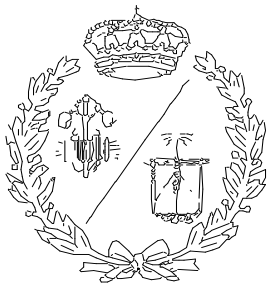
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN
UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Grado

IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE VELOCIDAD PARA UN PROTOTIPO DE VEHÍCULO ELÉCTRICO AUTÓNOMO

(Implementation of a Speed Control System for a Prototype of Autonomous Electric Vehicle)

Para acceder al Título de

GRADUADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Diego Sánchez López

Septiembre - 2019

AGRADECIMIENTOS:

Quisiera aprovechar la ocasión para mostrar mi agradecimiento a mi tutor de este Proyecto Fin de Grado, Luciano Alonso Rentería, por su ayuda y disponibilidad durante la realización de este proyecto.

Asimismo, agradecer el apoyo y compresión mostrado por mi familia, amigos y compañeros durante estos años, los cuales siempre han estado ahí cuando los he necesitado.

ÍNDICE DE CONTENIDOS

1 INTRODUCCIÓN 1
1.1 PLANTEAMIENTO Y OBJETIVO 1
1.2 BREVE HISTORIA DEL AUTOMÓVIL 1
1.3 TIPOS DE VEHÍCULOS 3
1.3.1 Vehículo gasolina 3
1.3.2 Vehículo diésel 4
1.3.3 Vehículo híbrido eléctrico 5
1.3.4 Vehículo híbrido enchufable 6
1.3.5 Vehículo híbrido de GLP o GNC 7
1.3.6 Vehículo eléctrico 7
1.3.7 Vehículo autónomo 10
1.4 ETIQUETAS AMBIENTALES 11
2 VISIÓN GENERAL 13
2.1 EL VEHÍCULO EXPERIMENTAL 13
2.2 UNIDAD PRINCIPAL DEL SISTEMA 14
3 ARDUINO 17
3.1 TIPOS DE PLACAS 17
3.1.1 Arduino Uno 17
3.1.2 Arduino Due 19
3.1.3 Arduino Mega 2560 20
3.2 ARDUINO IDE 21
3.2.1 Interfaz 22
3.2.2 Estructura de un sketch 23
3.2.3 Librerías 24
3.2.4 Monitor serie 24
3.3 MÉTODOS DE TRANSMISIÓN DE INFORMACIÓN 25
3.3.1 Comunicación Serie mediante puerto USB 25
3.3.2 Comunicación UDP mediante puerto Ethernet 26
4 SISTEMA DE CONTROL 27
4.1 ESTRUCTURA GENERAL 27
4.2 CONTROLADOR PID 28
4.2.1 Acción proporcional 28
4.2.2 Acción derivativa 29
4.2.3 Acción integral 30
4.2.4 Programación del controlador 32

4.3 ELEMENTOS PARA CONSEGUIR UN VEHÍCULO AUTÓNOMO 33
4.3.1 Volante 33
4.3.2 Pedal de acelerador y freno 34
4.3.3 Sensor LIDAR 35
4.3.4 Radar 36
4.3.5 Cámaras 36
4.4 ELEMENTOS DE ALIMENTACIÓN DEL VEHÍCULO 37
4.4.1 Motor 37
4.4.2 Baterías 37
4.4.3 Seta de emergencia 38
4.4.4 Rectificador 38
4.4.5 Transformador 39
4.4.6 Inversor 39
5 SISTEMA DE VELOCIDAD: LA CONTROLADORA 41
5.1 AJUSTES SOBRE LA CONTROLADORA 41
5.1.1 Control de velocidad manual o autónomo 42
5.1.2 Conocer la velocidad actual mediante el sensor hall 45
5.1.3 Parámetros programables de la controladora 47
6 DESARROLLO 51
6.1 PWM 51
6.1.1 Transformación señal PWM en tensión 52
6.1.2 Pruebas de funcionamiento 54
6.2 SENSOR HALL 57
6.2.1 Filtrado señal sensor Hall 61
6.2.2 Pruebas de funcionamiento 62
6.3 SISTEMA DE CONTROL DE VELOCIDAD FINAL 65
7 MONTAJE DE CIRCUITOS 66
7.1 MONTAJE PRUEBA PWM 66
7.2 MONTAJE PRUEBA SENSOR HALL 67
7.3 MONTAJE FINAL DEL CONTROL DE VELOCIDAD 67
8 MARCADOR INTERACTIVO 68
8.1 PROCESSING 68
8.1.1 Interfaz 68
8.1.2 Monitor visual 69
8.2 ELEMENTOS 70
8.2.1 Cuentakilómetros 70
8.2.2 Batería 70

8.2.3 Testigos de luz 71
8.2.4 Intermitentes 71
8.2.5 Marchas 71
8.3 RESULTADO Y PRUEBAS DE FUNCIONAMIENTO 71
9 POSIBLES FUTURAS FUENTES DE INVESTIGACIÓN 75

ANEXO 1. CÓDIGOS PARCIALES DE ARDUINO 76
ANEXO 2. CÓDIGO ARDUINO FINAL DEL SISTEMA DE CONTROL DE VELOCIDAD 79
ANEXO 3. CÓDIGO PROCESSING MARCADOR DE VELOCIDAD INTERACTIVO 82
ANEXO 4. HOJAS DE CARACTERÍSTICAS (DATASHEETS) Y MANUALES 88

PRESUPUESTO 89

BIBLIOGRAFÍA 90

ÍNDICE DE FIGURAS

Figura 1.1 El Fadier de Nicolas Cugnot 2
Figura 1.2 La Jamais Contente 2
Figura 1.3 Diagrama P-V motor Otto 3
Figura 1.4 Benz Patent-Motorwagen 4
Figura 1.5 Diagrama P-V motor diésel 5
Figura 1.6 Mercedes-Benz 260 D 5
Figura 1.7 Toyota Prius de 1997 6
Figura 1.8 F3DM de BYD 6
Figura 1.9 Autobús hibrido con GLP 7
Figura 1.10 Punto de carga de un garaje privado 9
Figura 1.11 Punto de carga público Tesla 9
Figura 1.12 Sistema de detección de presencia humana 10
Figura 1.13 Vehículo Waymo de Google 10
Figura 1.14 Elementos para la autonomía de Waymo 11
Figura 1.15 Etiquetas de contaminación ambiental de la DGT 12

Figura 2.1 Vehículo experimental ALSO2 13
Figura 2.2 Interruptor modo liebre / tortuga 14
Figura 2.3 Ordenador industrial NISE 3500 14
Figura 2.4 Interior NISE 3500 15
Figura 2.5 SSD 15
Figura 2.6 Módulo de memoria RAM 16
Figura 2.7 Pantalla portátil 16

Figura 3.1 Logotipo de Arduino 17
Figura 3.2 Arduino Uno 18
Figura 3.3 Pines del chip ATMega328 19
Figura 3.4 Arduino Due 19
Figura 3.5 Arduino Mega 2650 21
Figura 3.6 Elementos de la interfaz Arduino IDE 22
Figura 3.7 Partes principales de un sketch 23
Figura 3.8 Monitor serie 24
Figura 3.9 Arduino Uno conectado mediante USB 25
Figura 3.10 Ethernet Shield para Arduino Uno 26

Figura 4.1 Sistema de lazo cerrado 28
Figura 4.2 Respuesta al aumentar Kp 29
Figura 4.3 Respuesta al aumentar Kd 30
Figura 4.4 Respuesta al aumentar Ki 31
Figura 4.5 Suma de las tres acciones 32
Figura 4.6 Motor del volante 34
Figura 4.7 Potenciómetro del acelerador 34
Figura 4.8 Sistema de frenado 35
Figura 4.9 Radar frontal 36
Figura 4.10 Cámara del retrovisor 36
Figura 4.11 Motor del vehículo 37
Figura 4.12 Baterías instaladas 37
Figura 4.13 Seta de emergencia 38
Figura 4.14 Rectificador 38
Figura 4.15 Toma de conexión de carga 39
Figura 4.16 Transformador 39
Figura 4.17 Inversor 40

Figura 5.1 Controladora Curtis 1268 41
Figura 5.2 Conexiones de alta corriente 41
Figura 5.3 Conexiones de serie de la controladora Curtis 1268 42
Figura 5.4 Conexiones de la controladora 42
Figura 5.5 Esquema básico selección modo 43
Figura 5.6 Pines de la controladora encargados de la velocidad 43
Figura 5.7 Esquema conexión pines con el potenciómetro 44
Figura 5.8 Circuitos paralelos con clemas 44
Figura 5.9 Configuración de los interruptores 45
Figura 5.10 Pines del sensor Hall 45
Figura 5.11 Rueda del vehículo 46
Figura 5.12 Pines de programación 47
Figura 5.13 Programador Curtis 1311 47
Figura 5.14 Gráfica Throttle Map 49
Figura 5.15 Gráfica del freno regenerativo 49
Figura 5.16 Gráfica del mapeo del campo 50

Figura 6.1 Entrada de control frente a la intensidad luminosa 51
Figura 6.2 Filtro RC con amplificador 52

Figura 6.3 Salidas que admiten PWM 52
Figura 6.4 Montaje filtro RC con amplificador 53
Figura 6.5 Registro B del Timer1 54
Figura 6.6 Señal PWM al 0 % y tensión de 0 V 55
Figura 6.7 Señal PWM al 25 % y tensión de 1.25 V 55
Figura 6.8 Señal PWM al 50 % y tensión de 2.5 V 56
Figura 6.9 Señal PWM al 75% y tensión de 3.75 V 56
Figura 6.10 Señal PWM al 100 % y tensión de 5 V 57
Figura 6.11 Sensor Hall 58
Figura 6.12 Osciloscopio Fluke 199C 59
Figura 6.13 Señal de pulsos del sensor Hall al 25 % 60
Figura 6.14 Señal de pulsos del sensor Hall al 50 % 60
Figura 6.15 Señal de pulsos del sensor Hall al 75 % 61
Figura 6.16 Señal de pulsos del sensor Hall al 100 % 61
Figura 6.17 Filtro paso bajo 62
Figura 6.18 Montaje filtro paso bajo 62
Figura 6.19 Señal sin filtrar y filtrada al 25 % 63
Figura 6.20 Señal sin filtrar y filtrada al 50 % 63
Figura 6.21 Señal sin filtrar y filtrada al 75 % 64
Figura 6.22 Señal sin filtrar y filtrada al 100 % 64
Figura 6.23 Montaje resultante del Sistema de Control de Velocidad 65

Figura 7.1 Esquema del montaje para prueba de la transformación PWM a tensión 66
Figura 7.2 Esquema del montaje para prueba del sensor Hall 67
Figura 7.3 Esquema del montaje final del Sistema de Control de Velocidad 67

Figura 8.1 Logotipo de Processing 68
Figura 8.2 Elementos de la interfaz Processing 69
Figura 8.3 Monitor visual 70
Figura 8.4 Marcador interactivo 71
Figura 8.5 Prueba marcador 1 72
Figura 8.6 Prueba marcador 2 73
Figura 8.7 Prueba marcador 3 74

Figura 9.1 Montaje conjunto al sistema de navegación GPS 75

ÍNDICE DE TABLAS

Tabla 1. Características Arduino Uno 18
Tabla 2. Características Arduino Due 20
Tabla 3. Características Arduino Mega 2650 21
Tabla 4. Selección del prescaler 54
Tabla 5. Presupuesto 89

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO Y OBJETIVO

El objetivo del trabajo es el diseño e implementación de un sistema de control de velocidad para un vehículo eléctrico experimental.

El vehículo dispone de serie de un motor de corriente continua de 48V junto con su controladora electrónica y un sensor de efecto Hall que proporciona una señal de pulsos de frecuencia proporcional a la velocidad del motor.

Sobre el vehículo se ha instalado un computador que ejercerá la tarea de control de alto nivel. Este computador envía al sistema desarrollado la velocidad de referencia deseada.

El sistema desarrollado recibe la velocidad de referencia del computador central, así como la señal procedente del sensor de efecto Hall, y ejecuta el algoritmo de control de bajo nivel. Dicho algoritmo obtiene el valor de tensión a enviar a la controladora para adquirir la velocidad deseada.

Todo el sistema se ha implementado mediante una placa Arduino, junto con la electrónica de acondicionamiento de las señales desarrollada para este propósito.

Como complemento, se ha realizado un marcador interactivo mediante el software Processing para mostrar información relevante sobre la conducción al ocupante del vehículo.

Este sistema será instalado en un vehículo eléctrico propiedad del departamento de TEISA de la Universidad de Cantabria con el objetivo final de obtener un vehículo totalmente autónomo.

1.2 BREVE HISTORIA DEL AUTOMÓVIL

En 1769, Nicolas Cugnot, escritor e inventor francés, construyó el primer vehículo de tres ruedas impulsado por vapor denominado “El Fadier” (Fig. 1.1) que poseía una marmita adaptada a la parte delantera. La función inicial de este vehículo fue el transporte de piezas de artillería pesadas. Este consiguió alcanzar velocidades de hasta 4 km/h y su conducción era muy compleja, ya que era un vehículo muy pesado.

Figura 1.1 El Fadier de Nicolas Cugnot

Una vez que las primeras baterías recargables se dieron a la luz en 1860, se desarrolló el primer vehículo eléctrico capaz de superar la barrera de los 100km/h, conocido como La Jamais Contente (Fig. 1.2), construido con una carrocería de aleación ligera de aluminio, tungsteno y magnesio en forma de torpedo por el belga Camilles Jenatzy, hijo de un fabricante de neumáticos.

Como bien se comentó antes, este vehículo alcanzó una velocidad de exactamente 105,88 km/h, batiendo un récord (el cual perduró tres años) y superando en aquel momento al Conde Gastón de Chasseloup-Laubat con una marca de 92,78 km/h lograda en 1899.

Figura 1.2 La Jamais Contente

1.3 TIPOS DE VEHÍCULOS

1.3.1 Vehículo gasolina

Estos vehículos presentan un motor gasolina o también conocido como motor Otto. Tienen como principal ventaja frente al diésel que su contaminación es muy inferior pero el consumo de estos es superior.

Este tipo de motores presentar 4 etapas o tiempos diferenciadas:

- Admisión: Se abre la válvula de admisión y la mezcla de aire y combustible va entrando en la cámara, haciendo que el pistón descienda hasta el PMI (a presión constante ya que la válvula está abierta y la presión es igual a la exterior).

- Compresión: El pistón asciende hasta el PMS comprimiendo la mezcla (compresión adiabática)

- Combustión y expansión: Antes de alcanzar el PMS, una bujía hace saltar una chispa, que inicia la combustión de la mezcla (el volumen se mantiene constante y la presión aumenta). Como consecuencia de esta combustión, la temperatura aumenta y hace que el pistón baje hasta el PMI (expansión adiabática)

- Escape: Finalmente, la válvula de escape se abre y el pistón asciende (volumen constante y la presión disminuye) hasta el PMS y el gas sale al exterior por la válvula.

Una vez terminado el ciclo (Fig. 1.3), vuelve a empezar desde el principio.

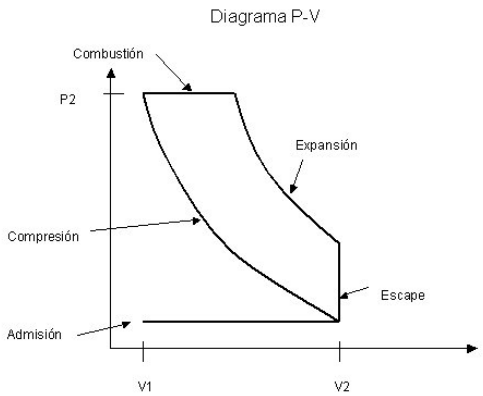


Figura 1.3 Diagrama P-V motor Otto

El primer vehículo de combustión interna de gasolina fue un biplaza de tres ruedas patentado por Karl Benz en 1885 conocido como" Benz Patent-Motorwagen" (Fig. 1.4). En 1888, su mujer Bertha realizó un viaje con el vehículo de 104 kilómetros, lo que hasta entonces era algo nunca visto, puesto que solo se realizaban trayectos cortos de prueba.

Figura 1.4 Benz Patent-Motorwagen

1.3.2 Vehículo diésel

Este tipo de motores presentan una mayor contaminación que los motores gasolina, por ello en la actualidad, ya existen medidas para que, con el paso de los años, en un futuro no muy lejano, los vehículos de motor diésel desaparezcan del mercado.

El motor diésel, al igual que el de gasolina, consta de cuatro etapas o tiempos, pero presentan algunas diferencias en la admisión y combustión:

- Admisión: Se abre la válvula de admisión y va entrando aire en la cámara del pistón, haciendo que el pistón descienda hasta el PMI (a presión constante ya que la válvula está abierta y la presión es igual a la exterior).

- Compresión: El pistón asciende hasta el PMS comprimiendo el aire (compresión adiabática).

- Combustión y expansión: Antes de que el pistón alcance el PMS, un inyector introduce gasoil que inicia la combustión (la presión se mantiene constante y el volumen aumenta). Como consecuencia de esta combustión, la temperatura aumenta y hace que el pistón baje hasta el PMI (expansión adiabática).

- Escape: Finalmente, la válvula de escape se abre y el pistón asciende (volumen constante y la presión disminuye) hasta el PMS y el gas sale por la válvula de escape.

Una vez terminado el ciclo (Fig. 1.5), vuelve a empezar desde el principio.

