
}
//Definición de las funciones auxiliares que serán llamadas desde el bucle principal
Modo de operación: Retroceso
void atras(){
digitalWrite(motorDer1,HIGH);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,HIGH);
digitalWrite(motorIzq2,LOW);
analogWrite(PWM_Derecho,200);   //Velocidad motor derecho 200
analogWrite(PWM_Izquierdo,200); //Velocidad motor izquierdo 200
}

//Modo de operación: Avance
void adelante(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,HIGH);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);
```

```
analogWrite(PWM_Derecho,220);   //Velocidad motor derecho220
analogWrite(PWM_Izquierdo,220); //Velocidad motor izquierdo 220
}
//Modo de operación: Giro en sentido horario.
void giraDerecha(){
digitalWrite(motorDer1,HIGH);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);
analogWrite(PWM_Derecho,255);   //Velocidad motor derecho 255
analogWrite(PWM_Izquierdo,255); //Velocidad motor izquierdo 255
}

//Modo de operación: Giro en sentido antihorario.
void giraIzquierda(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,HIGH);
digitalWrite(motorIzq1,HIGH);
digitalWrite(motorIzq2,LOW);
analogWrite(PWM_Derecho,255);   //Velocidad motor derecho 255
analogWrite(PWM_Izquierdo,255); //Velocidad motor izquierdo 255
}
//Modo de operación: Pararse
void parar(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,LOW);
}
//Modo de operación: luces On/Off
void lucesOn(){
digitalWrite(luces,HIGH);
}
void lucesOFF(){
digitalWrite(luces,LOW);
}
```

```
//Definidas las funciones auxiliares se programa el código principal que se repetirá en bucle
void loop() {
    if (Serial_2.available()>0) {
Mensaje = Serial_2.read();    }
      switch(Mensaje){
    case 'A':
       adelante();
        break;
         case 'B':
    atras();
        break;
       case 'C':
       giraDerecha();
    break;
         case 'D':
        giraIzquierda();
        break;
     case 'E':
        parar();
        break;
       case 'F':
        lucesOn();
        break;
     case 'G':
     lucesOFF();
    break;
    }
    }
```

- **Código final: Función esquiva obstáculos**

Se ha expuesto los algoritmos y códigos correspondientes al control de los motores, por medio del envío de datos (caracteres concretamente) mediante comunicación bluetooth, y por otra parte los pertinentes a la función de piloto automático, capaz de sortear esquivar objetos.
Por lo tanto solo queda pendiente poder acceder a ambos modos de operación de manera simultánea, es decir, sin tener que cargar dos códigos distintos en el microcontrolador para poder acceder de una función a otra. Además de esto se configura una última función para el control del encendido y apagado de las luces de posición.

Diagrama de flujo:

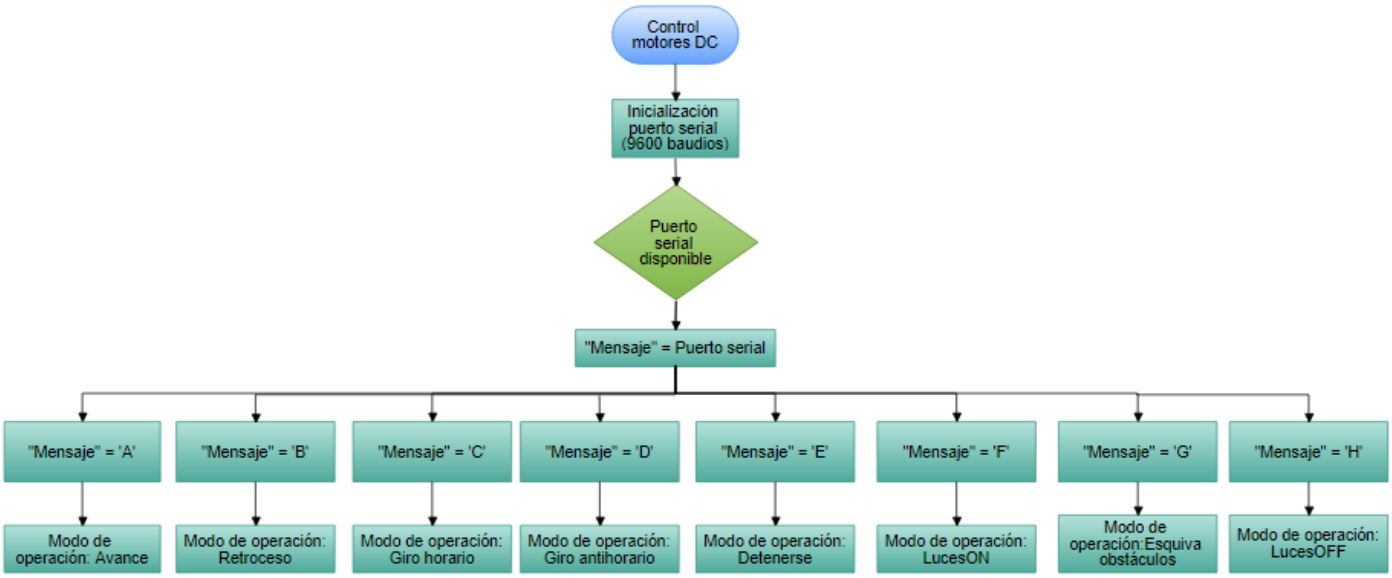


Diagrama de flujo 7: Control motores DC (final)

El modo de operación esquiva obstáculos ya se ha analizado previamente. Cada modo de operación es controlado desde el siguiente menú con las distintas opciones de modos de operación:

Figura 50: Menú de control (final)

Destacar que el apagado de las luces será necesario acceder al "modo terminal" de la aplicación bluetooth para poder introducir manualmente el carácter 'H'.