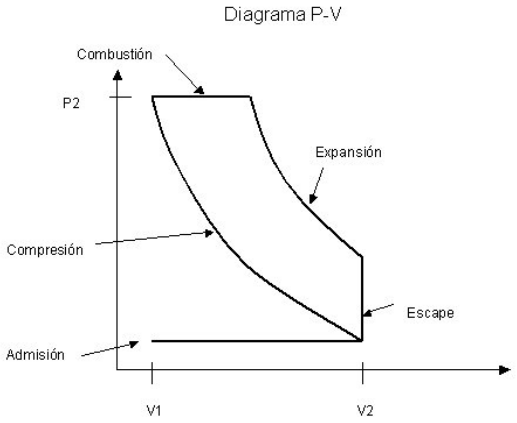


Figura 1.5 Diagrama P-V motor diésel

El primer vehículo diésel fue el Mercedes-Benz 260 D (Fig. 1.6) presentado en plena Alemania nazi en el salón de Berlín en 1936, que poseía un motor de 4 cilindros de 2545 cc., que alcanzaba una velocidad máxima de 94km/h.

Figura 1.6 Mercedes-Benz 260 D

1.3.3 Vehículo híbrido eléctrico

Un vehículo híbrido es aquel que cuenta con dos motores diferentes, por un lado, un motor de combustión interna (normalmente gasolina) y por otro un potente motor eléctrico. De este modo, el vehículo puede alternar o utilizar ambos motores en función de la calzada o las necesidades del conductor.

El primer vehículo hibrido comercializado en serie fue el Prius (Fig. 1.7) de la marca japonesa Toyota en 1997, siendo en la actualidad el vehículo híbrido más vendido en el mundo.

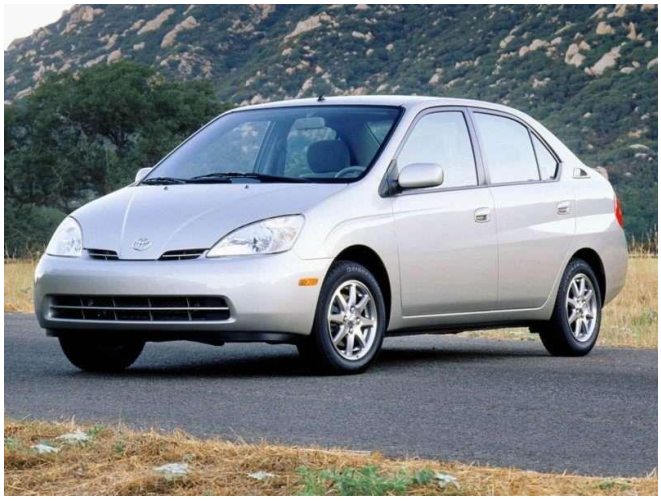


Figura 1.7 Toyota Prius de 1997

1.3.4 Vehículo híbrido enchufable

El vehículo híbrido enchufable es similar al anteriormente descrito pero este cuenta con una toma para poder recargar la batería del motor eléctrico.

El primer vehículo de este tipo que se puso a la venta en el mercado fue el sedán F3DM (Fig. 1.8), fabricado por la empresa china BYD Auto en 2008.

Figura 1.8 F3DM de BYD

1.3.5 Vehículo híbrido de GLP o GNC

Estos tipos de vehículos cuentan con un motor alimentado, ya sea por gas natural comprimido (GNC) o de gas licuado del petróleo (GLP) y otro de gasolina. Pueden existir variantes en los que se utilice electricidad en vez de gasolina. Durante el funcionamiento del vehículo, podemos elegir el tipo de combustible a usar mediante un interruptor. Con el uso del gas se obtiene una disminución considerable de la contaminación y del desgaste de las mecánicas del vehículo, además de ser un combustible más barato que el resto.

Actualmente es más frecuente en Europa el uso del GLP y en Latinoamérica el GNC, aunque con el paso de los años el GNC ganará un mayor atractivo en nuestro continente.

En la década de los 80, llega a España este tipo de vehículo para su implementación en autobuses (Fig. 1.9) y taxis inicialmente.

Figura 1.9 Autobús hibrido con GLP

1.3.6 Vehículo eléctrico

En este apartado se hará un mayor hincapié debido a que el vehículo a tratar en este trabajo es de este tipo.

Un vehículo eléctrico es aquel que cuenta únicamente con un motor eléctrico alimentado por una batería.

El coste de la sustitución de una de estas baterías supera los 5.000 €, aunque los fabricantes suelen optar por ofrecer el alquiler de la batería pagando un coste en torno a 70 € / mes, o lo que es lo mismo, 7.500€ / año.

En nuestro caso, nuestro vehículo no cuenta con una batería de estas características ya que es un vehículo experimental. Más adelante hablaremos sobre la alimentación de nuestro vehículo.

Las ventajas de un coche eléctrico son las siguientes:

- Son más silenciosos que los vehículos diésel y gasolina.
- Contaminan menos ya que no expulsan humos a la atmósfera.
- No necesitan circuitos de refrigeración ni aceite, por lo que son poseen motores más ligeros y simples.
- No poseen ni pedal de embrague ni palanca de cambios, por lo que su conducción es más sencilla.
- Bajo mantenimiento del motor debido a su simplicidad.
- Coste por kilómetro inferior que los vehículos de combustible, debido a su mayor eficiencia.
- Posibilidad de acceder a más zonas del casco urbano por su distintivo ambiental impuesto por la DGT en España.

Aunque hasta ahora este tipo de vehículo parecía que proporcionaba una gran fuente de ventajas, este también conlleva algunos inconvenientes:

- Su autonomía es menor, lo que puede llegar a ser un inconveniente en viajes lejanos, donde dependeremos de encontrar un lugar donde exista un punto de carga de coches eléctricos, ya que actualmente no existen muchos pero que con el tiempo iremos viendo más frecuentemente.
- Los tiempos de carga de carga pueden variar entre 30 minutos y 8 horas, lo que puede suponer un punto negativo si necesitamos disponer de nuestro vehículo y este no se encuentra cargado lo suficiente.
- Necesidad de un garaje donde poder cargar tu vehículo.
- Al adquirir un vehículo eléctrico, deberás realizar una inversión inicial mayor que al comprar un coche convencional y será preciso realizar una instalación de un punto de carga especial en tu garaje (Fig. 1.10) que actualmente está subvencionado y ronda en torno a los 1000€.
- Coste de mantenimiento de la batería elevado.

Figura 1.10 Punto de carga de un garaje privado

Una vez conocidos los diferentes pros y contras que conlleva un coche eléctrico la pregunta es, ¿es rentable y apropiado un vehículo eléctrico?

La respuesta a esta pregunta es sencilla. Si dispones de garaje propio donde poder instalar y cargar tu vehículo y destinarás el uso de tu vehículo a rodar por ciudad y valoras el medio ambiente, este es tu tipo de vehículo. Además, debemos tener en cuenta que actualmente los puntos de carga de vehículos eléctricos (Fig. 1.11) en España no abundan por igual en todo el país, siendo Madrid y Barcelona las ciudades con mayor número de estaciones con posibilidad de carga de vehículo eléctrico.

Figura 1.11 Punto de carga público Tesla

Si alguno de estos requisitos no encaja con tus necesidades, deberás recurrir a la compra de un vehículo de los citados anteriormente.

1.3.7 Vehículo autónomo

Un vehículo autónomo es aquel capaz de desplazarse sin el manejo de un conductor, por lo que cuenta de una gran cantidad de sensores, cámaras, radares, sistema de posicionamiento y otros sistemas para hacer esto posible sin peligros. En la actualidad, aún no existe un mercado de este tipo de vehículo, pero si se está trabajando en ello.

Figura 1.12 Sistema de detección de presencia humana

Como bien se comentó anteriormente, el objetivo del departamento es conseguir un coche totalmente autónomo, existiendo así proyectos ya realizados por antiguos estudiantes sobre el vehículo y actualmente otros en paralelo a este.

Un ejemplo es el vehículo Waymo fabricado por Google (Fig. 1.13), el cual no necesita ni el manejo de un conductor ni una conexión a Internet continua ya que todos los sistemas necesarios se encuentran abordo y solo es necesario una conexión a menudo para recibir información concreta como puede ser el estado del tráfico. De esta manera, al no existir una conexión continua con Internet, se disminuye la posibilidad de que el vehículo pueda recibir un ciberataque y crear una catástrofe.

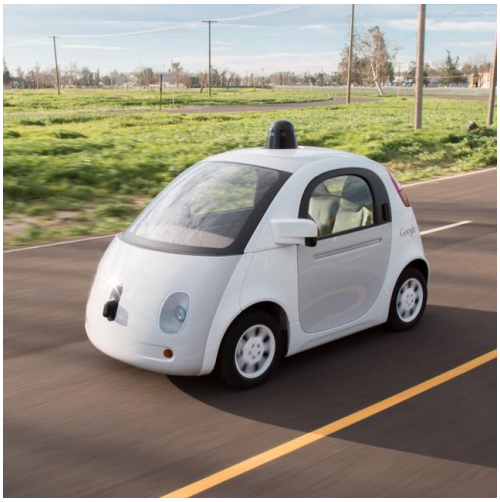


Figura 1.13 Vehículo Waymo de Google

Para conseguir esta total autonomía, Waymo cuenta con una gran cantidad de cámaras, entre ellas una cámara 360º ubicada en el techo, sensores LIDAR, radares y más elementos para hacer esto posible como se puede observar en la Figura 1.14.

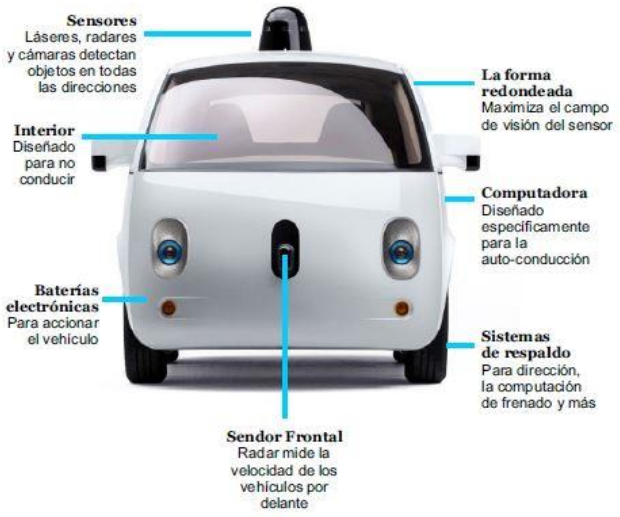


Figura 1.14 Elementos para la autonomía de Waymo

En 2012, Steven Mahan, un ciudadano con discapacidad visual de un 95 por ciento, realizó el primer viaje oficial en un vehículo autónomo, un Toyota Prius equipado con tecnología de Google en Austin, Texas.

1.4 ETIQUETAS AMBIENTALES

Existen cuatro distintos ambientales (Fig. 1.15) diferentes en España en función del impacto medioambiental que genere un vehículo.

Esta distinción tiene origen en el Plan nacional de calidad del aire y protección de la atmósfera 2013-2016 en el que se declara que las tanto las partículas como el dióxido de nitrógeno (NO2) son la principal fuente de emisión de contaminación y propone una clasificación de los vehículos en cuatro categorías diferentes en función de la contaminación que estos emiten.

Esta clasificación permite el acceso o la denegación de un vehículo en ciertas partes de grandes ciudades como pueden ser Madrid y Barcelona y la posibilidad de estar exento del pago de impuestos en algunos casos.

En los vehículos eléctricos, como es en nuestro caso, la asignada es la etiqueta ambiental “0 Azul”.

Figura 1.15 Etiquetas de contaminación ambiental de la DGT

La descripción de los vehículos a los que se les adjudica esta etiqueta es la siguiente:

“Vehículos clasificados en el Registro de Vehículos como vehículos eléctricos de batería (BEV), vehículo eléctrico de autonomía extendida (REEV), vehículo eléctrico híbrido enchufable (PHEV) con una autonomía mínima de 40 kilómetros o vehículos de pila de combustible.”

Esta etiqueta es la menos restrictiva de las cuatro existentes, por lo que es una buena apuesta con vistas al futuro un vehículo de este tipo.

2 VISIÓN GENERAL

2.1 EL VEHÍCULO EXPERIMENTAL

El vehículo experimental sobre el que se realizará este trabajo se trata de un vehículo de tracción delantera biplaza de cuatro ruedas del 2008 con un peso en vacío de aproximadamente 600 Kg modelo ALSO2 (Fig. 2.1) creado de la mano de la empresa TEYCARS, compañía la cual cuenta con 16 años de experiencia en la venta de vehículos 100% eléctricos.

Figura 2.1 Vehículo experimental ALSO2

Cuenta con una autonomía entorno a 90km con una carga completa de las baterías y una velocidad máxima de aproximadamente 45 km/h, disponiendo así de un interruptor con dos modos, el modo ‘liebre’ y el modo ‘tortuga’ (Fig. 2.2).

- Modo liebre: El acelerador es más sensible al aumento de velocidad, pudiendo alcanzar la velocidad máxima permitida del vehículo.
- Modo tortuga: El acelerador es menos sensible que en el modo anterior y alcanza una velocidad máxima en torno a 30 km/h.

Figura 2.2 Interruptor modo liebre / tortuga

Por otro lado, cuenta con un interruptor para alternar entre la marcha directa y marcha atrás, haciendo activar en esta última una bocina para indicar que el vehículo está dando marcha atrás.

Este vehículo fue obtenido de segunda mano por el departamento de TEISA para realización de diferentes trabajos de investigación con el objetivo final de conseguir un vehículo eléctrico totalmente autónomo.

2.2 UNIDAD PRINCIPAL DEL SISTEMA

Para la carga en placa de los diferentes programas y para la ejecución de las diferentes pruebas, se dispone de un ordenador industrial modelo NISE 3500 (Fig. 2.3) de la marca NEXCON. Es denominado industrial porque está pensado para funcionar en ambientes peligrosos, donde pueden existir grandes temperaturas o partículas que lo dañen, entre otros.

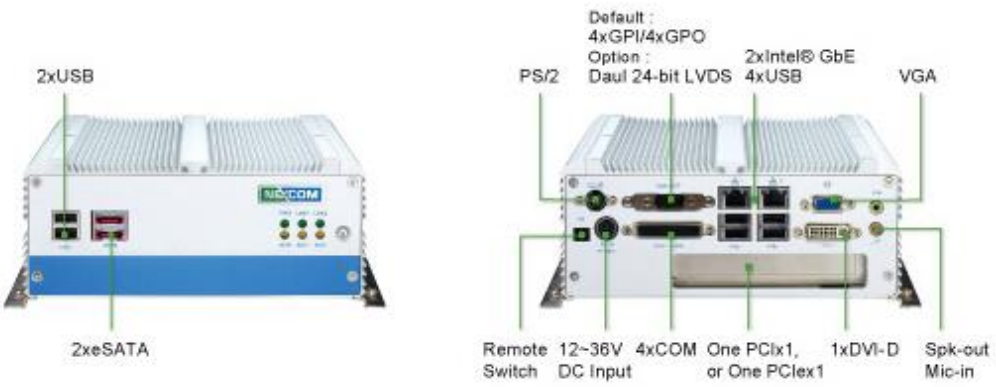


Figura 2.3 Ordenador industrial NISE 3500

Presenta unas dimensiones de 195 mm de ancho, 268 mm de largo y 80 mm de alto. En su parte inferior posee dos soportes con agujeros para fijar el ordenador a una superficie mediante tornillos.

Una de las grandes diferencias con un ordenador convencional es que este no cuenta con ventiladores para disipación de calor como se puede ver en la Figura 2.4, realizando esta gracias a su chasis construido en aluminio, que hace contacto con el disipador colocado encima del procesador. Con ello, conseguimos reducir ruidos y evitar la introducción de partículas de polvo en el interior del equipo.

Figura 2.4 Interior NISE 3500

Adicionalmente, se remplazó la unidad de disco duro (HDD) por un unidad de estado sólido (SSD) de 120GB de la marca WD (Fig. 2.5). Estos tipos de dispositivo de almacenamiento mejorar notablemente los tiempos de inicio del sistema operativo como la velocidad de lectura y escritura de datos.

Figura 2.5 SSD

Para la mejora del funcionamiento del equipo a la hora de ejecutar simultáneamente varias simulaciones del presente trabajo y de futuros, se remplazó el módulo de memoria RAM que poseía actualmente de 2GB DDR3 por uno de 4GB DDR3 como el de la Figura 2.6.

Figura 2.6 Módulo de memoria RAM

Para la realización de las diferentes pruebas en el taller donde se encuentra el vehículo, contamos con una pantalla portátil de 7” (Fig.2.7) donde poder visualizar la imagen emitida por el ordenador, la cual puede ser colocado en la parte central superior del salpicadero del vehículo mediante un soporte previamente instalado.

Figura 2.7 Pantalla portátil

Mediante el uso de este ordenador, se realizó el sistema de control de velocidad mediante programación en código C++ en el software Arduino IDE.

3 ARDUINO

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, por lo que cualquiera puede crear, utilizar y compartir programas realizados por otros usuarios.

Figura 3.1 Logotipo de Arduino

Arduino nació en el año 2003 cuando varios estudiantes de instituto de Itala decidieron crear una alternativa más económica a las costosas placas que existían en aquel momento, las “BASIC Stamp”, que costaban alrededor de cien dólares.

El resultado fue la creación de una placa en la que se encuentra instalado un microcontrolador, el cual es el encargado de guardar la información y procesarla. Para ello, existen una gran cantidad de salidas y entradas tanto analógicas como digitales con diferentes funciones, donde es posible la conexión de infinidad de elementos.

3.1 TIPOS DE PLACAS

En Arduino puedes encontrarte con diferentes tipos de modelos, entre los que destacan Arduino Uno, Due y Mega 2560.