Código final: función esquiva obstáculos

```
//En primer lugar se introducen las librerías que facilitan el manejo y programación de los componentes
#include <SoftwareSerial.h>//Permite la comunicación serie en otros pines digitales además del 0 y el 1 (pines de comunicación serie por defecto)
#include <NewPing.h>          //Librería para facilitar el uso del sensor ultrasónico
#include <Servo.h>    //Librería para facilitar el uso del servo motor

#define TRIG_PIN A0//Define el pin A0 para vincularlo al pin Trigger del sensor ultrasónico
#define ECHO_PIN A1//Define el pin A1 para vincularlo al pin Trigger del sensor ultrasónico
#define MAX_DISTANCE 300//Define la distancia máxima medible por el sensor (en el caso de no especificarlo son 500cm por defecto)

NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);  //Mediante este comando asociamos los pines definidos previamente al sensor y la distancia máxima de sensado
Servo myservo;//Incluye el uso de servomotor
SoftwareSerial Serial_2 (2, 4);   // Crea nueva conexión serie- Pin2(RX) a TX y Pin4(TX) a RX
int distance;    //Variable para medir distancias
int  Mensaje=0;              // Variable de cadena de caracteres para almacenar el mensaje
int motorDer1=7;      //El pin 7 a In1 del L298N
int motorDer2=6;      //El pin 6 a In2 del L298N
int motorIzq1=5;      //El pin 5 a In3 del L298N
int motorIzq2=10;     //El pin 10 a In4 del L298N
int PWM_Derecho=11;    //El pin 11 de Arduino se conecta con el pin EnB del L298N
int PWM_Izquierdo=3;    //El pin 3 de Arduino se conecta con el pin EnA del L298N
int luces=13;                //El pin13 se conecta a la alimentación de las luces de posición
```

```
int readPing() {          //Declaración de función para llevar a cabo el sensado
  delay(70);
  int cm = sonar.ping_cm(); //Devuelve la distancia en centímetros
if(cm==0)
  {    cm = 250;  }
  return cm;  }
int lookRight()           //Función para medir la distancia a la derecha
{    myservo.write(12);   //Servomotor a 12º
    delay(300);
int distance = readPing();       //Calcula la distancia en centímetros
delay(100);
  myservo.write(90);      //Servomotor a 90º
return distance;}//Devuelve el valor de la distancia medida

int lookLeft()            //Función para medir la distancia a la izquierda
{    myservo.write(160);        //Servomotor a 160º
    delay(300);
int distance = readPing();       //Calcula la distancia en centímetros
delay(100);
 myservo.write(90);    //Servomotor a 90º
return distance;          //Devuelve el valor de la distancia medida
    delay(100);}

void setup(){             //Configuramos los pines de  Arduino como pines de salida
pinMode(motorDer1, OUTPUT);
pinMode(motorDer2, OUTPUT);
pinMode(motorIzq1, OUTPUT);
pinMode(motorIzq2, OUTPUT);
pinMode(PWM_Derecho, OUTPUT);
pinMode(PWM_Izquierdo, OUTPUT);
myservo.attach(9);  //Asigna el pin 9 de Arduino para el control de la posición del servomotor
myservo.write(90);  //Servomotor a 90º
```

```
delay(2000);
distance = readPing();          //Mide y almacena el valor de la medida en la variable
delay(100);
Serial_2.begin(9600);           //Inicia puerto serial a 9600 baudios
}
//Comenzamos a definir las funciones auxiliares
//Modo de operación: Retroceso
void atras(){
digitalWrite(motorDer1,HIGH);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,HIGH);
digitalWrite(motorIzq2,LOW);
analogWrite(PWM_Derecho,200); //Velocidad motor derecho 200
analogWrite(PWM_Izquierdo,200); //Velocidad motor izquierdo 200
}
//Modo de operación: Avance
void adelante(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,HIGH);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);
analogWrite(PWM_Derecho,210);//Velocidad motor derecho 210
analogWrite(PWM_Izquierdo,210);//Velocidad motor izquierdo 210
}
//Modo de operación: Giro en sentido horario
void giraDerecha(){
digitalWrite(motorDer1,HIGH);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,HIGH);
analogWrite(PWM_Derecho,255);//Velocidad motor derecho 255
analogWrite(PWM_Izquierdo,255);//Velocidad motor izquierdo 255
```

```
}
//Modo de operación: Giro en sentido antihorario
void giraIzquierda(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,HIGH);
digitalWrite(motorIzq1,HIGH);
digitalWrite(motorIzq2,LOW);
analogWrite(PWM_Derecho,255); //Velocidad motor derecho 255
analogWrite(PWM_Izquierdo,255); //Velocidad motor izquierdo 255
}
//Modo de operación: Pararse
void parar(){
digitalWrite(motorDer1,LOW);
digitalWrite(motorDer2,LOW);
digitalWrite(motorIzq1,LOW);
digitalWrite(motorIzq2,LOW);
}
//Modo de operación: Esquiva obstáculos
void EsquivaObstaculos() {
int distanceR = 0;       //Variable para almacenar distancia a la derecha
int distanceL = 0; //Variable para almacenar distancia a la izquierda
distance = readPing(); //Mide y almacena el valor de la medida en la variable
delay(40);
 if(distance<=15)
 { parar();
  delay(100);
atras();
  delay(650);
 parar();
  delay(100);
  distanceR = lookRight(); //Mide y almacena el valor de la distancia a la derecha
  delay(200);
```

```
  distanceL = lookLeft(); //Mide y almacena el valor de la distancia a la izquierda
  delay(100);
  if(distanceR>=distanceL)
  {
giraDerecha();
delay (650);
parar();
}
else
  {
giraIzquierda();
delay (650);
parar();
}
}
else
{
adelante();
}
 distance = readPing();
}
//Modo de operación: Control de luces de posición
void lucesON(){
digitalWrite(luces,HIGH);
}
void lucesOFF(){
digitalWrite(luces,LOW);
}
```

```
//Una vez definidas las funciones auxiliares pasamos al bucle principal del programa
void loop() {
if (Serial_2.available()>0) {
Mensaje = Serial_2.read();    }
  switch(Mensaje){
case 'A':
   adelante();
    break;
      case 'B':
atras();
    break;
   case 'C':
   giraDerecha();
break;
     case 'D':
    giraIzquierda();
    break;
 case 'E':
    parar();
    break;
   case 'F':
lucesON();
   break;
   case 'G':
EsquivaObstaculos() ;
   break;
  case ‘H’:
lucesOFF();
   break;
}
 }
```

3. RESULTADOS

-A continuación se muestra el esquema y una tabla final correspondiente a las conexiones finales del proyecto. Finalmente se dispone de tres fuentes de alimentación independientes, una destinada a la alimentación del puente en H y los motores, otra fuente se encarga de proporcionar alimentación al Arduino y sensor ultrasónico y la tercera fuente alimenta el servomotor, el módulo bluetooth y el regulador de tensión. Para el control de estas tres fuentes de alimentación se habilitan tres interruptores acoplados al chasis del vehículo.

- Esquema de conexiones:

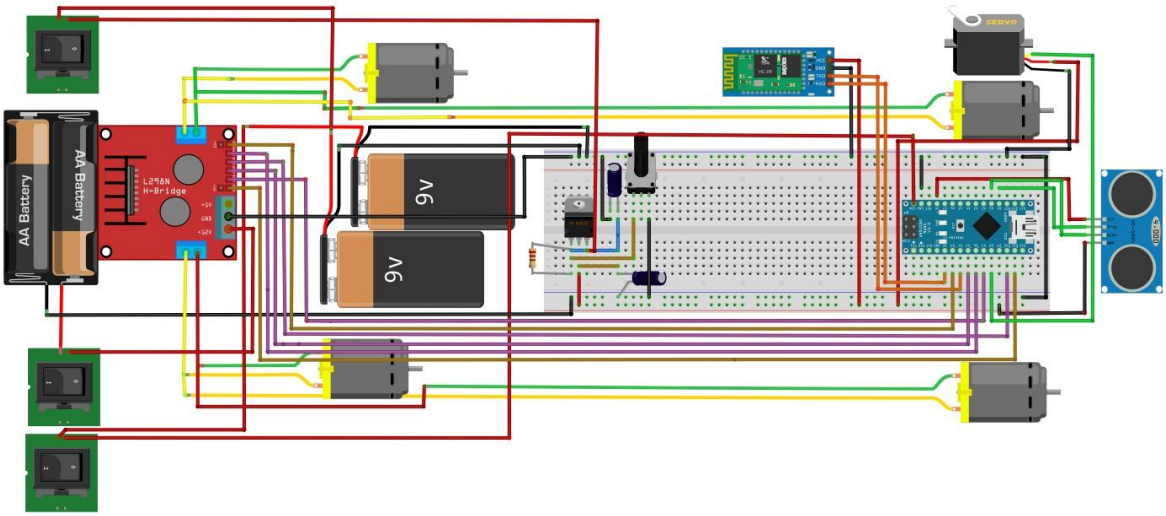


Figura51: Esquema de conexiones final

- Tabla de conexiones:

Microcontrolador Arduino		
Pin D2	Pin RX	Módulo Bluetooth
Pin D3	EnA	Puente en H
Pin D4	Pin TX	Módulo Bluetooth
Pin D5	In3	Puente en H
Pin D6	In2	Puente en H
Pin D7	In1	Puente en H
Pin D9	Pin 3 Servo	Servomotor
Pin A1	Pin Echo	Sensor ultrasonidos
Pin D10	In4	Puente en H
Pin D11	EnB	Puente en H
Pin A0	Pin Trig	Sensor ultrasonidos

Tabla 11: Tabla de conexiones final

La presentación del resultado final del proyecto queda de la siguiente forma:

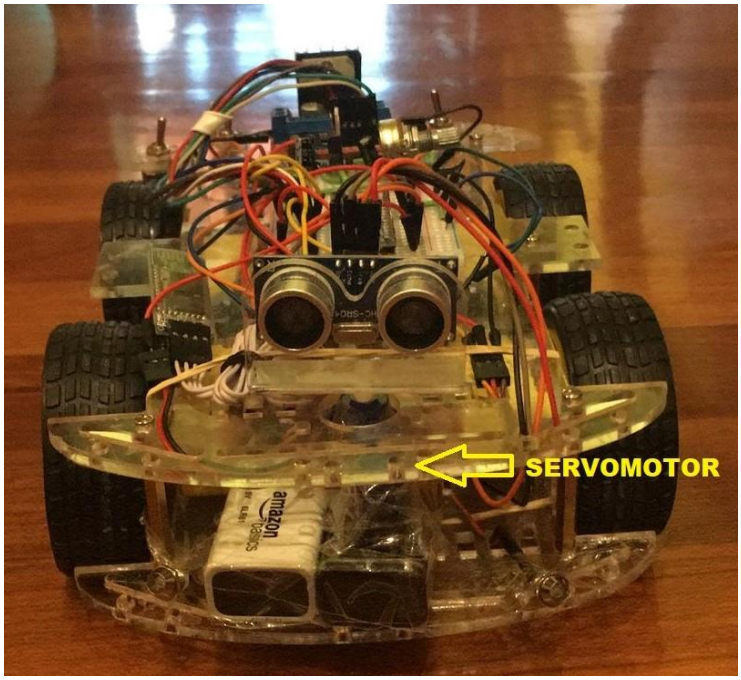


Figura 52: vista frontal

Se expone a continuación una vista lateral del vehículo junto con el despiece de los componentes principales señalizado.

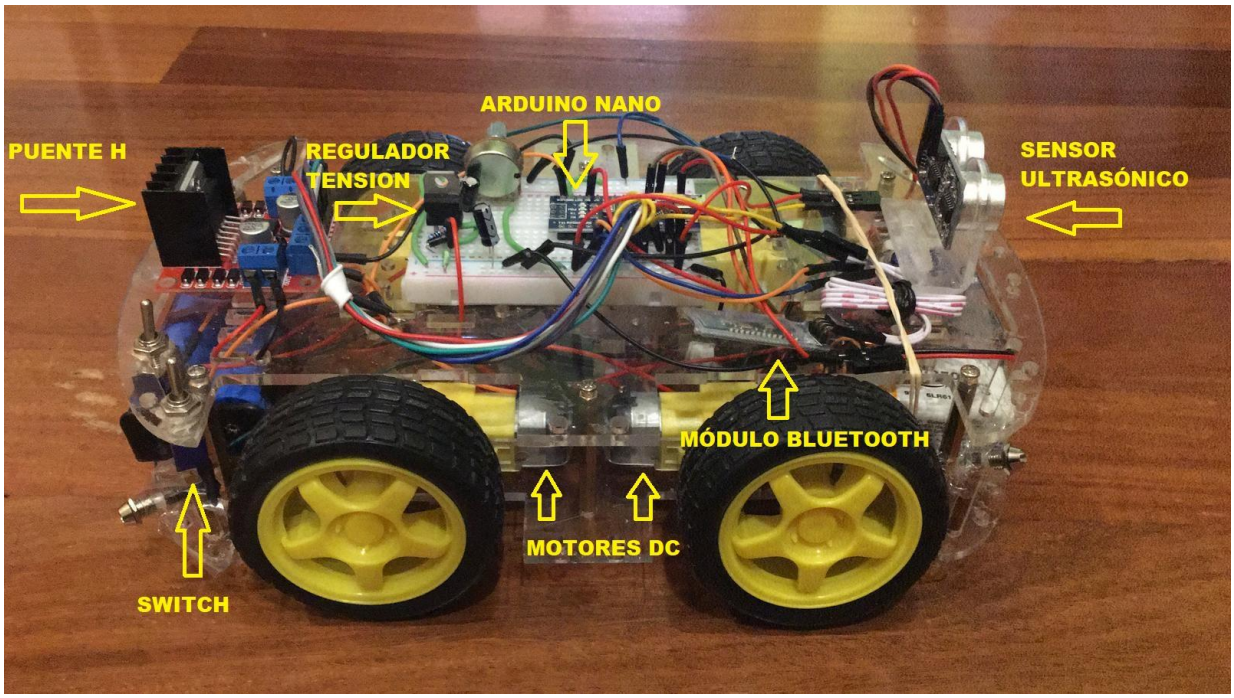


Figura 53: vista lateral

Para apreciar de forma directa su funcionamiento se expone el siguiente video como muestra del funcionamiento: *https://www.youtube.com/watch?v=JhmOrO04iDU*

3.1 PROBLEMAS ENCONTRADOS

-Durante la realización del proyecto han ido surgiendo cierto de tipo de inconvenientes y problemas que han requerido un proceso de aprendizaje continuo a la hora de solventarlos. A nivel teórico es relativamente sencillo entender un proyecto cualquiera, sin embargo, es en la práctica donde de verdad reside el verdadero desafío, puesto que se tienen en cuenta una serie factores que no juegan un papel importante en la teoría pero sí a la hora de observar los resultados prácticos reales.