3.1.1 Arduino Uno

Arduino Uno (Fig. 3.2) es una placa de microcontrolador basada en ATmega328P.Tiene 14 pines de entrada/salida digital (de los cuales 6 se pueden usar como salidas PWM), 6 entradas analógicas, un reloj de 16 MHz, una conexión USB, un conector de alimentación, un conector ICSP y un botón de reset.

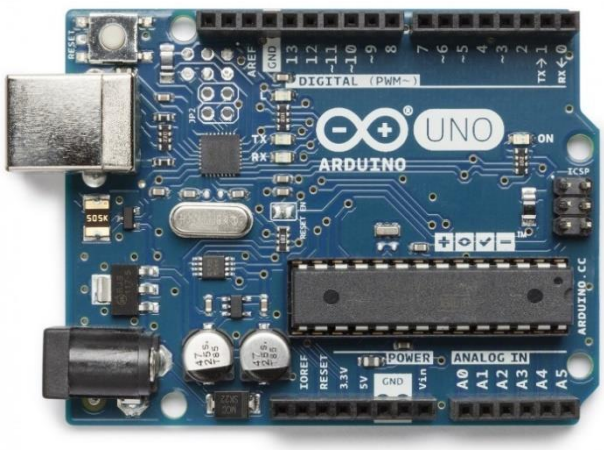


Figura 3.2 Arduino Uno

Algunas de las características más importantes de este Arduino se recogen a continuación en la siguiente tabla:

Tabla 1. Características Arduino Uno

Microcontrolador	ATmega328
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
Pines de entrada y salida digital	14
Pines de entrada y salida PWM	6
Pines de entradas analógicas	6
Memoria flash	32 KB
Velocidad del reloj	16 Mhz
Número de leds	13
Largo	68.6 mm
Ancho	53.4 mm
Peso	25 g

La placa puede ser alimentada mediante USB, que además no sirve como medio de comunicación con el ordenador o mediante un transformador que cumpla los valores de tensión admitidos.

El chip microcontrolador de bajo consumo ATMega328 instalado en esta placa, cuenta con 28 pines que se muestran en la Figura 3.3.

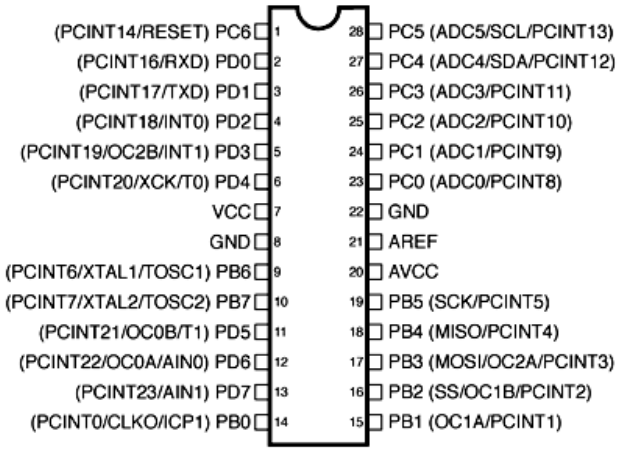


Figura 3.3 Pines del chip ATMega328

En nuestro trabajo, se hizo uso de este tipo de placa por su sencillez y bajo precio entre los ofertados, ya que dispone de los pines y memoria suficiente para la realización del presente trabajo.

3.1.2 Arduino Due

El Arduino Due (Fig. 3.4) es una placa de microcontrolador basada en la CPU Atmel SAM3X8E. Es la primera placa Arduino basada en un microcontrolador de núcleo ARM de 32 bits. Tiene 54 pines de entrada/salida digital (de los cuales 12 se pueden usar como salidas PWM), 12 entradas analógicas, 4 UART, un reloj de 84 MHz, una conexión compatible con USB OTG, 2 DAC (digital a analógico), 2 TWI, un conector de alimentación, un conector SPI, otro JTAG y un botón de reset.

Figura 3.4 Arduino Due

A diferencia de la mayoría de las placas Arduino, la placa Arduino Due funciona a 3.3V. El voltaje máximo que los pines soportan es de 3.3 V. La aplicación de voltajes superiores a 3.3V podrían dañar la placa.

Algunas de las características más importantes de este Arduino se recogen a continuación en la siguiente tabla:

Tabla 2. Características Arduino Due

Microcontrolador	AT91SAM3X8E
Tensión de funcionamiento	3.3 V
Voltaje de entrada (recomendado)	7-12 V
Voltaje de entrada (límite)	6-16 V
Pines de entrada y salida digital	54
Pines de entrada y salida PWM	12
Pines de entradas analógicas	12
Pines de salidas analógicas	2
Memoria flash	512 KB
Velocidad del reloj	84 Mhz
Número de leds	13
Largo	101.52 mm
Ancho	53.3 mm
Peso	36 g

3.1.3 Arduino Mega 2560

El Arduino Mega 2560 (Fig. 3.5) es una placa de microcontrolador basada en el ATmega2560. Tiene 54 pines de entrada/salida digital (de los cuales 15 se pueden usar como salidas PWM), 16 entradas analógicas, 4 UART, un reloj de 16 MHz, una conexión USB, un conector de alimentación, un conector CSP y un botón de reset.

Figura 3.5 Arduino Mega 2650

Algunas de las características más importantes de este Arduino se recogen a continuación en la siguiente tabla:

Tabla 3. Características Arduino Mega 2650

Microcontrolador	ATMega2560
Tensión de funcionamiento	5 V
Voltaje de entrada (recomendado)	7-12 V
Voltaje de entrada (límite)	6-20 V
Pines de entrada y salida digital	54
Pines de entrada y salida PWM	15
Pines de entradas analógicas	16
Memoria flash	256 KB
Velocidad del reloj	16 Mhz
Número de leds	13
Largo	101.52 mm
Ancho	53.3 mm
Peso	37 g

3.2 ARDUINO IDE

El software gratuito Arduino IDE (Integrated Development Environment) se compone de una interfaz simple y sencilla para su fácil uso, consiguiendo que esta sea muy intuitiva para el uso de cualquier usuario.

3.2.1 Interfaz

En esta interfaz (Fig. 3.6), se pueden distinguir los siguientes elementos:

- Menú: Barra ubicada en la parte superior donde se encuentran diferentes herramientas relacionadas principalmente con ajustes de la programación.

- Botones de acceso rápido: Con estos intuitivos botones, podemos realizar rápidamente, de izquierda a derecha, verificar el código, subirlo, crear un nuevo sketch, abrir un sketch, guardar y abrir el monitor serie.

- Editor de texto: En esta ventana se mostrará el 'sketch' programado, es decir, el archivo de programación el cual cargaremos en nuestro Arduino.

- Area de mensajes: muestra el estado del sketch, si este está compilándose o compilado y si está subiéndose o subido a la placa.

- Consola: muestra en el momento de compilación de los posibles errores cometidos en la escritura del código de nuestro sketch. Además, cuando el código sea totalmente correcto, mostrará el tamaño que ocupará el sketch dentro de la memoria flash del microcontrolador de la placa.

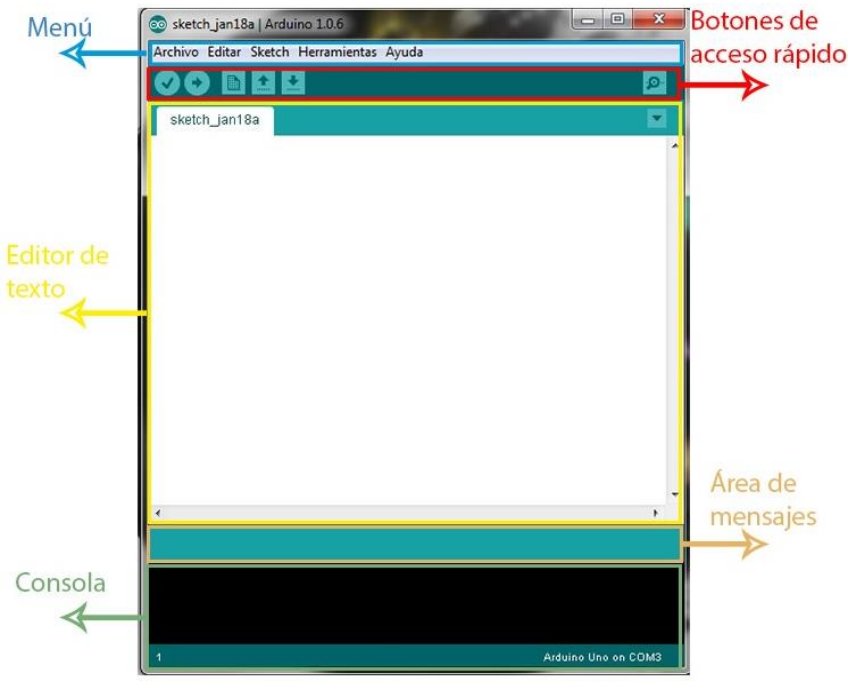


Figura 3.6 Elementos de la interfaz Arduino IDE

3.2.2 Estructura de un sketch

La estructura básica de un programa realizado en Arduino IDE consta al menos de dos partes donde se incluyen las diferentes declaraciones, estamentos e instrucciones.

Estos dos partes principales (Fig. 3.7) son las siguientes:

- setup(): es la parte encargada de la configuración. Solo se ejecutará una vez en todo el programa y se encargará de la configuración de pines y de la configuración de la comunicación, ya sea serie o UDP. Esta parte debe ser incluida en el programa, aunque no se realice ninguna declaración.

- loop(): contiene el programa que se ejecutará constantemente (cíclicamente). Esta parte es el núcleo del programa, por lo tanto, la más importante y la que más trabaja.

Figura 3.7 Partes principales de un sketch

Un sketch de Arduino no funcionaría sin al menos estas dos partes. Si fuese necesario el uso de alguna librería se añadirá en la parte superior de este. Además, toda la declaración de variables será escrita antes del setup(), justo después de las librerías.

Los sketch de Arduino únicamente tienen un tipo de extensión llamada “.ino”, por lo que este tipo de archivos son fácilmente reconocidos al solo existir uno.

Si a lo largo del sketch es necesario la inclusión de un comentario explicativo sobre el funcionamiento de alguna función o parte del código, este se puede añadir con la utilización de dos barras “//” antes del comentario. Este comentario será informativo para el usuario, el programa al ejecutarse lo ignora y no ocupa espacio en la memoria.

3.2.3 Librerías

Como bien se comentó antes, dentro del programa principal se pueden incluir librerías que incluyen funciones útiles para la programación, de las que el programa principal extrae la información necesaria para el correcto funcionamiento.

Al instalar Arduino IDE, se incluyen una serie de librerías estándar, que son aquellas que los creadores estimaron que serían muy utilizadas, entre las que se encuentran las librerías “SoftwareSerial”,” Servo”, “SPI” y ”Ethernet” entre otras.

También existe la posibilidad de descargar otro tipo de librerías que no vienen inicialmente con el software, por ejemplo, las creadas por otros usuarios del mundo. Para ello, podremos descargarlas directamente desde Internet e introduciéndolas en el directorio de carpetas destinado a las librerías donde se instaló Arduino IDE o utilizar el buscador de librerías integrado en el software.

Para cargar estas librerías en nuestro sketch, basta con añadir en la parte superior de nuestro sketch el comando “#include” seguido del nombre de la librería que debe ir limitado por los símbolos: < y >. Además, no es necesario incluir un punto y coma al final de esta como es norma en las demás líneas de código.

Las librerías tienen dos tipos de extensiones llamadas “.h” y “.cpp”, siendo más común la primera de estas.

3.2.4 Monitor serie

El monitor serie (Fig. 3.8) es una ventana en la cual te permite enviar y recibir datos de la placa mediante el puerto serie.

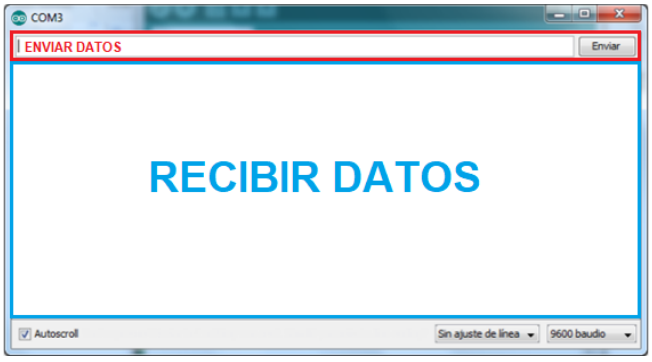


Figura 3.8 Monitor serie

Es necesario incluir en el setup() la función “Seial.Begin()”, donde entre los paréntesis incluiremos el número de bits por segundo llamados baudios. Una vez añadida esta función en nuestro programa, es necesario configurar el mismo número de baudios seleccionados en la ventana del monitor serie en la parte inferior derecha de esta.

3.3 MÉTODOS DE TRANSMISIÓN DE INFORMACIÓN

Para la transmisión de información y conexión de periféricos, el ordenador cuenta con 4 puertos USB 2.0 en su parte posterior y otros 2 en su parte frontal.

Mediante estos puertos podemos realizar una transmisión de información denominada Serie.

Por otro lado, en la parte trasera el ordenador cuenta con 2 puertos Ethernet mediante los cuales se puede realizar una comunicación UDP, dejando así libre los puertos USB para el uso de otros elementos.

3.3.1 Comunicación Serie mediante puerto USB

Este tipo de comunicación que es el más conocido y el más común será el utilizado para realizar las diferentes pruebas en nuestro trabajo. La placa Arduino Uno utilizada, cuenta con este tipo de conexión para la trasmisión de información con el que, a su vez, suministra 5V para dar tensión a la propia placa como se muestra en la Figura 3.9.

Figura 3.9 Arduino Uno conectado mediante USB

3.3.2 Comunicación UDP mediante puerto Ethernet

El UDP (User Datagram Protocol) es un protocolo basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento. No tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros y tampoco se conoce si ha llegado correctamente, ya que no exsite confirmación de recepción.

Mediante este tipo de conexión, el Arduino Uno y el ordenador pueden comunicarse, pero para ello es necesario el uso de un Ethernet Shield como el de la Figura. 3.10, ya que el propio Arduino Uno no dispone de este tipo de conexión.

Figura 3.10 Ethernet Shield para Arduino Uno

Este se insertará sobre el Arduino Uno, el cual posee el mismo número de conexiones para hacer coincidir uno sobre otro. De esta manera, conseguimos tener además de la comunicación Serial, comunicación UDP y no perderemos las diferentes entradas y salidas, puesto que el Ethernet Shield aloja en su parte superior las mismas conexiones que el Arduino Uno.

4 SISTEMA DE CONTROL

4.1 ESTRUCTURA GENERAL

Los sistemas de control realizan operaciones de mando con verificación y en algunos casos, de regulación. En estos sistemas, la señal de salida poder operar o no sobre el funcionamiento del sistema. Por esa razón, se pueden distinguir dos tipos de sistemas: sistema de lazo abierto y de lazo cerrado.

Los sistemas de lazo abierto son aquellos en los que la señal de salida no afecta al funcionamiento del sistema total. La salida del sistema solo depende de la entrada, por lo que no se controla si la salida es la deseada o no. En la práctica, el lazo abierto sólo se utiliza si se conoce bien la relación entre la entrada y la salida y si no hay perturbaciones internas ni externas.

Un ejemplo sencillo es al hacer una tostada, lo que hacemos es controlar el tiempo de tostado de ella misma indicándole el grado de tostado que queremos. En definitiva, el que nosotros introducimos como parámetro es el tiempo.

En cambio, los sistemas de lazo cerrado se tienen en cuenta la salida para actuar sobre la entrada. Para ello, se realiza una realimentación, en la cual se encuentra un sensor que es el encargado de medir la señal de salida y actuar sobre la señal de entrada, con el fin de reducir el error y llevar la salida del sistema a un valor conveniente.

Un ejemplo claro de un sistema de lazo cerrado es el que controla la temperatura de una habitación mediante un termostato. El termostato es un dispositivo que compara la temperatura de referencia seleccionada con la existente en la habitación; en caso de que ambas no sean iguales, genera una señal que actúa sobre el sistema de calefacción, hasta conseguir que la temperatura de la habitación se la misma que la de referencia.

En nuestro caso, como se desea seleccionar una velocidad y que el vehículo la adquiera con el menor error posible, se hizo uso de un sistema de lazo cerrado con realimentación (Fig. 4.1) donde se utilizó el sensor Hall que lleva la controladora, de la cual hablaremos más tarde y de un controlador PID.

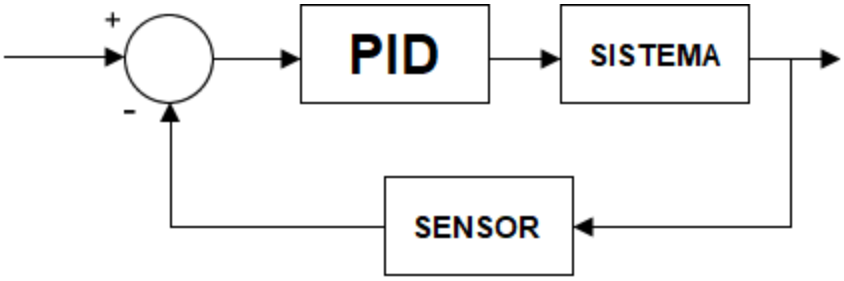


Figura 4.1 Sistema de lazo cerrado

4.2 CONTROLADOR PID

Un controlador PID permite controlar un sistema de lazo cerrado para que este alcance en su salida el estado deseado.