En esta sección se resume brevemente los inconvenientes más importantes encontrados durante la implementación de este proyecto. Se pueden clasificar en dos categorías de problemas encontrados, los debidos a un fallo de la programación introducida (software) y los surgidos en base a una mala alimentación de los componentes utilizados (hardware).

En cuanto a los problemas relacionados con el código introducido (software) destacar los comandos relacionados con la recepción de los caracteres introducidos desde el smartphone para saltar entre los distintos modos de operación.

La recepción de los caracteres enviados no ha supuesto problema, pero sí el salto entre distintos modos de operación en función de estos datos. Hasta averiguar el funcionamiento del comando "switch()", para poder ir saltando de un modo de operación a otro, se intentó conseguir la misma funcionalidad a través de funciones condicionales, sin tener éxito alguno.
En referencia a los inconvenientes surgidos debido a un mal uso o implementación de los componentes (hardware), se pueden reducir a una mala alimentación de cada uno de estos. Al tratarse de un proyecto con varios componentes se precisa mínimo de dos fuentes de alimentación independientes. Esto se debe en parte a que solo los cuatro motores requieren una demanda alta de corriente para poder funcionar a máxima potencia. Es por esto donde se encuentra el primer error, intentando alimentar dichos motores mediante una pila de 9V, siendo esta incapaz de entregar la intensidad requerida. El resultado de esto se refleja en la potencia que son capaz de suministrar los motores, siendo tan insuficiente que el vehículo no es capaz ni de realizar giros debidamente. Por ello para alimentación de los motores se emplean dos pilas de 3,7V y 2200mA.
En cuanto al siguiente inconveniente es el resultado de intentar alimentar al Arduino junto con el resto de componentes al mismo tiempo, con una sola pila de 9V. El servomotor debido a la demanda de corriente que conlleva es aconsejable alimentarlo desde una fuente independiente a la de Arduino, sumado a la alimentación del sensor ultrasónico, módulo bluetooth, regulador de tensión y luces todo esto resulta en un funcionamiento poco eficiente. Estos inconvenientes se traducen en constantes conexiones y desconexiones del módulo bluetooth, problemas de medición con el sensor ultrasónico y el servomotor actuando de manera más lenta o directamente no funcionando.

Por todo ello para alimentar el servomotor y el módulo bluetooth (componentes en los que se ha encontrado más problemas) se ha empleado otra pila de 9V que serán regulados a 5V para ajustarse a las especificaciones de estos. El sensor ultrasónico se podría alimentar de esta misma fuente de tensión, sin embargo, se aprecia un mejor funcionamiento alimentándolo desde la salida de 5V del microcontrolador.
Para finalizar, cabe la pena destacar que a medida que se van descargando las pilas de 9V se vuelven a apreciar el mismo tipo de fallos en los componentes. Por lo tanto es imprescindible disponer de pilas recargables o nuevas para asegurar un buen funcionamiento.

3.2 CONCLUSIONES

-Para llevar a cabo la realización de este proyecto han sido necesarios parte de los conocimientos adquiridos a lo largo de toda la formación universitaria. Sin embargo, las nociones sobre electrónica digital y analógica son claramente las más empleadas a nivel práctico, todo esto unido a una base previa de conocimientos sobre programación. Mediante la suma de estos se ha hecho posible la realización del proyecto.
Una buena forma de valorar los resultados de un proyecto es remitirse a los objetivos marcados en un principio e intentar realizar una valoración objetiva.
En cuanto al primero de ellos se puede afirmar que se ha conseguido un buen entendimiento del funcionamiento de todos los componentes hardware utilizados, además de un buen control y manejo del microcontrolador empleado, descubriendo y aplicando las distintas herramientas de las que dispone.
Otro de los objetivos propuestos era obtener un control total del vehículo pudiendo manejarlo de forma fluida, objetivo que se puede dar por alcanzado, puesto que mediante el control de la velocidad de los distintos modos de operación se ajusta la conducción a las necesidades del entorno (dependiendo del terreno se precisará de más potencia de los motores para manejarse correctamente).
En referencia a los objetivos secundarios, se ha conseguido implementar un algoritmo capaz de esquivar todo tipo de obstáculos en conducción automática, además de haber añadido luces de posición (controladas desde el smartphone).
En cuanto a posibles aspectos a mejorar, la presentación por ejemplo, se podría mejorar la imagen añadiendo una carrocería o usando otro tipo de materiales comerciales para conseguir una distribución de los componentes más compacta. Con respecto al software, se podría mejorar el algoritmo de la función esquiva obstáculos para ser más eficiente, tomando más medidas de distancia, y conseguir de esta forma una conducción más fluida.

3.3 PRESUPUESTO

-Se expone la lista de materiales y componentes adquiridos para la implementación del proyecto realizado. Cierto tipo de componentes específicos, como los condensadores o los diodos, se han adquirido en lotes de varias unidades (de 20 unidades en el caso de los condensadores) puesto que es la forma usual de comercializarlos por medio de plataformas online. El resto de componentes se adquieren por unidades. Componentes como el smartphone o las resistencias (regulador de tensión) no se incluyen es las tablas debido a que ya se disponía de ellos. El gasto total se divide entre dos secciones, una para los componentes principales y otra para los auxiliares o secundarios.

COMPONENTES PRINCIPALES	Nº UNIDADES	PRECIO
HC-06 Wireless Bluetooth Transceiver RF Master Module	1	4.04€
High Performance DIY Nano V3.0 Atmel Atmega328P Mini USB Development Board for Arduino	1	5.90€
L298N H-Bridge Stepper Motor Driver Module for Arduino	1	2.23€
HC - SR04 Arduino Ultrasonic Distance Measuring Sensor Module Measure Detector with Transparent MountedHode	1	3.43€
5X Regulador de Tensión Lineal LM317T LM317 Regulador de Voltaje Ajustable	10	2.34€
Mini SG90 servo micro motor p		13.22€
ViewTek CR0031-4WD Kit de chasis Coche Robot Inteligente Arduino	1	29.99€
	TOTAL	61.15€

Tabla 12: Presupuestos principales

COMPONENTES SECUNDARIOS	Nº UNIDADES	PRECIO
Condensadores electrolíticos 10uF 50v ±20% 4x8mm	20	0,95 €
Condensadores electrolíticos 22uF 50v ±20% 6x8mm	20	1,13 €
Diodo rectificador 1N4007 1000v 1A	20	0,76 €
Mini SG90 servo micro motor		13,22€
Dupont Male to Female Jumper Wire 40PCS	1	2.01€
High Performance 400 Tie Points Solderless Breadboard for DIY(protoboard)	1	1.86€
Zanflare Cargador C2 Inteligente, Pantalla LCD Cargador de Pilas Universal Rápido, y para Baterías Recargables Ni-MH 18650 26650 26500	1	16,99€
Dailyinshop 4pcs RC Drift Car 3 mm 3 mm Rojo y Negro Faros Faros para Coche Modelo del RC	1	2,99€
Aussel 10 piezas AC 125V 6A ON-ON 3 pines 2 posiciones Mini interruptor de palanca para Arduino	10	7,99€
18650 Batería de litio recargable de 2200 mAh ICR18650 3,7 V 2 piezas con 2 x 18650 caja de almacenamiento de batería	2	13,19€
	TOTAL	61.09€

Tabla 13: Presupuesto secundarios

Calculando el total resultante entre componentes principales y secundarios se obtiene un **presupuesto de 122,24€** llevado a cabo para la implementación del proyecto. Mencionar que no se ha tenido en cuenta el número de horas trabajado, sin embargo, este es un factor a tener en consideración en cualquier tipo de proyecto que se lleve a cabo.

3.4 DIAGRAMA GANTT

-Para reflejar las fases que se han seguido para completar el proyecto se elabora un diagrama de Gantt, donde se refleja el tiempo empleado para cada tarea y el orden en el que se han ejecutado.

Proyecto	Fecha inicio prevista	Dias trabajados	Fecha final prevista
Documentación del proyecto	7-sep-18	142	1-feb-19
Estudio de los resultados	14-ene-19	13	28-ene-19
Ajuste del software	5-ene-19	7	12-ene-19
Pruebas prácticas	3-ene-19	9	12-ene-19
Estudio y análisis del software	5-nov-18	53	28-dic-18
Montaje de la estructura hardware	20-oct-18	56	15-dic-18
Análisis práctico de componentes	29-oct-18	18	16-nov-18
Análisis teórico de componentes	1-oct-18	27	28-oct-18
Elección y compra de componentes	20-sep-18	10	30-sep-18
Recopilación de información	7-sep-18	21	28-sep-18
Elección del proyecto	1-sep-18	6	7-sep-18

Tabla 14: Fases proyecto

En el diagrama de Gantt expuesto se puede observar la duración de cada fase del proyecto. Cabe destacar como se solapan muchas de las fases, como por ejemplo la documentación del proyecto, siendo la tarea que más trabajo ha requerido y abarcando desde el inicio de este hasta su final.

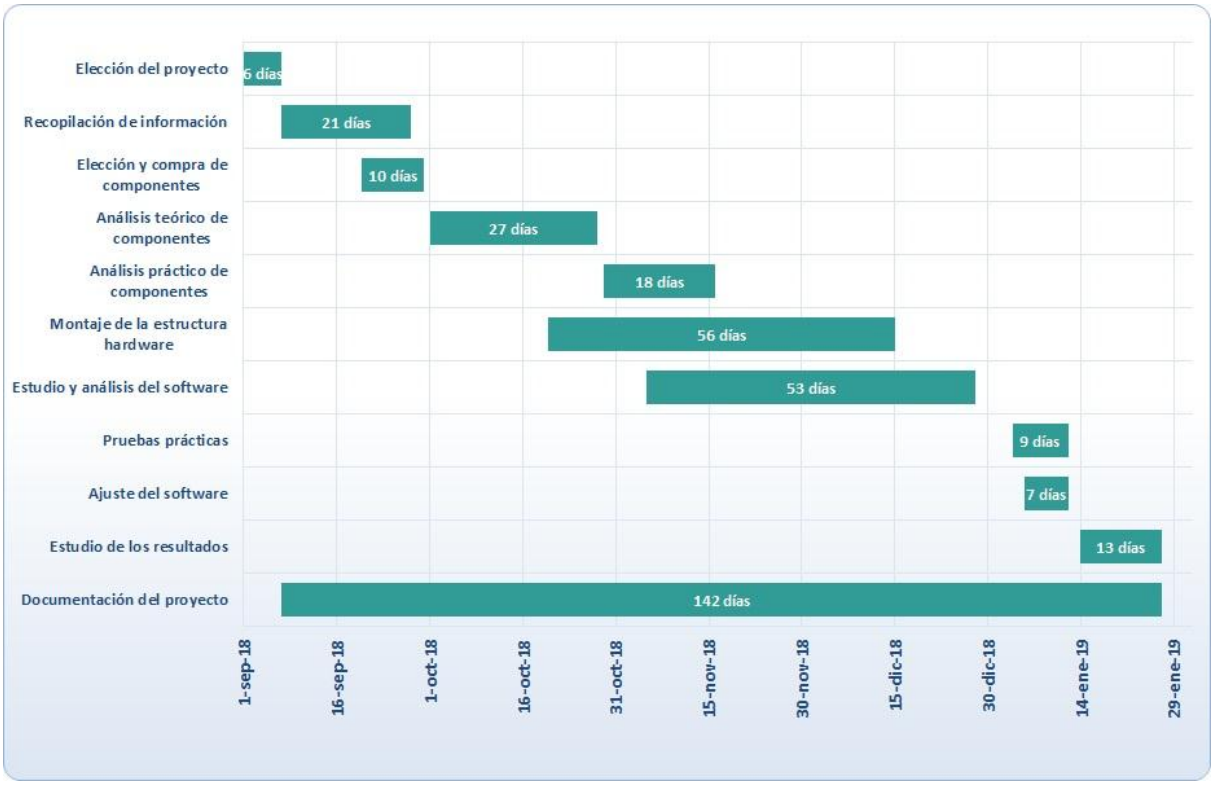


Diagrama de flujo 8: Diagrama Gantt

Mencionar que en el caso del "Montaje de la estructura hardware" ha requerido un periodo amplio de tiempo debido a que se ha ido realizando a medida que se avanzaba junto con otras tareas, como el "Análisis práctico de los componentes" y el "Estudio de la arquitectura software".

3.5AMPLIACIONES DEL PROYECTO

-En esta sección se analizan las distintas variantes del proyecto más comunes, haciendo referencia al tipo de comunicación o control empleado para el manejo del vehículo.

- **Coche controlado por radiofrecuencia**

[11] Se trata de la forma más clásica y tradicional de implementar el control de un vehículo de control remoto. Para establecer la conexión se necesita un módulo emisor y otro receptor de radiofrecuencia.

Figura54: Coche controlado por radiofrecuencia

Uno de los módulos más empleados (especialmente con Arduino) es el RF 433 MHz.
Los módulos de radio frecuencia RF 433MHz son transmisores/receptores inalámbricos que podemos emplear como forma de comunicación entre procesadores.