El PID está compuesto por tres elementos diferentes que proporcionan la acción Proporcional, Integral y Derivativa, dando con la primera letra de cada una de ellas nombre al controlador.

4.2.1 Acción proporcional

Como su propio nombre indica, la acción es proporcional a la señal del error *(1)*, es decir, la acción proporcional multiplica a la señal del error por una constante Kp.

$$P = Kp * e(t) \qquad (1)$$

Aumentar la Kp en nuestro sistema conlleva los siguientes efectos:

- Aumenta la velocidad de respuesta del sistema
- Disminuye el error del sistema en régimen permanente
- Aumenta la inestabilidad del sistema

Como se puede apreciar, los dos primeros efectos son positivos, mientras que el tercero no, por lo que hay que intentar minimizarlo al máximo. Para ello, existe un punto de equilibrio donde se consigue una suficiente rapidez de respuesta y reducción del error sin que el sistema se vuelva inestable.

En las gráficas de la Figura 4.2 se puede observar cómo va aumentando la velocidad de respuesta y disminuyendo el error sin que el sistema se vuelva inestable para unos valores bajos de Kp pero para un valor alto de Kp este se vuelve muy inestable.

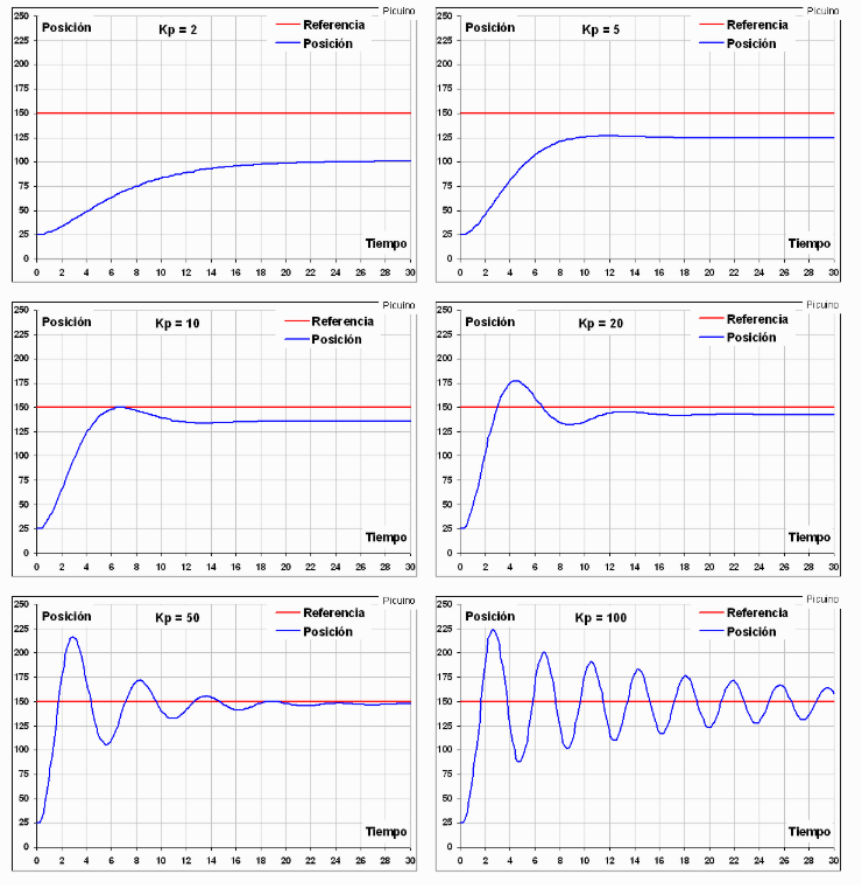


Figura 4.2 Respuesta al aumentar Kp

4.2.2 Acción derivativa

Como su propio nombre indica, es proporcional a la derivada de la señal de error *(2)*. Esta acción consigue estabilizar una respuesta que oscile demasiado.

$$D = K_D * \frac{d}{dt} e(t) \qquad (2)$$

Aumentar la constante derivativa Kd en nuestro sistema conlleva los siguientes efectos:

- Aumenta la estabilidad
- Disminuye un poco la velocidad del sistema
- El error en régimen permanente no se ve afectado

En las gráficas de la Figura 4.3 se puede observar cómo al aumentar Kd se aumenta la estabilidad del sistema a cambio de disminuir un poco la velocidad de este.

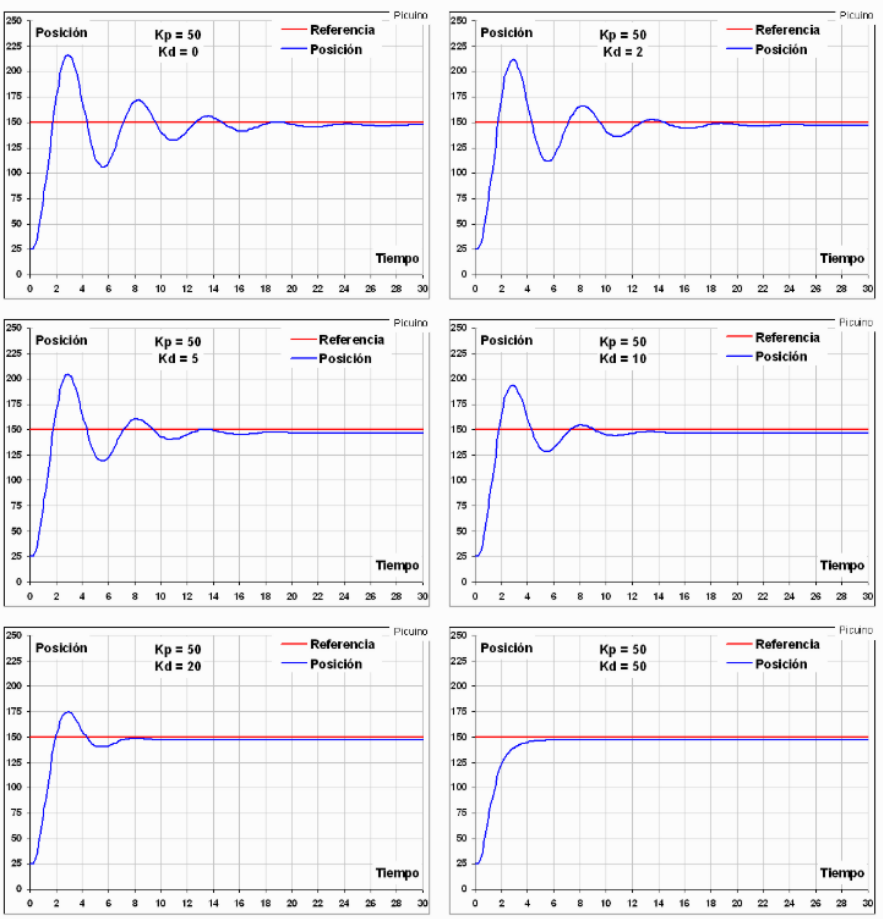


Figura 4.3 Respuesta al aumentar Kd

4.2.3 Acción integral

Como su propio nombre indica, esta acción calcula la integral de la señal del error *(3)*. La desventaja de esta acción es que hace el sistema más inestable debido a que añade una cierta inercia al sistema.

$$I = K_I \int_0^t e\,(\tau)\, d\tau \qquad (3)$$

Aumentar la constante derivativa Ki en nuestro sistema conlleva los siguientes efectos:

- Disminuye el error en régimen permanente
- Aumenta la inestabilidad del sistema
- Aumenta un poco la velocidad del sistema

En las gráficas de la Figura 4.4 se puede observar cómo al aumentar Ki se aumenta la inestabilidad del sistema, a cambio de hacer el sistema más rápido y disminuir el error, como se puede ver en la señal de color verde.

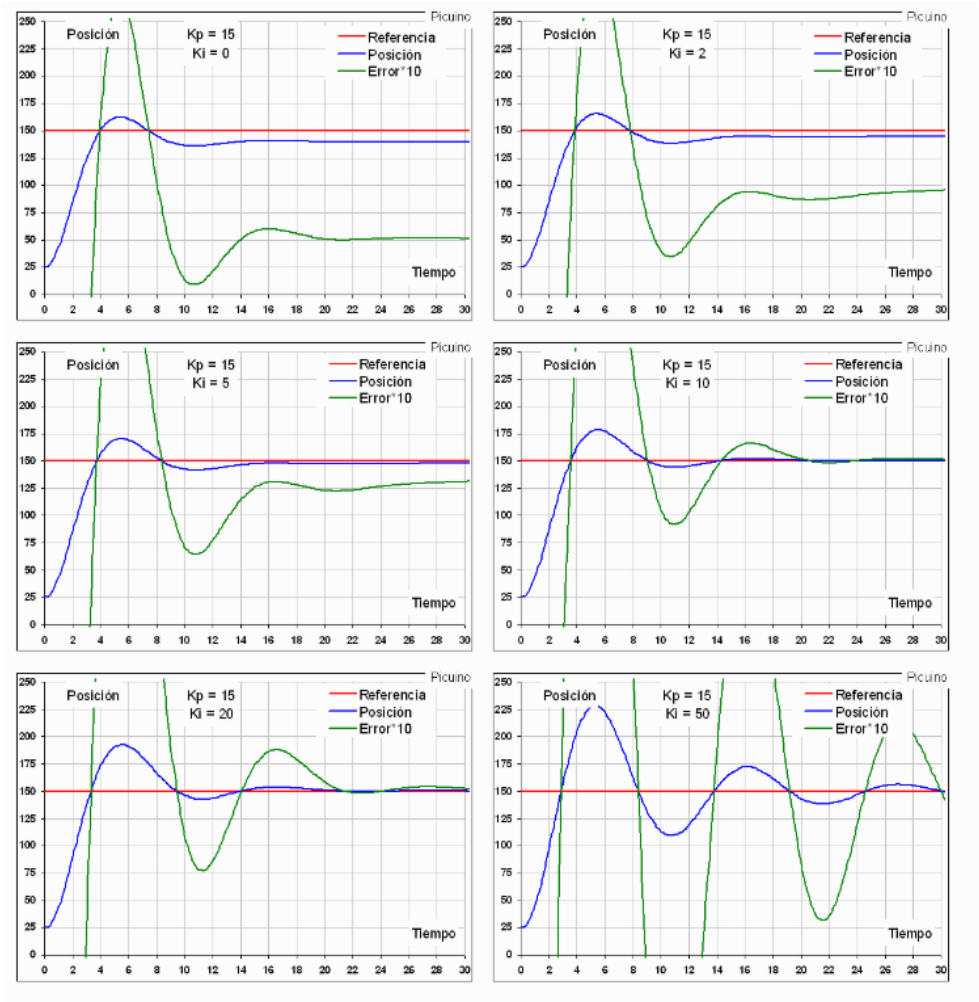


Figura 4.4 Respuesta al aumentar Ki

Finalmente, si sumamos las tres acciones de control (Fig. 4.5) anteriormente descritas obtenemos un controlador PID, aunque no es necesario en todos los casos la fusión de estas tres como podremos comprobar en el presente trabajo, existiendo así controladores P, I, D, PI y PD.

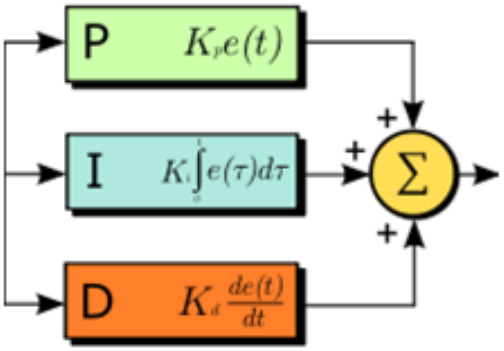


Figura 4.5 Suma de las tres acciones

Por lo tanto, para un PID, obtenemos la siguiente expresión:

$$PID = Kp * e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D * \frac{d}{dt} e(t) \quad (4)$$

También es común encontrarnos con la expresión del PID *(5)* en el dominio de Laplace.

$$PID(s) = Kp + \frac{K_I}{s} + K_D * s \quad (5)$$

4.2.4 Programación del controlador

A la hora de programar nuestro controlador en Arduino, se hizo uso de las expresiones *(6), (7)* y *(8)* para el cálculo de las diferentes acciones que lo constituyen en su forma discreta.

$$P_k = Kp * e_k \quad (6)$$

$$I_k = I_{k-1} + K_I * \left(\frac{e_k + e_{k-1}}{2}\right) * Ts \quad (7)$$

$$D_k = K_D * \left(\frac{e_k - e_{k-1}}{Ts}\right) \quad (8)$$

Una vez calculadas las tres partes, se sumaron y se normalizaron, dando lugar a un valor entre -1 y 1.

El cálculo del controlador PID se realizará cada 0.1 segundos para tener una velocidad de refresco alta, por lo que nuestro tiempo de periodo será este.

Al ser un sistema no lineal, se optó por realizar un ajuste manual de las tres componentes que lo conforman llegando a la siguiente combinación de las tres acciones:

$$\mathrm{Kp} = 0.4 \qquad \mathrm{Ki} = 0.1 \qquad \mathrm{Kd} = 0 \tag{9}$$

La acción derivativa no fue necesaria para obtener un buen resultado, por lo que finalmente nuestro regulador será PI.

Estos valores fueron los elegidos estando el vehículo elevado, es decir, las ruedas no se encuentran sobre el suelo, por lo que una vez bajado a tierra, estos valores habría que recalcularlos ya que existen diferentes factores que afectan como puede ser la fricción con el asfalto.

4.3 ELEMENTOS PARA CONSEGUIR UN VEHÍCULO AUTÓNOMO

Como se comentó anteriormente, el objetivo del departamento TEISA es la modificación y creación de este vehículo eléctrico en uno totalmente autónomo, es decir, que no sea necesaria la manipulación de un pasajero para la conducción. Para ello, el vehículo cuenta con diferentes elementos para conseguir este objetivo final.

4.3.1 Volante

El volante de nuestro vehículo es un volante convencional a simple vista, pero cuenta en su parte trasera con un motor instalado junto a un encoder (Fig. 4.6), conectado a su vez a una controladora, lo que permite la automatización del giro del vehículo, consiguiendo así que el vehículo se adapte al entorno y gire según proceda mediante programación, pero esto queda fuera de nuestro ámbito de trabajo.

Figura 4.6 Motor del volante

4.3.2 Pedal de acelerador y freno

Al tratarse de un vehículo eléctrico sin marchas, este dispone únicamente de dos pedales, el acelerador y el freno.

Como ya se comentó anteriormente, el objetivo de este trabajo es actuar sobre el acelerador de manera que el vehículo alcance una velocidad solicitada mediante programación.

El acelerador cuenta con un potenciómetro (Fig. 4.7) conectado al pedal y a la controladora del vehículo que más tarde veremos.

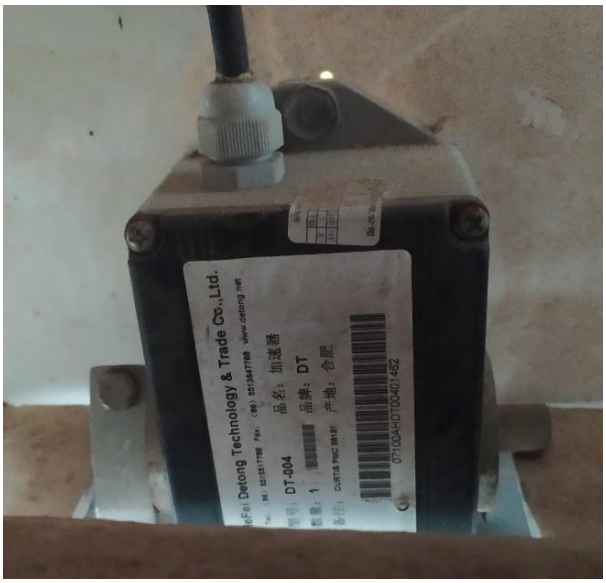


Figura 4.7 Potenciómetro del acelerador

El sistema de frenado del vehículo (Fig. 4.8) cuenta con dos tipos de frenos: frenos de tambor en el eje trasero y frenos de disco en el eje delantero.

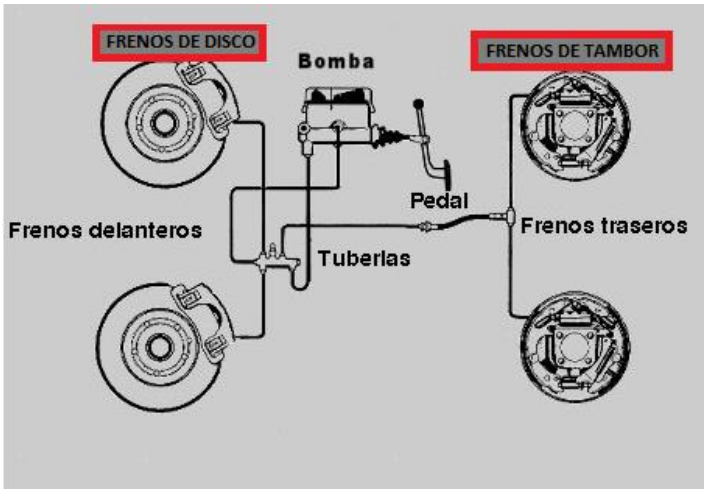


Figura 4.8 Sistema de frenado

Esta combinación de dos tipos de frenos en el mismo vehículo es muy habitual en la actualidad. Para entender mejor el porqué de ello, es necesario considerar que en la frenada existe una mayor carga de trabajo sobre el eje delantero que en el trasero debido al motor, por lo que los frenos de disco son colocados en el eje delantero dejando así en el eje trasero los de tambor, frenos de menor potencia, reduciendo así costes.