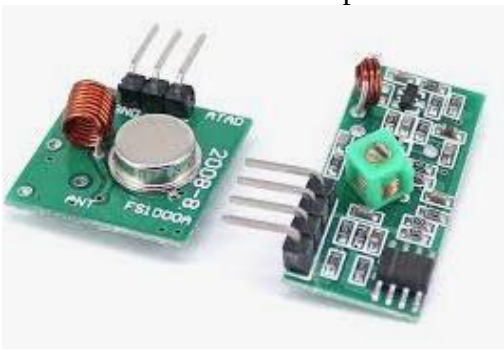


Figura 55: Emisor y receptor RF 433MHz

La frecuencia de operación es de 433MHz, no obstante, existen también módulos similares a 315MHz. **[12]** La comunicación es por un único canal y unidireccional (simplex) y tienen baja velocidad de transmisión (típicamente 2400bps), se realiza por modulación ASK (amplitude shift keying). No disponen de filtro ni ID por hardware, por lo que si queremos una comunicación robusta tendremos que implementarlo por software.
Resultan muy útiles en aplicaciones sencillas que no requieran una comunicación bidireccional. Se emplean por ello muy frecuentemente en proyectos caseros de electrónica y robótica, debido a su bajo precio y medio-largo alcance.

- **Vehículo de control remoto por infrarrojos**

Una de las formas más habituales de establecer el control remoto de un robot es por un mando mediante infrarrojos.

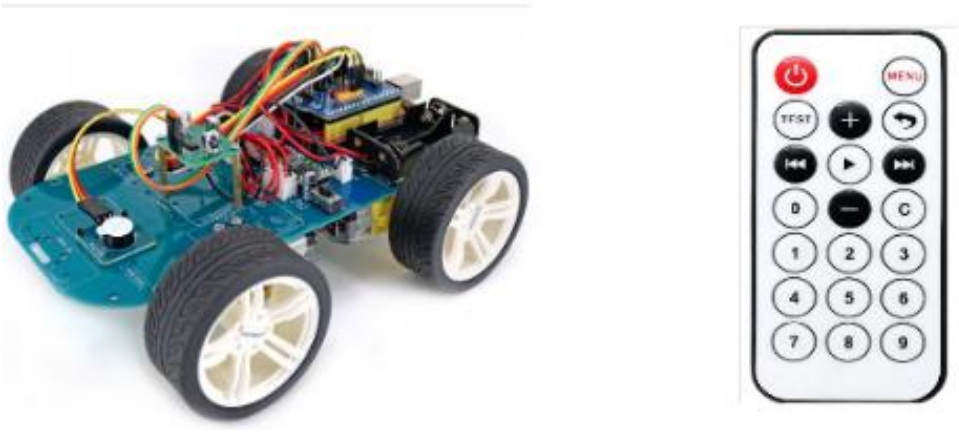


Figura 56: Coche controlado por infrarrojos

[13]Un mando a distancia es un dispositivo de control que emplea un LED infrarrojo para enviar una señal a un receptor. La señal puede ser detectada para controlar un autómata o un procesador como Arduino en este caso. Como módulo receptor de infrarrojos en el caso de Arduino se suele utilizar el siguiente:

Figura 57: Módulo Receptor de Infrarrojos Compatible con Arduino

Este sensor incorpora un filtro interno para detectar solo frecuencias de infrarrojos cercanas a 38KHz, por lo tanto es compatible con la mayoría de mandos infrarrojos. Dispone 3 pines de conexión GND, VCC y DATA, el cual nos permite conectar directamente a un pin digital de nuestro Arduino.

Para establecer la comunicación no existe un único protocolo adoptado como estándar. En su lugar cada fabricante ha desarrollado los suyos propios (RC-5 y RC-6 de Philips, el SIRC de Sony, el protocolo NEC de Nippon Electronic Company)

La información nunca se envía directamente como un pulso, si no que se envía modulada sobre una onda portadora. Esto repercute en una mejora para el rechazo al ruido y a la luz ambiental.

El protocolo más común empleado con Arduino es el protocolo NEC, que emplea una onda portadora de 38 kHz y modulación por distancia de pulsos (PDM Pulse Distance Modulation).

3.6 ANEXOS

- Anexos I: MICROTONTROLADOR ARDUINO

-En esta sección se exponen las especificaciones técnicas para los tipos de microcontrolador Arduino más empleados mencionados previamente, además de la distribución de pines correspondiente al tipo empleado para la realización del proyecto.

Las características principales del **Arduino Uno** son las siguientes:

Microcontrolador	Atmel Atmega328P
Tensión de funcionamiento	5 V
Tensión de entrada (recomendada)	7-12 V
Tensión de entrada (mínimo y máximo)	6-20 V
Pines digitales E/S	14 (6 de los cuáles para generar PWM)
Pines analógicos	60
Corriente DC por pin	40mA
SRAM	2 KB
EEPROM	1 KB
Velocidad del reloj	16 MHz
Memoria Flash	32 KB
Dimensiones	80mm x 5,5mm

Tabla 15: Especificaciones Arduino Uno

Las características principales del **Arduino Mega** son las siguientes:

Microcontrolador	Atmel Atmega2560
Tensión de funcionamiento	5 V
Tensión de entrada (recomendada)	7-12 V
Tensión de entrada (mínimo y máximo)	6-20 V
Pines digitales E/S	54 (15 de los cuáles para generar PWM)
Pines analógicos	16
Corriente DC por pin	20mA
SRAM	8 KB
EEPROM	4 KB
Velocidad del reloj	16 MHz
Memoria Flash	32 KB
Dimensiones	101.5mm x 53.3mm

Tabla 16: Especificaciones Arduino Mega

Las características principales del **Arduino Leonardo** son las siguientes:

Microcontrolador	ATmega32u4
Tensión de funcionamiento	5 V
Tensión de entrada (recomendada)	7-12 V
Pines digitales E/S	20 (7 de los cuáles para generar PWM)
Pines de entrada analógicos	12
Corriente máxima DC por pin	40mA
SRAM	2.5 KB
EEPROM	1 KB
Velocidad del reloj	16 MHz
Memoria Flash	32 KB(4 KB usados para el bootloader)
Dimensiones	68.6 x 53.3mm.

Tabla 17: Especificaciones Arduino Leonardo

Las características principales del **Arduino Nano** son las siguientes:

Microcontrolador	Atmel ATmega328
Tensión de funcionamiento	5 V
Tensión de entrada (recomendada)	7-12 V
Tensión de entrada (mínimo y máximo)	6-20 V
Pines digitales E/S	14 (6 de los cuáles para generar PWM)
Pines analógicos	8
Corriente DC por pin	40mA
SRAM	2 KB
EEPROM	1 KB
Velocidad del reloj	16 MHz
Memoria Flash	32 KB
Dimensiones	18,5mm x 43,2mm

Tabla 18: Especificaciones Arduino Nano

En la siguiente imagen observamos la distribución de los distintos pines del **Arduino Nano**:

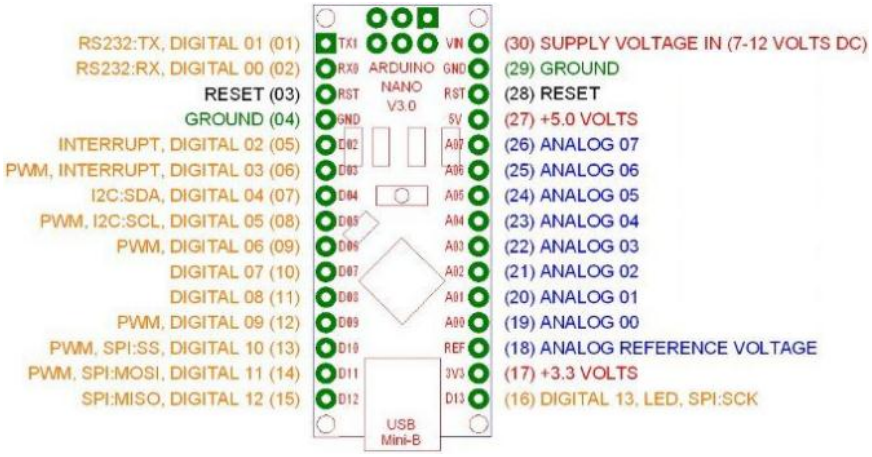


Figura 58: Distribución de pines Arduino Nano

- Anexos II:DISIPACION DE CALOR

En todo componente que genere calor se debe que mantener la temperatura de la junta o unión **(Tj)** por debajo del valor máximo indicado por el fabricante. La junta es el lugar donde se genera el calor y se trata de la propia pastilla o "chip". Se trata de una zona muy pequeña que puede alcanzar fácilmente los 150ºC, lo que suele llevar al transistor a su destrucción.

Para conseguir un flujo de energía calorífica de un punto a otro, debe existir una diferencia de temperatura. El calor se desplazará del punto más caliente al punto más frío, pero diferentes factores dificultan dicho paso. A estos factores se les denomina, resistencias térmicas para asimilarlas a las resistencias eléctricas.

La siguiente figura 1 muestra la arquitectura de un circuito integrado y sus componentes resistivos térmicos descritos.

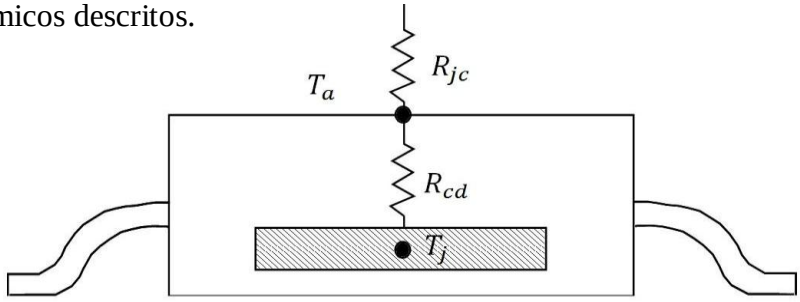


Figura 59: CI con encapsulado plástico

Componentes metálicos transfieren con más facilidad el calor que genera el chip, puesto que disponen de una superficie mejor conductora del calor y por convección dicho calor se transfiere al aire que los rodea (Convección: enfriamiento debido al movimiento ascendente del aire caliente y la reposición de aire frio). De la misma forma estos dispositivos nos permiten realizar un mejor acoplamiento con otros elementos metálicos que a su vez absorben calor también, además de permitir una mayor superficie de contacto con el aire que es el modo más económico de disipar calor.

Anexos II: Ejemplos radiadores

Un disipador puede ser un pedazo de aluminio que este colocado en una cara del regulador de voltaje. En esta sección se dispone una serie de posibles ejemplos de radiadores a emplear con nuestro regulador de tensión LM317.

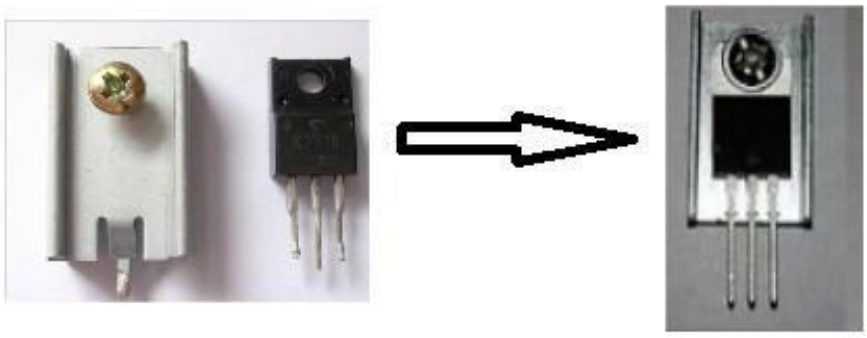


Figura 60: Disipador de calor de aluminio

Se muestran algunos ejemplos de otros tipos de disipadores con distinta forma pero del mismo material

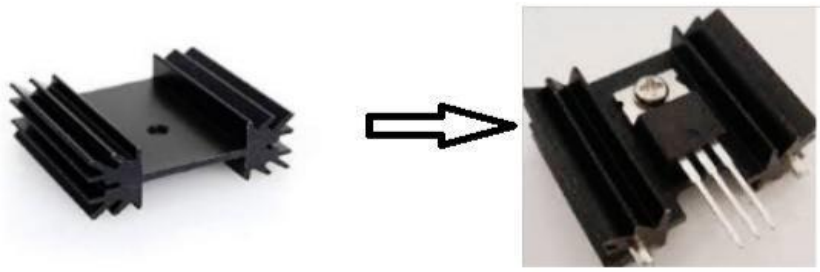


Figura 61: 25 x 34 x 12mm disipador de calor de aluminio

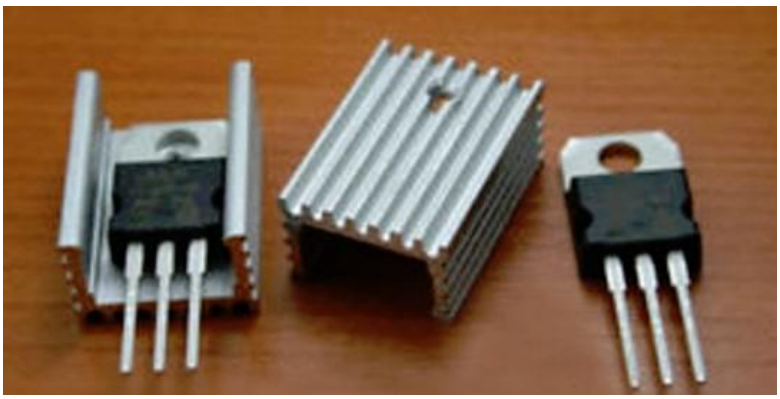


Figura 62: 21x15x10mm disipador de calor de aluminio

- Anexos III : KIT ROBOT-CAR ARDUINO 4WD

1. CHASIS (PLACAS METACRILATO)

El primer componente son dos placas de metacrilato que servirán de chasis del automóvil. Estas dos placas vienen recubiertas con un papel adhesivo a modo de protección.