El objetivo sobre el freno es que, junto al sistema de control de velocidad del vehículo, este se accione cuando fuera necesario una disminución de la velocidad mediante el uso de una central hidráulica conectada en paralelo al circuito manual actual, por ejemplo, al detectar una curva cerrada, pero esto queda fuera del ámbito de este trabajo.

Para conseguir esta total autonomía del vehículo, este cuenta además con un sensor LIDAR, un radar y una cámara.

4.3.3 Sensor LIDAR

El vehículo cuenta con un sensor LIDAR actualmente desconectado. LIDAR proviene del nombre “Laser Imaging Detection and Ranging” y se trata de un foco emisor de haces de rayos infrarrojos y de una lente receptora capaz de captar esos haces láser. Estos haces ‘rebotan’ sobre los objetos y la lente se encarga de captarlos.

Se obtiene así una serie de puntos que son procesados creando una imagen artificial y tridimensional del entorno, conociéndose para cada uno de estos puntos su posición y distancia hasta este.

4.3.4 Radar

Al igual que Waymo de Google, nuestro vehículo cuenta con un radar (Fig. 4.9) en su parte frontal para medir la velocidad del vehículo que se encuentre por delante de este.

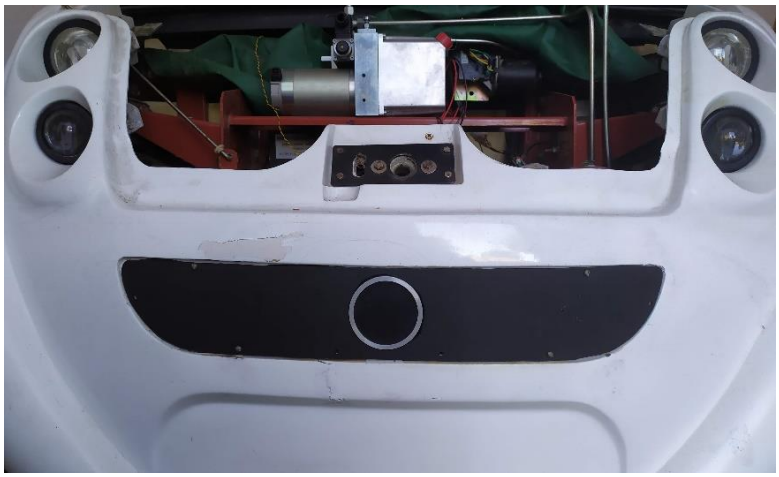


Figura 4.9 Radar frontal

4.3.5 Cámaras

El vehículo cuenta con diferentes cámaras (Fig. 4.10) previamente instaladas para tener un control del entorno que nos rodea, permitiéndonos, por ejemplo, la detección de personas, detección señales de velocidad y facilitar el estacionamiento del vehículo.

Figura 4.10 Cámara del retrovisor

4.4 ELEMENTOS DE ALIMENTACIÓN DEL VEHÍCULO

A continuación, se exponen los elementos que proporcionan la alimentación a nuestro vehículo.

4.4.1 Motor

El vehículo cuenta con un motor de corriente continua de 48V y 4.3kW de excitación Shunt (Fig. 4.11) aplicado sobre el eje delantero de este, es decir, tracción delantera. En este tipo de motores, la bobina de excitación se conecta en paralelo con el inducido y el devanado de los polos se conecta en serie con el inducido.

Figura 4.11 Motor del vehículo

4.4.2 Baterías

Para alimentar a este motor y los diferentes componentes eléctricos del vehículo cuenta con un total de 8 baterías (Fig. 4.12) de 6V 210Ah cada una, ubicadas bajo los asientos, proporcionándonos así un total de 48V. Tienen un tiempo de carga de aproximadamente 10 horas.

Figura 4.12 Baterías instaladas

4.4.3 Seta de emergencia

Por otro lado, existe una seta de emergencia de color naranja (Fig. 4.13) ubicada en la parte inferior de los asientos, junto a los pies, por si fuera necesario ante una situación de emergencia y descontrol del vehículo al realizar pruebas sobre él, cortar la corriente suministrada por las baterías, apagando por completo el vehículo experimental.

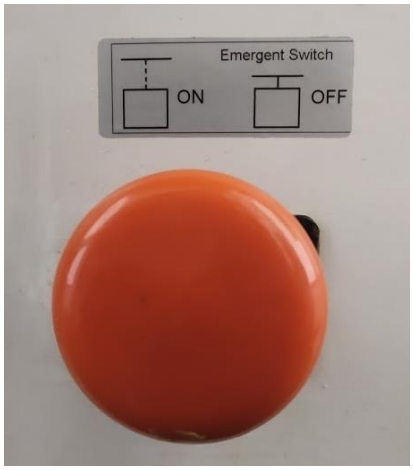


Figura 4.13 Seta de emergencia

4.4.4 Rectificador

Para cargar las baterías, el vehículo cuenta con un rectificador (Fig. 4.14) que transforma 220V de corriente alterna procedentes de un enchufe a 48V para la carga de estas.

Figura 4.14 Rectificador

Para ello, existe una toma de conexión (Fig. 4.15) ubicada debajo de la radio, donde se conectará un cable de cabezal azul y su otro extremo a una toma de corriente convencional de 220V.

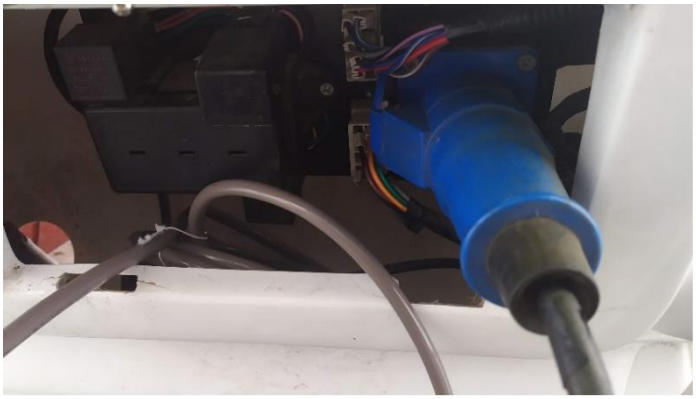


Figura 4.15 Toma de conexión de carga

4.4.5 Transformador

Con estos 48V y con la ayuda de un transformador (Fig. 4.16), reducimos la tensión a 12V para alimentar diferentes partes del vehículo como pueden ser la toma de mechero, claxon, luces, radio, etc.

Figura 4.16 Transformador

4.4.6 Inversor

Por último, y en cuanto a alimentación se refiere, se dispone de un inversor (Fig. 4.17) de corriente continua (48V) a corriente alterna (220V) para la conexión del ordenador en el interior del vehículo con las baterías que este dispone.

Figura 4.17 Inversor

5 SISTEMA DE VELOCIDAD: LA CONTROLADORA

Para la realización del sistema de control de velocidad haremos uso de la controladora Curtis 1268 (Fig. 5.1), la cual podemos decir que es el 'cerebro' del nuestro vehículo.

Esta controladora se encuentra instalada bajo el capó del vehículo junto al motor y demás elementos.

Figura 5.1 Controladora Curtis 1268

Dispone en su parte superior de 5 conexiones de alta corriente (Fig. 5.2) para su alimentación (B+ y B-), en este caso a 48V; conexión con la armadura del motor (M-) y conexión con el propio motor (F1 y F2).

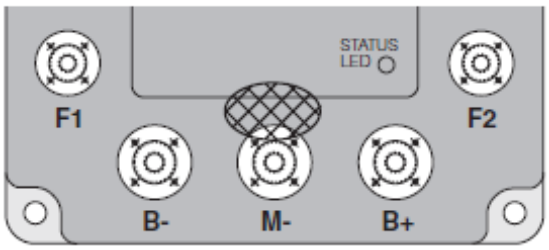


Figura 5.2 Conexiones de alta corriente

5.1 AJUSTES SOBRE LA CONTROLADORA

Inicialmente y sin ningún ajuste realizado por nuestra parte para la realización del presente trabajo la controladora presenta el conexionado que se puede observar en la Figura 5.3.

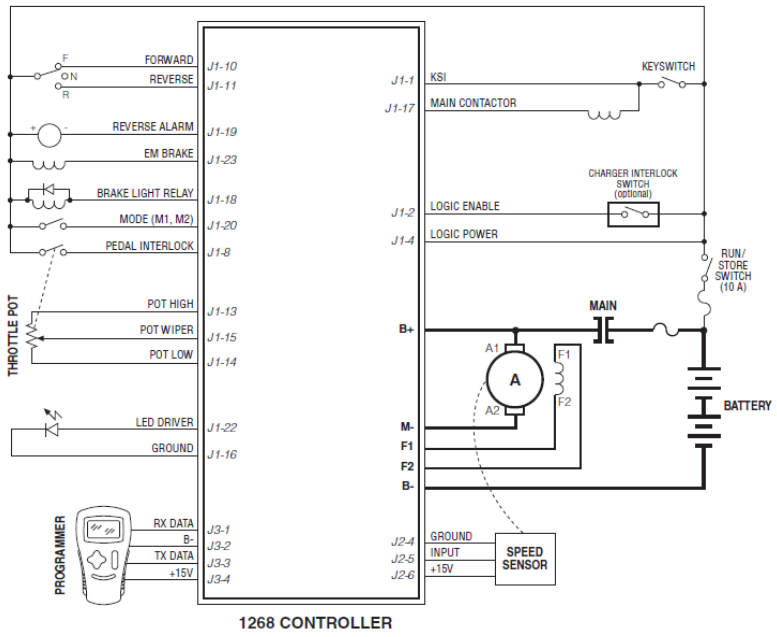


Figura 5.3 Conexiones de serie de la controladora Curtis 1268

A continuación, se exponen todos los ajustes necesarios a realizar sobre la controladora para conseguir nuestro objetivo final, un sistema de control de velocidad.

La controladora cuenta en su parte superior con tres partes diferenciadas de conexiones: J1, J2 y J3 (Fig. 5.4).

- El grupo de conexiones J1 cuenta con 24 pines dedicados señales de control.
- El grupo de conexiones J2 está formado por 6 pines para conocer la velocidad a la que gira el motor mediante un sensor Hall.
- El grupo de conexiones J3 es utilizado para cambiar los parámetros de la controladora mediante un programador conectado a los 4 pines que lo compone.

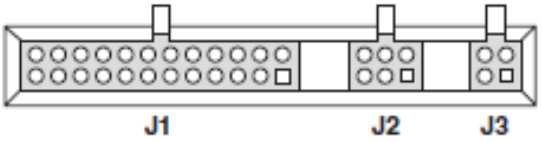


Figura 5.4 Conexiones de la controladora

5.1.1 Control de velocidad manual o autónomo

Como nuestro objetivo es poder controlar el movimiento del vehículo de manera autónoma, realizaremos los pertinentes cambios sobre la configuración de la controladora sin perder la

posibilidad del manejo manual del mismo mediante el pedal del acelerador. Para ello, se realizará un montaje en paralelo al actual para tener ambos modos, el autónomo y el manual (Fig. 5.5).

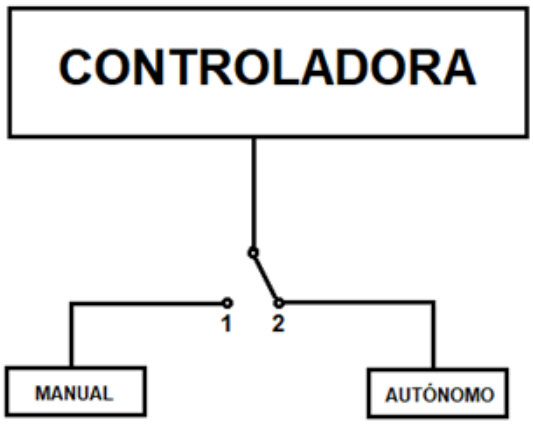


Figura 5.5 Esquema básico selección modo

Observando la hoja de características de la controladora utilizada, nos encontramos con los pines 13,14 y 15 (grupo de conexiones J1, ver Figura 5.6) encargados del control de la velocidad donde se encuentran los diferentes cables conectados hacia el potenciómetro del acelerador.

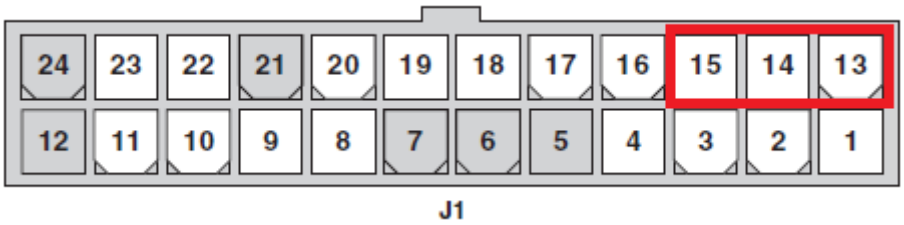


Figura 5.6 Pines de la controladora encargados de la velocidad

La función de los tres pines son las siguientes:

- J1-13: Pot High
- J1-14: Pot Low
- J1-15: Pot Wiper

El potenciómetro conectado al pedal de valor 5kΩ transforma una señal entre 0 y 5V, la cual es recibida mediante estos 3 pines, en una excitación del motor eléctrico entre 0 y el 100%. Este conexionado se puede observar esquemáticamente en la Figura 5.7.

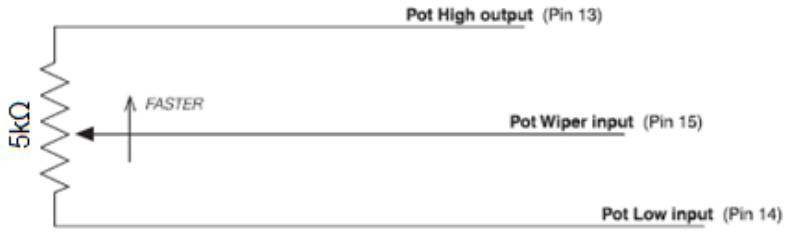


Figura 5.7 Esquema conexión pines con el potenciómetro

Como se comentó anteriormente, de los tres pines anteriores, se sacó un circuito en paralelo para obtener tanto el modo manual como el autónomo.

Con la ayuda de unas clemas, conseguimos el circuito en paralelo (Fig. 5.8), manteniendo la posibilidad de activar el modo manual o autónomo. Para ello, se instalaron un conjunto de tres interruptores de tres polos con dos posiciones posibles mediante los cuales si todos se encontraban en una misma posición se encontraba en un modo y viceversa.

Figura 5.8 Circuitos paralelos con clemas

Se utilizó como polo común de cada uno de estos interruptores cada uno de los tres cables extraídos anteriormente de los pines 13, 14 y 15 del grupo J1, en el polo de un lado los tres cables procedentes del potenciómetro y en el polo del otro lado dos cables, uno a tierra y otro a la salida del filtro PWM del cual hablaremos en el siguiente capítulo, quedando finalmente los interruptores como se puede ver en la Figura 5.9.

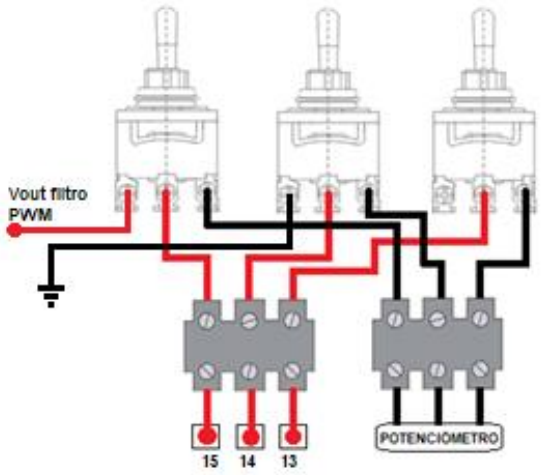


Figura 5.9 Configuración de los interruptores

5.1.2 Conocer la velocidad actual mediante el sensor hall

Para obtener la velocidad actual del vehículo se necesita obtener la señal del sensor Hall proporcionada por la controladora. Para ello, haremos uso de dos pines ubicados en el grupo de conexiones J2 (Fig. 5.10), los pines 4 y 5.

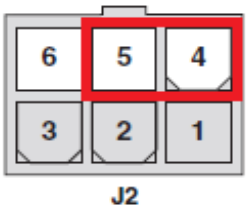


Figura 5.10 Pines del sensor Hall

La función de los dos pines son las siguientes:

- J2-4: Ground
- J2-5: Input

El pin 6 será el encargado de alimentar este sensor, pero no haremos ningún ajuste sobre él ya que ya se encuentra conectado a tensión.

De estos dos pines se sacaron dos cables en paralelo a los existentes con la ayuda de una clema (ver Fig. 5.8 clema 3) que irán conectaos a nuestro circuito del cual hablaremos más tarde.

Una vez realizado este ajuste sobre los dos pines anteriormente mencionados, se procedió al cálculo de la velocidad actual del vehículo.

Para ello, partimos del dato de que, en una vuelta de la rueda (Fig. 5.11), el sensor Hall de la controladora emite 80 pulsos.

Para el cálculo de velocidad también es necesario conocer la longitud de la rueda para conocer la distancia por cada vuelta, por lo que se utilizó una cinta métrica, obteniendo un valor de 176 cm, es decir, 1,76 m.

Figura 5.11 Rueda del vehículo

Conocido estos dos parámetros, podemos conocer la distancia recorrida en cada momento mediante la siguiente ecuación:

$$Distancia = \frac{pulsos}{pulsos/vuelta} * \frac{distancia}{vuelta}\ [m] \qquad (10)$$

Como bien sabemos, la velocidad es el cociente entre la distancia y el tiempo, por lo que podemos calcularla con la siguiente ecuación:

$$Velocidad = \frac{Distancia\ en\ metros}{Tiempo\ en\ segundos} * \frac{1\ km}{1000\ m} * \frac{3600\ segundos}{1\ hora}\ [km/h] \qquad (11)$$

Se aplicó una conversión para obtener la velocidad en km/h en vez de m/s, ya que en los vehículos esta no se usa.