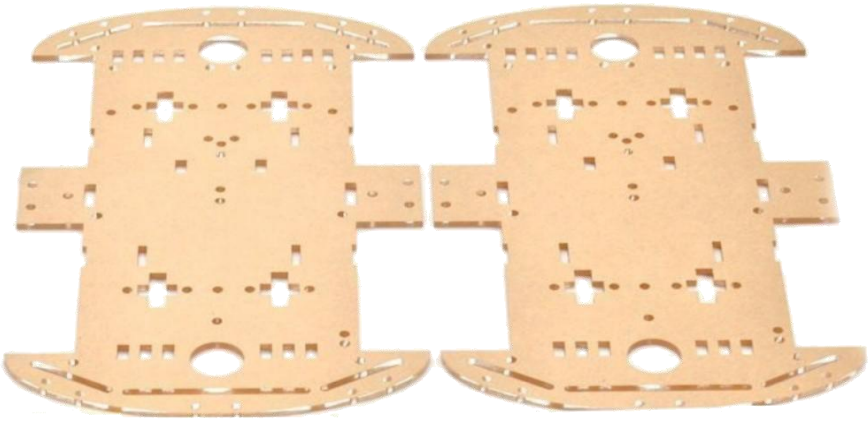


Figura 63: Chasis vehículo

El poli metacrilato de metilo (PMMA), comúnmente conocido como metacrilato, acrílico, plexiglás es un material muy conocido, versátil, fácil de manipular. Se considera una alternativa óptima al vidrio (también como "vidrio acrílico"), además es uno de los materiales plásticos de mayor consumo.

En comparación al vidrio mucho más flexible, más transparente, más resistente a impactos y mejor aislante térmico y acústico. La única desventaja es que se raya con facilidad, es por esto que es un material poco recomendado para casos en los que se aplica un desgate mecánico.

2. RUEDAS DE DIRECCION

Para este proyecto se dispone de cuatro ruedas de dirección. En una primera instancia se valora la opción de utilizar solo dos (con una tercera rueda loca, es decir, sin motor) pero se perdería parte de la potencia y de la velocidad a la hora de la práctica.

Figura 64: Ruedas de dirección

3. ZÓCALO PARA PILAS (18650 Batería de litio recargable 3,7 V)

Se trata de la base de lo que se emplea como fuente de alimentación para alimentar los motores DC además del puente en H. Está diseñado para utilizar pilas de litio de 3,7V. Las dimensiones son las siguientes: 7,3cm x 2,1cm ancho x 2 cm alto.

Figura 65: Zócalo porta pilas

4. MOTOR DC

Por lo general cualquier tipo de robot necesita de algún tipo de componente capaz de producir y transmitir movimiento, para esto los más utilizados son los motores DC (corriente continua) y los servomotores.

Figura 66: Motor DC

Los motores DC están compuestos de un estator y un rotor. Por lo general en los motores más pequeños el estator se compone de imanes para general el campo magnético. Para motores de mayor tamaño se suelen utilizar devanados de excitación de campo para conseguir ese campo magnético. Este motor utilizado se alimenta de 3-6V y tiene una ratio de reducción de 1:48

5. ENCODERS DE VELOCIDAD

El encoder se trata de un transductor rotativo, que sirve para conocer la, velocidad, aceleración y posición angular de un eje del rotor de un motor mediante una señal eléctrica. Está formado por un disco (de vidrio o plástico) que se encuentra "codificado" con partes transparentes y partes opacas que bloquean el paso de la luz emitida por una fuente de luz (normalmente se usan emisores infrarrojos).

Figura 67: Encoder de velocidad

Para este proyecto no se hace uso de ellos puesto que no merece la pena calcular la velocidad del automóvil. En el caso de querer hacerlo sería necesario adquirir emisores de infrarrojos.

6. TORNILLERIA

Por último se dispone de una serie de tornillería para llevar a cabo el montaje y la sujeción de la estructura del chasis junto con motores y ruedas.

-16 tuercas hexagonales de 2,5mm de radio y ancho de 1,5mm para el soporte de los motores a la placa inferior del chasis.

Figura 68: Tuerca hexagonal

-8 tornillos largos de 30mm para fijar los 4 motores (mediante las tuercas hexagonales)

Figura 69: Tornillo largo

-6 pasadores de 30mm que utilizamos para fijar las dos placas de metacrilato.

Figura 70: Pasadores

-12 tornillos cortos de 6 mm le largo para el ajuste de los pasadores a las dos placas.

Figura 71: Tornillo corto

4. BIBLIOGRAFIA Y REFERENCIAS

- Referencias bibliográficas:

[9] Ribas, J. "Arduino Práctico (Manuales Imprescindibles)", España: Anaya, 2013
[10] Ruiz, J. (2012) "Manual de Programación Arduino".

- Citas:

[1]*"Conjunto de los componentes que integran la parte material de una computadora"* Diccionario de la lengua española.
[8] "*the highest-level concept of a system in its environment"IEEE, Recommended Practice for Architectural Description,* IEEE Computer Society, 2000.

- Páginas web:

[7] Concepto: Arquitectura de software
https://cgrw01.cgr.go.cr/rup/RUP.es/SmallProjects/core.base_rup/guidances/concepts/software_architecture_4269A354.html [consulta: 12 noviembre 2018]

[4]Configuración del módulo bluetooth HC-05 usando comandos AT

https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html [consulta: 15 diciembre 2018]

[12]Comunicación inalámbrica en Arduino con módulos RF 433MHz

https://www.luisllamas.es/comunicacion-inalambrica-en-arduino-con-modulos-rf-433mhz/ [consulta: 12enero 2019]

[13]Controlar Arduino con mando a distancia infrarrojo

https://naylampmechatronics.com/blog/36_Tutorial-Arduino-y-control-remoto-Infrarrojo.html [consulta: 3 enero 2019]

[11] CONTROL REMOTO RF SENCILLO

https://www.prometec.net/control-remoto-rf/ [consulta: 4enero 2019]

[2] Diferencias entre microcontroladores y microprocesadores
https://www.monografias.com/trabajos27/microcontroladores/microcontroladores.shtml [consulta: 9 octubre 2018]

[5] LM317 – Circuito para regulador de voltaje variable

https://hetpro-store.com/TUTORIALES/lm317/[consulta: 10 noviembre 2018]

[6]Servomotor 0º-180º - Salidas analógicas PWM con Arduino

https://www.programoergosum.com/cursos-online/arduino/255-salidas-analogicas-pwm-con-arduino/servomotor-0-180[consulta: 2 septiembre 2018]

[3] Tutorial de Uso del Módulo L298N

https://naylampmechatronics.com/blog/11_Tutorial-de-Uso-del-M%C3%B3dulo-L298N.html[consulta: 21 septiembre 2018]