Si sustituimos la distancia en esta última y el resto de las variables conocidas, obtenemos la ecuación *(12)* de la velocidad desglosada totalmente y con una única incógnita, los pulsos.

$$Velocidad = \frac{\frac{pulsos}{80/vuelta} * \frac{1{,}76\ m}{vuelta}}{0{,}1\ segundos} * \frac{1\ km}{1000\ m} * \frac{3600\ segundos}{1\ hora}\ [km/h] \qquad (12)$$

Está última ecuación, se ejecutará cada iteración del bucle una vez para conocer la velocidad actual del vehículo y realizar los pertinentes cálculos para alcanzar la velocidad deseada.

5.1.3 Parámetros programables de la controladora

Por último, y en cuanto a grupos de conexiones se refiere, la controladora dispone del grupo de conexiones J3 (Fig. 5.12) formado por 4 conexiones para la programación de una serie de parámetros.

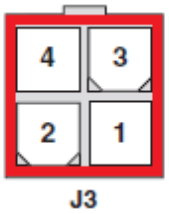


Figura 5.12 Pines de programación

La función de los pines anteriores son las siguientes:

- J3-1: Rx Data
- J3-2: B –
- J3-3: Tx Data
- J3-4: + 15V

A estos pines se deberá conectar un programador específico, del cual no disponemos, conocido como programador de mano 1311 (Fig. 5.13) de la propia marca Curtis.

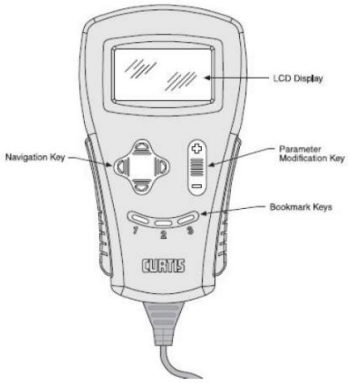


Figura 5.13 Programador Curtis 1311

Con la ayuda de este programador, es posible el ajuste de los siguientes parámetros extraídos de la hoja de características de la controladora:

Parámetros de aceleración

- La tasa de aceleración: define el tiempo en segundos que tarda el controlador en acelerar del 0% al 100%.

- La tasa de desaceleración: define el tiempo en segundos que el controlador tarda en reducir el voltaje promedio en la salida de la armadura de 100% PWM a 0% PWM, es decir, tiempo que tarda en frenar.

Parámetros de velocidad

- El parámetro de velocidad de avance define el límite de velocidad máxima del vehículo cuando se encuentra en marcha directa.

- El parámetro de velocidad de retroceso define el límite de velocidad máxima del vehículo cuando se encuentra en marcha atrás.

- El parámetro de conversión rpm a km/h es un factor de conversión usado para generar una estimación de la velocidad a partir de la entrada del sensor de velocidad Hall.

- El parámetro de polos del tacómetro es el número de polos que tiene el imán del sensor, ajustable entre 4 y 20.

Parámetros del acelerador

- El parámetro del 0% define el voltaje de entrada necesario a aplicar sobre el pedal del acelerador para que el motor comience a moverse.

- El parámetro del 100% define el voltaje de entrada necesario a aplicar sobre el pedal del acelerador para que el motor adquiera la máxima velocidad.

- El parámetro 'Throttle Map' (Fig. 5.14) modifica la respuesta del vehículo a la entrada del acelerador. Este parámetro determina la salida del controlador para una cantidad dada de aceleración aplicada. Para un valor de este del 50%, la respuesta en la salida será lineal.

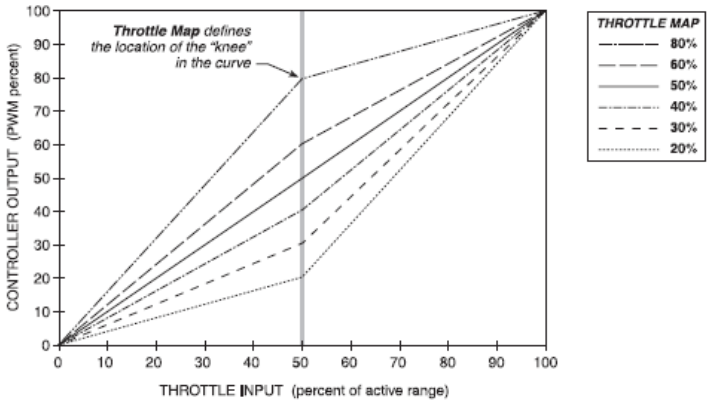


Figura 5.14 Gráfica Throttle Map

- Los umbrales de fallo de voltaje señalaran una falla en el acelerador cuando se aplique un voltaje por encima o por debajo de estos umbrales.

Parámetros de corriente

Diferentes parámetros relacionados con la corriente máxima aplicada al motor eléctrico en diferentes situaciones de funcionamiento.

Parámetros de mapeo del freno

Diferentes parámetros relacionados con la intensidad de frenado para las diferentes velocidades, es decir, para el freno regenerativo (Fig. 5.15).

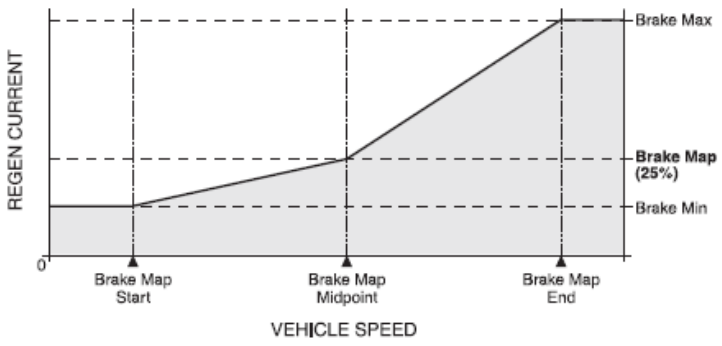


Figura 5.15 Gráfica del freno regenerativo

Parámetros de mapeo del campo

Estos parámetros determinan cuanta corriente de campo se aplica para una corriente de armadura dada (ver Figura 5.16).

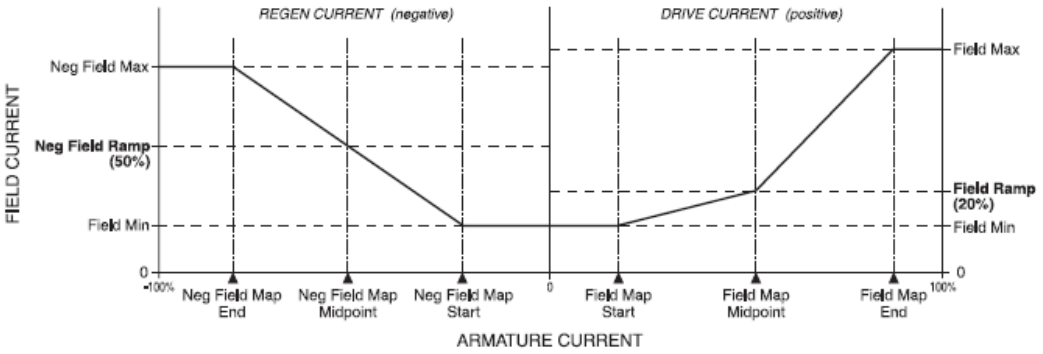


Figura 5.16 Gráfica del mapeo del campo

Parámetros de fallo

Parámetros relacionados con diferentes tipos de fallos con el objetivo de proteger el motor y evitar averías.

Como bien se comentó antes, no disponemos de este programador debido a su elevado precio, por lo que nos ajustaremos a los parámetros establecidos actualmente por el anterior dueño.

6 DESARROLLO

6.1 PWM

La modulación por ancho de pulsos (o en inglés las siglas PWM) de una señal es una técnica en la que se modifica el ciclo de trabajo de una señal periódica la cual tiene dos valores posibles de tensión: alto (HIGH) y bajo (LOW).

Existe también otro parámetro en las señales PWM que es el ciclo de trabajo (Duty Cycle en inglés), el cual es el porcentaje de tiempo que el pulso está en HIGH durante un ciclo.

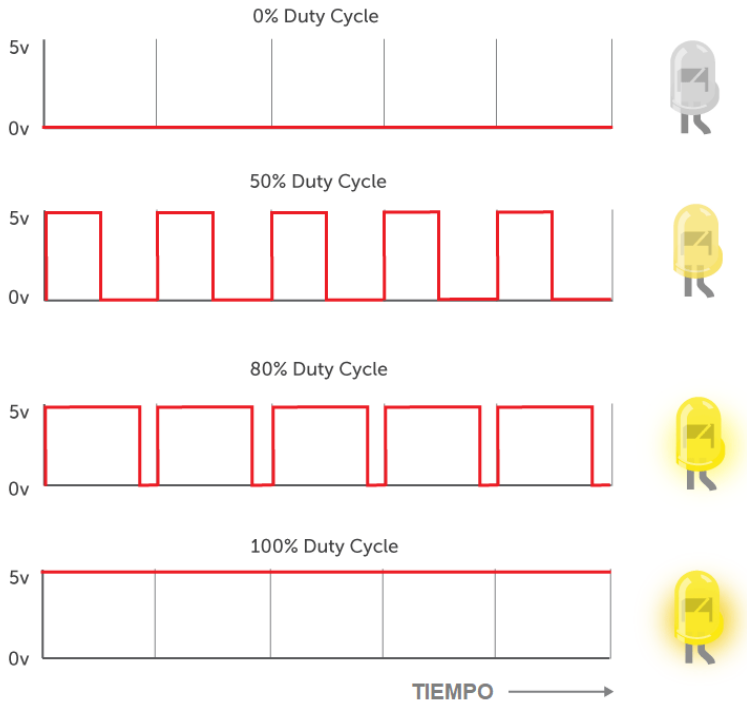


Figura 6.1 Entrada de control frente a la intensidad luminosa

Por tanto, el PWM es una técnica que consiste en variar el ancho de pulso de una señal de voltaje cuadrada con el objetivo de controlar la cantidad de potencia administrada, como se puede ver en la Figura 6.1. En este ejemplo, a medida que el ciclo de trabajo aumenta, la intensidad luminosa del led aumenta.

Como se comentó anteriormente, el Arduino no dispone de ninguna salida analógica que proporcione valores de tensión entre 0 y 5V para nuestro propósito de enviar una tensión a los pines de la controladora, pero es posible transformar una señal PWM en una tensión comprendida en este rango mediante el uso de un filtro RC y un amplificador operacional.

6.1.1 Transformación señal PWM en tensión

Se realizó el montaje de un filtro RC con un amplificador operacional LM358 en modo seguidor de tensión (Fig. 6.2).

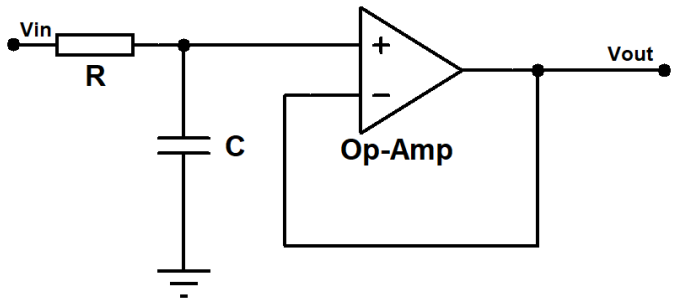


Figura 6.2 Filtro RC con amplificador

Un seguidor de tensión es aquel que tiene una ganancia de tensión de 1, es decir, el amplificador no realiza ninguna amplificación a la señal, por lo que la entrada y la salida es la misma.

El Arduino dispone de varias salidas PWM de 8 bits (Fig. 6.3) que ofrecen diferentes frecuencias:

- Pines 3, 9,10 y 11: frecuencia de 490 Hz
- Pines 5 y 6: frecuencia de 980Hz

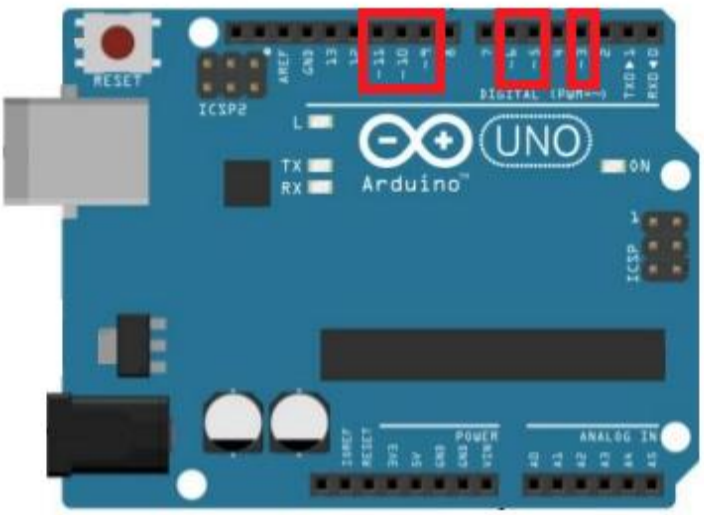


Figura 6.3 Salidas que admiten PWM

De estos pines ha sido usado el pin 9, por lo que la frecuencia es de 490 Hz.

Para el cálculo de los valores de la resistencia y del condensador se aplicó la siguiente fórmula:

$$fc = \frac{1}{2\pi RC} \quad (13)$$

Si tomamos como frecuencia de corte 15 Hz (frecuencia bastante alejada de la frecuencia ofrecida por el Arduino) y partimos de un condensador de 1 µF obtenemos que un valor de 10 kΩ para la resistencia. Para ello utilizaremos un potenciómetro de 10 kΩ.

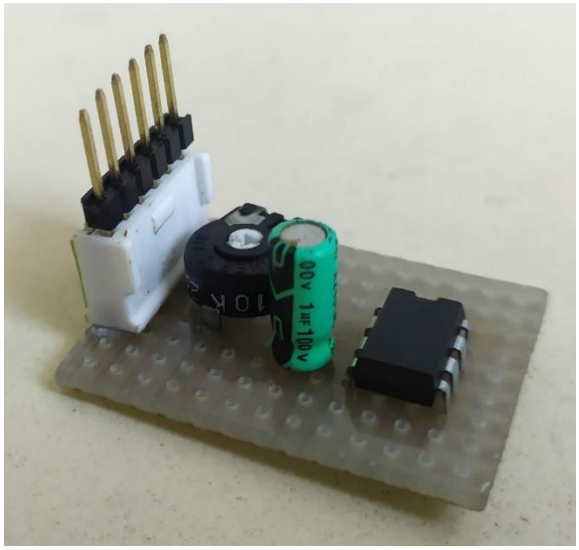


Figura 6.4 Montaje filtro RC con amplificador

Para mejor la precisión se optó por aumentar la frecuencia de la señal PWM. Como bien se dijo anteriormente, el pin 9 tiene una frecuencia de 490 Hz porque lleva de serie un prescaler, el cual podemos modificar para aumentar la frecuencia.

Haciendo uso de la hoja de características del microcontrolador del Arduino se obtuvo la siguiente información:

"Timer1. Asociado a los pines 9 y 10 con una frecuencia base de 31250 Hz con prescalers disponibles de 1, 8, 64, 256 y 1024."

Si se hace uso del Timer1, pueden existir errores de funcionamiento en algunas librerías, pero como nuestro programa no necesita el uso de ninguna no se tuvo ningún inconveniente para poder utilizarlo.

Dentro de la hoja de características del microcontrolador ATMega328 se localizó el registro del Timer1 (Fig. 6.5) necesario para el cambio del prescaler.

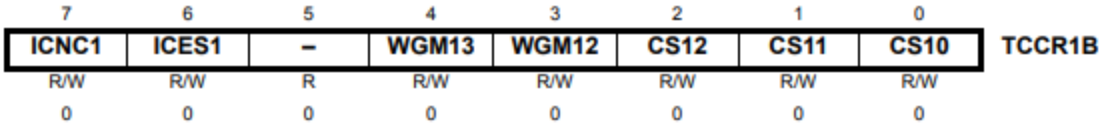


Figura 6.5 Registro B del Timer1

Atendiendo a los últimos tres dígitos de este, que corresponden al prescaler, se eligieron los correspondientes a un prescaler igual a 1.

Tabla 4. Selección del prescaler

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Para cambiar el prescaler a 1 del Timer1 fue necesario añadir el siguiente comando a nuestro sketch:

TCCR1B = 0b11111**001** *(14)*

6.1.2 Pruebas de funcionamiento

A continuación, se realizaron diferentes pruebas de funcionamiento para diferentes valores de la señal PWM comprendidos entre 0 y 255.

Mediante el cálculo de una regla de tres, se calculó que, para unos valores de 0 (0 %), 64 (25 %), 128 (50 %), 191 (75 %) y 255 (100 %) obtuvimos un tensión de 0,1.25, 2.5, 3.75 y 5 V respectivamente (ver Figuras 6.6, 6.7, 6.8, 6.9 y 6.10).

Con el uso de un osciloscopio, se configuró una escala de 5 V por cuadrado y se visualizó la señal PWM en el canal A y la tensión correspondiente a esta en el canal B.

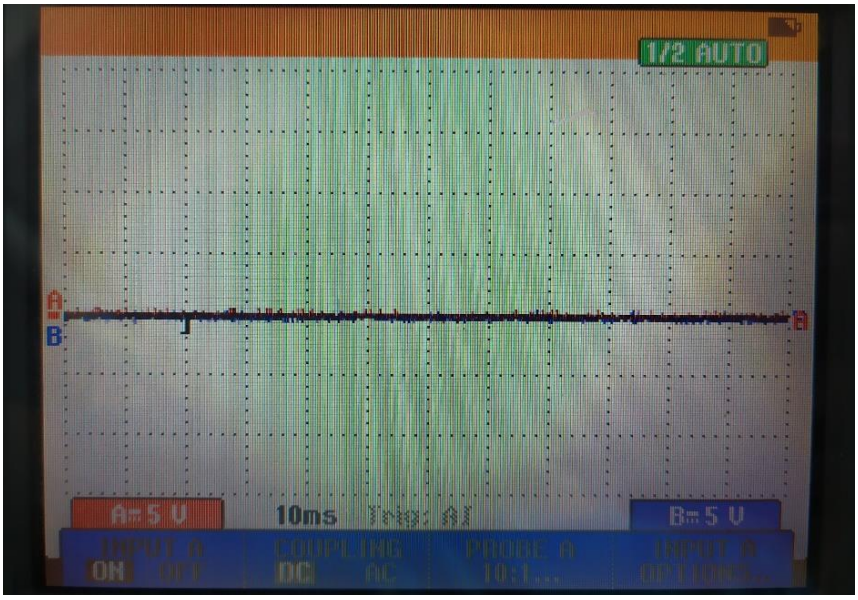


Figura 6.6 Señal PWM al 0 % y tensión de 0 V

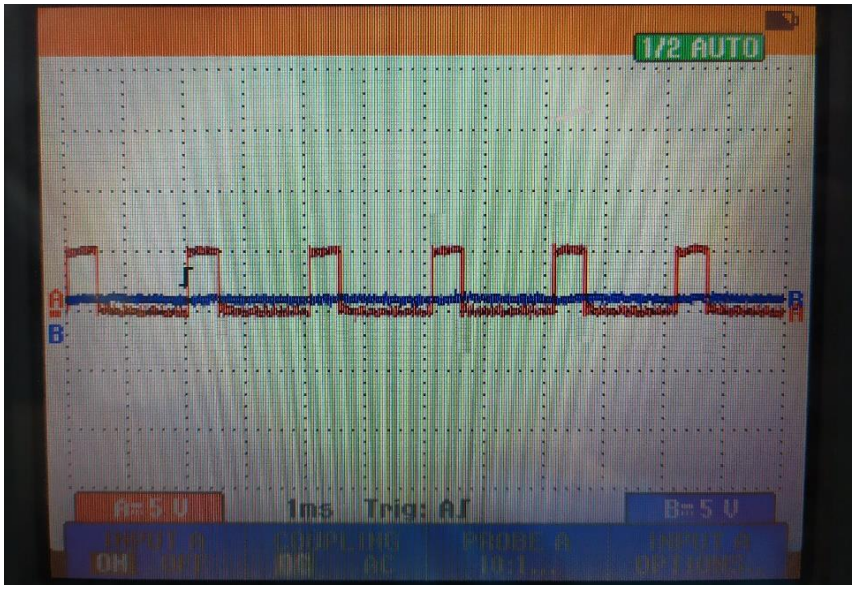


Figura 6.7 Señal PWM al 25 % y tensión de 1.25 V

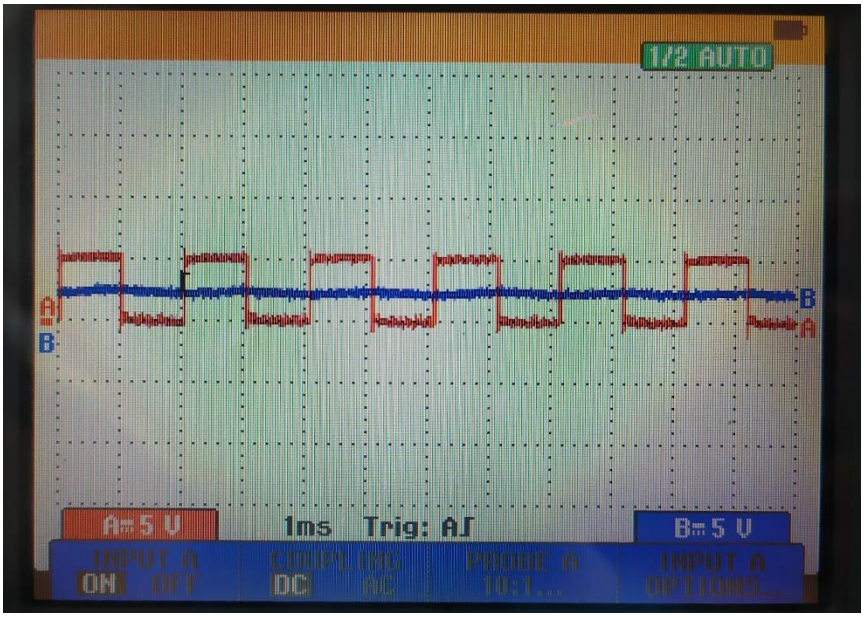


Figura 6.8 Señal PWM al 50 % y tensión de 2.5 V

Figura 6.9 Señal PWM al 75% y tensión de 3.75 V

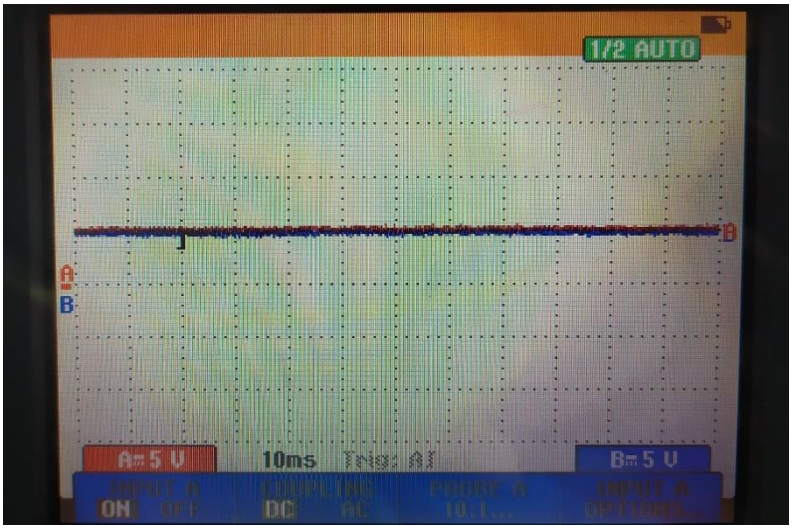


Figura 6.10 Señal PWM al 100 % y tensión de 5 V

Además, se colocó un polímetro en la salida del filtro y se comprobó que efectivamente, las tensiones eran las correctas.

Para la realización de estas pruebas se realizó un sketch de Arduino que se encuentra en *ANEXO 1. CÓDIGOS PARCIALES DE ARDUINO: Prueba del PWM.*

6.2 SENSOR HALL

Un sensor Hall recibe este nombre porque se basa en el efecto Hall, descubierto por Edwin Hall en 1879. Este se utiliza para la medición de campos magnéticos o para la determinación de la posición en la que está.

En la industria del automóvil es muy utilizado, por ejemplo, en sistemas de cierres de puertas, para el cambio de transmisión para el reconocimiento de posición del pedal o del asiento y para el reconocimiento del momento de arranque del motor.

En nuestro trabajo, el sensor Hall (Fig. 6.11) es el encargado de proporcionar una señal de pulsos de frecuencia proporcional a la velocidad del motor. Mediante estos pulsos se activa una interrupción en el sketch de Arduino.

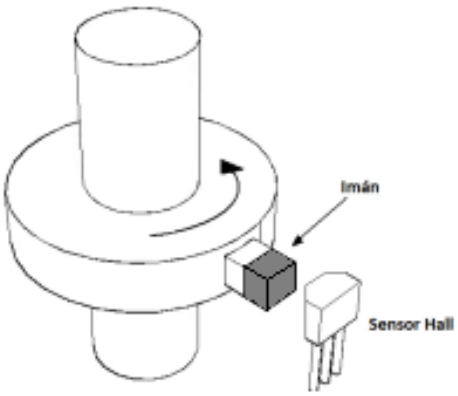


Figura 6.11 Sensor Hall

Una interrupción es una señal que realiza una llamada a nuestro microcontrolador para que este interrumpa su actividad y la atienda. Cuando esta es atendida, se ejecuta la función especial de interrupción llamada ISH (Interrupt Service Handler).

En nuestro caso, el Arduino Uno cuenta únicamente con dos pines que admiten la conexión de elementos de activación de interrupciones: el pin 2 y el 3.

En cuanto al tipo de activación o de disparo de la interrupción existen varios modos:

- LOW, La interrupción se dispara cuando el pin es LOW.
- CHANGE, Se dispara cuando pase de HIGH a LOW o viceversa.
- RISING, Dispara en el flanco de subida (Cuando pasa de LOW a HIGH).
- FALLING, Dispara en el flanco de bajada (Cuando pasa de HIGH a LOW).

En nuestro caso haremos uso del segundo tipo para una mayor precisión.

Para la llamada en nuestro sketch de la función especial de la interrupción será necesario incluir la siguiente línea de código:

```
attachInterrupt (interrupt, ISR, mode)
```

Donde:

- Interrupt: Número de la interrupción. Para saber qué número de interrupción estás asociada a un pin, debemos usar la función digitalPinToInterrupt(pin).
- ISR: Nombre de la función de las interrupciones
- Mode: Tipo de activación, en nuestro caso, CHANGE

Al utilizar interrupciones, debemos tener en cuenta lo siguiente:

- En las interrupciones, la función delay() y milis() no funcionan.

  - Los datos recibidos por el puerto serie se pueden perder mientras se está realizando la interrupción.

  - Se deben declarar como “volatile” cualquier variable que sea modificada dentro de la función llamada por una interrupción, por lo que la variable ‘pulsos’, que será el contador encargado de aumentar cada 0.1 segundos, será de este tipo.

Para comprobar el funcionamiento de la correcta lectura de los pulsos y, por lo tanto, que las interrupciones también se estén ejecutando, se hizo un sencillo montaje para comprobar si estos pulsos eran leídos correctamente por el microcontrolador y las interrupciones se realizaban correctamente.

Al probar su funcionamiento se descubrió que este no recibía datos correctos respecto a los pulsos, seguramente por el ruido de la señal emitida, lo que llevó a comprobarlo mediante el uso de un osciloscopio portátil de la marca Fluke modelo 199C (Fig. 6.12).

Figura 6.12 Osciloscopio Fluke 199C

Tras conectar el osciloscopio se comprobó que efectivamente la señal contenía mucho ruido, por lo que se decidió filtrar esa señal mediante un filtro (ver Figuras 6.13, 6.14, 6.15, 6.16).

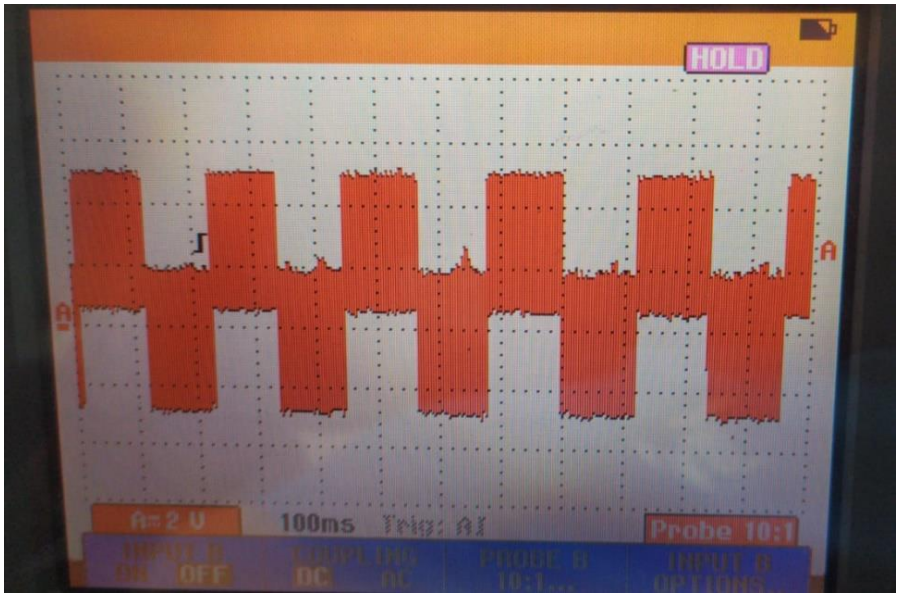


Figura 6.13 Señal de pulsos del sensor Hall al 25 %

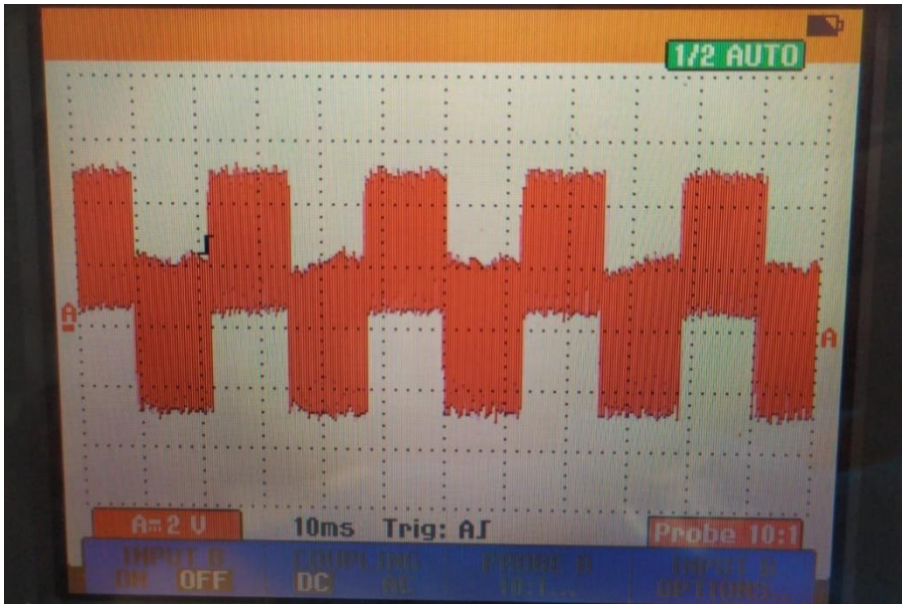


Figura 6.14 Señal de pulsos del sensor Hall al 50 %

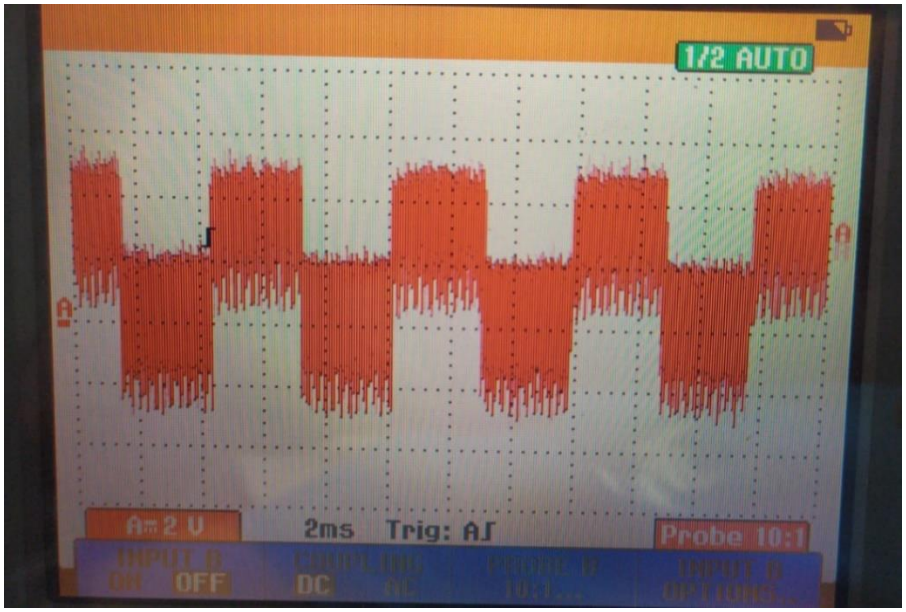


Figura 6.15 Señal de pulsos del sensor Hall al 75 %

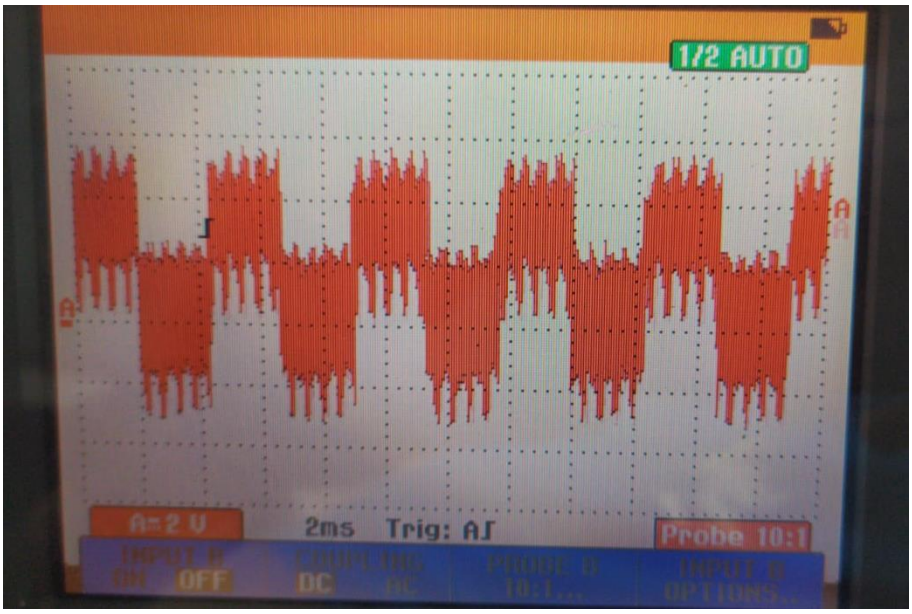


Figura 6.16 Señal de pulsos del sensor Hall al 100 %

6.2.1 Filtrado señal sensor Hall

Se eligió un filtro paso bajo de primer orden para el filtrado de los pulsos. La frecuencia mayor de la señal de pulsos es de 300 Hz, por lo que la frecuencia de corte debe estar por encima de esta aplicando la siguiente formula:

$$fc = \frac{1}{2\pi RC} \qquad (15)$$

Un filtro paso bajo (Fig. 6.17) es aquel que introducen poca atenuación a las frecuencias menores que la frecuencia de corte y las que se encuentren por encima de esta son atenuadas fuertemente. Una aplicación muy utilizada es la de eliminar ruidos como es en nuestro caso.

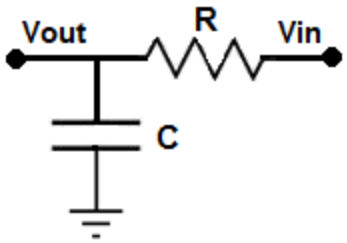


Figura 6.17 Filtro paso bajo

Si partimos de una frecuencia de corte de 50 kHz y de un condensador de 1 nF, obtenemos un valor para la resistencia de 3 kΩ, donde nuevamente utilizaremos otro potenciómetro que le ajustaremos a este valor mediante la muesca que tiene en su parte superior.

6.2.2 Pruebas de funcionamiento

Una vez realizado el montaje del filtro (Fig. 6.18) se incorporó sobre el circuito y mediante el uso del osciloscopio se pudo observar que la señal filtrada era correcta y el Arduino era capaz de reconocer los pulsos y así realizar las interrupciones correspondientes en la programación.

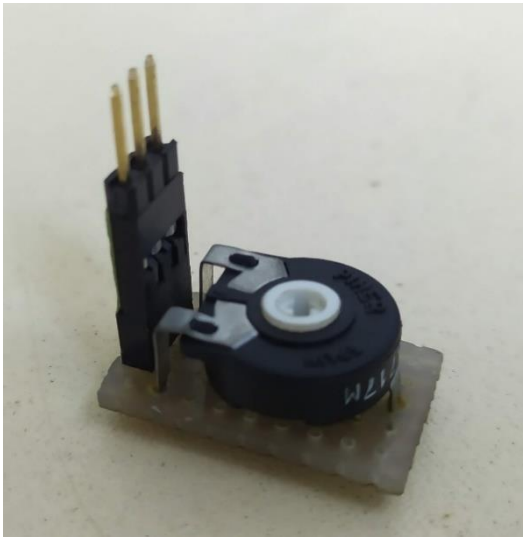


Figura 6.18 Montaje filtro paso bajo

Se realizaron pruebas para unos valores de 1.25 (25 %), 2.5 (50 %), 3.5 (75 %) y 5 V (100 %) aplicados sobre el potenciómetro y con un osciloscopio pudimos ver por el canal B la señal filtrada y por otro el A la señal sin filtrar (ver Figuras 6.19, 6.20, 6.21, 6.22).

Figura 6.19 Señal sin filtrar y filtrada al 25 %

Figura 6.20 Señal sin filtrar y filtrada al 50 %

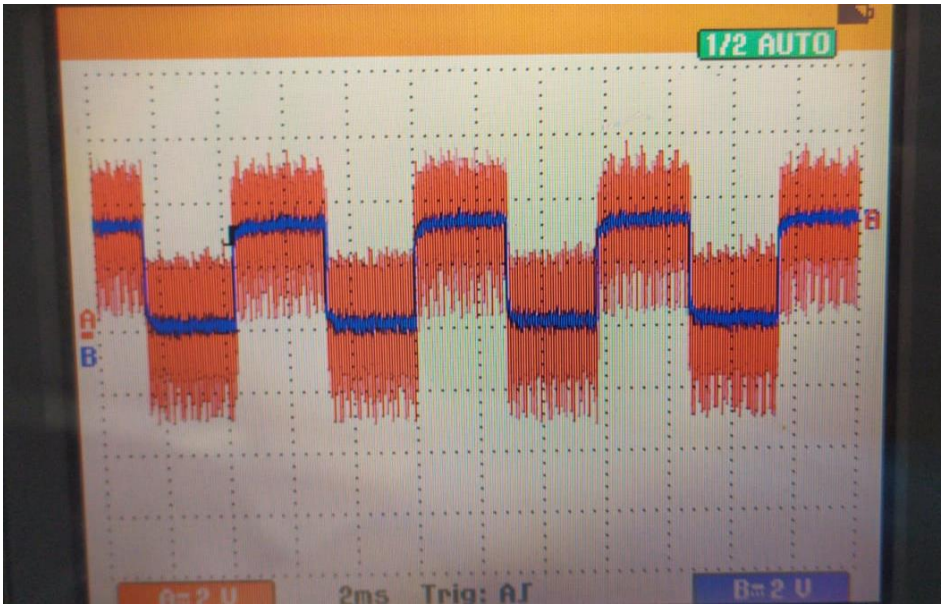


Figura 6.21 Señal sin filtrar y filtrada al 75 %

Figura 6.22 Señal sin filtrar y filtrada al 100 %

Para la realización de estas pruebas se realizó un programa de Arduino ubicado en *ANEXO 1. CÓDIGOS PARCIALES DE ARDUINO: Prueba del Sensor Hall.*

6.3 SISTEMA DE CONTROL DE VELOCIDAD FINAL

Una vez comprobado el correcto funcionamiento por separado de la señal emitida por el sensor Hall y la conversión de la señal PWM a tensión en los subapartados previos, se procedió a la unión de ambas partes dando como resultado el montaje de la Figura 6.23.

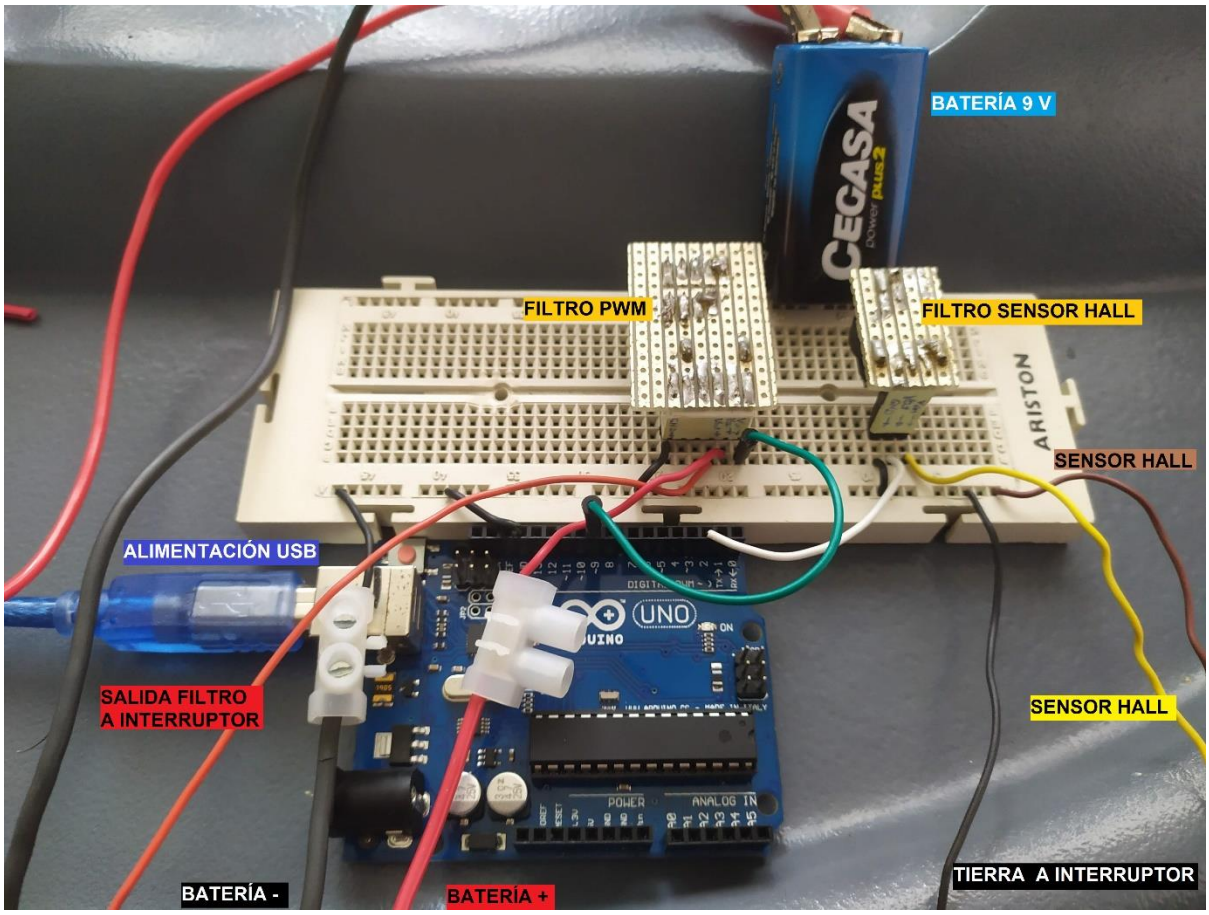


Figura 6.23 Montaje resultante del Sistema de Control de Velocidad

Con el vehículo encendido, se realizaron diferentes pruebas a diferentes velocidades para comprobar su funcionamiento, obteniendo un aumento progresivo sin movimientos bruscos de la velocidad hasta alcanzar la misma velocidad en el marcador digital integrado en el vehículo que la velocidad deseada introducida por el usuario en el sketch de Arduino que se encuentra en *ANEXO 2. CÓDIGO ARDUINO FINAL DEL SISTEMA DE CONTROL DE VELOCIDAD.*

7 MONTAJE DE CIRCUITOS

7.1 MONTAJE PRUEBA PWM

El montaje que se encuentra realizado a continuación en la Figura 7.1 muestra las conexiones y componentes necesarios para la comprobación del correcto funcionamiento de la transformación de la señal PWM emitida por el Arduino Uno a través del pin 9 al pasar por el filtro RC con amplificador.

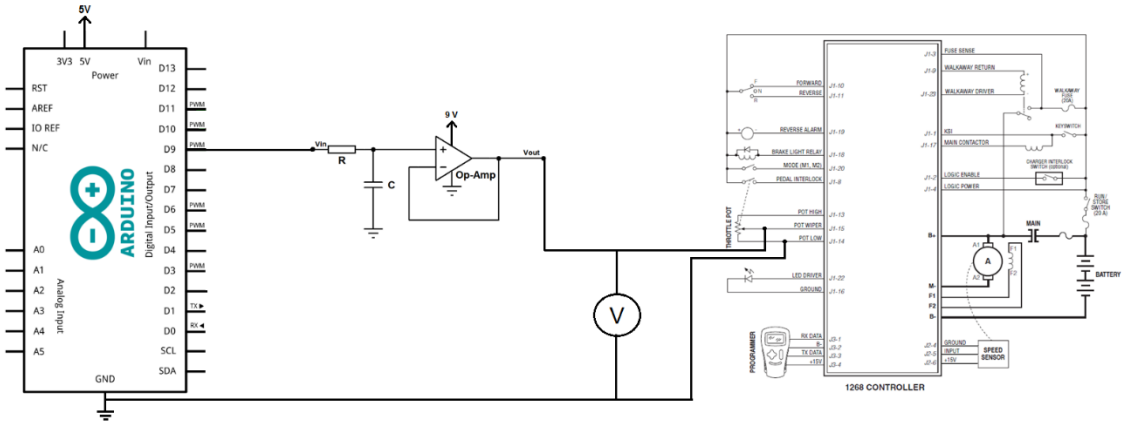


Figura 7.1 Esquema del montaje para prueba de la transformación PWM a tensión

Cabe destacar que para el funcionamiento del amplificador es necesario conectar este a una fuente para alimentarlo, por lo que se utilizó una pila de 9V.

7.2 MONTAJE PRUEBA SENSOR HALL

Con el siguiente circuito mostrado en la Figura 7.2 se pudo comprobar como el Arduino mostraba por el monitor serie de Arduino IDE los pulsos emitidos por el sensor Hall previamente filtrados con el filtro RC paso bajo para la eliminación del ruido que este poseía para su correcta identificación por parte del Arduino a través del pin 2.

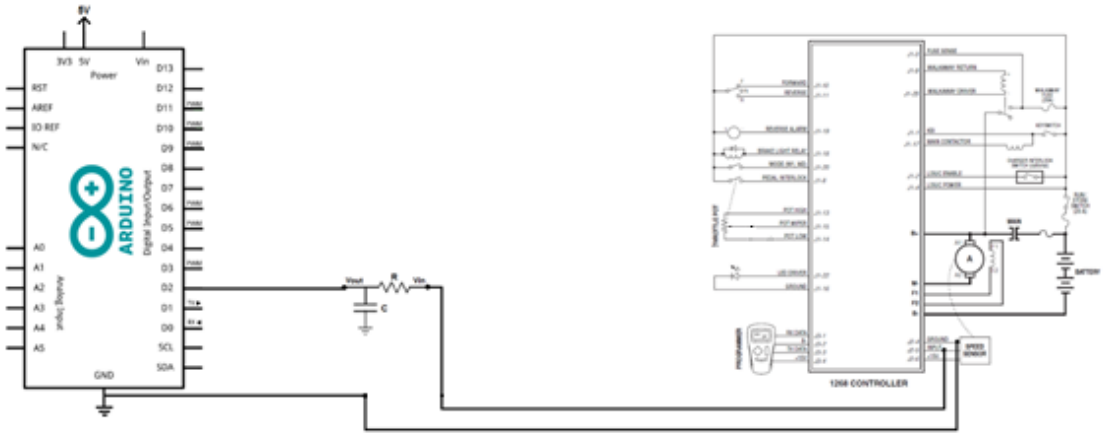


Figura 7.2 Esquema del montaje para prueba del sensor Hall

7.3 MONTAJE FINAL DEL CONTROL DE VELOCIDAD

Finalmente, se expone el montaje final (Fig. 7.3) para el correcto funcionamiento del sistema de control de velocidad para nuestro vehículo experimental, fruto de la fusión de los montajes parciales vistos anteriormente.

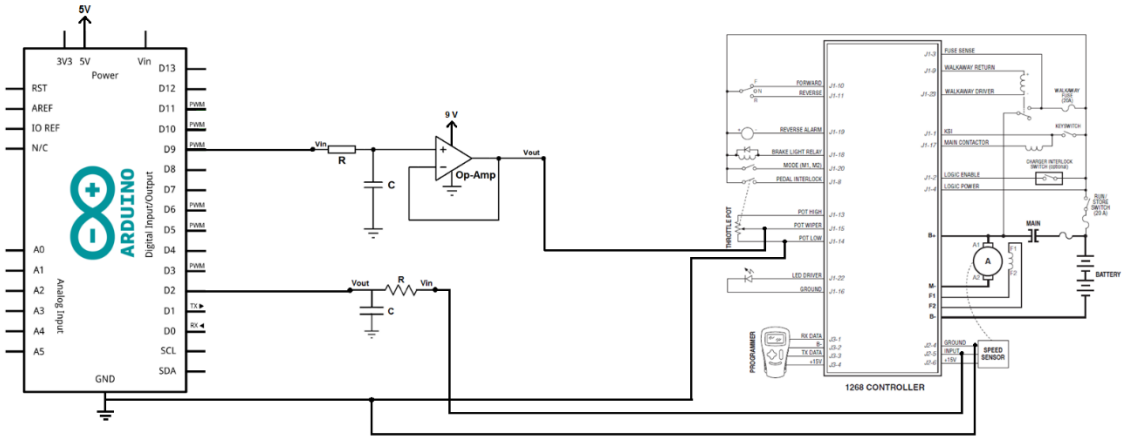


Figura 7.3 Esquema del montaje final del Sistema de Control de Velocidad

8 MARCADOR INTERACTIVO

Una vez implementado y comprobado el correcto funcionamiento del sistema de control de velocidad sobre el vehículo, se procedió a la realización de un marcador interactivo donde el usuario puede consultar información sobre la conducción mediante el software Processing.

8.1 PROCESSING

Processing es un lenguaje de programación y un software gratuito de código abierto basado en Java, utilizado comúnmente para la elaboración de proyectos multimedia e interactivos. Fue desarrollado en 2001 por Ben Fry y Casey Reas, miembros del departamento Media Lab de MIT.

Figura 8.1 Logotipo de Processing

8.1.1 Interfaz

Processing se compone de una interfaz simple y sencilla para su fácil uso, muy similar a la de Arduino, consiguiendo que esta sea muy intuitiva para el uso de cualquier usuario.

En esta interfaz (Fig. 8.2), se pueden distinguir los siguientes elementos:

- Menú: Barra ubicada en la parte superior donde se encuentran diferentes herramientas relacionadas principalmente con ajustes de la programación.

- Botones de acceso rápido: Con estos intuitivos botones, podemos rápidamente ejecutar el código, pararlo y cambiar el modo de programación, en nuestro caso Java.

- Editor de texto: En esta ventana se mostrará el ‘sketch’ programado, es decir, el archivo de programación realizado.

- Area de mensajes: muestra información sobre el sketch, por ejemplo, si este se ha guardado o cargado correctamente.

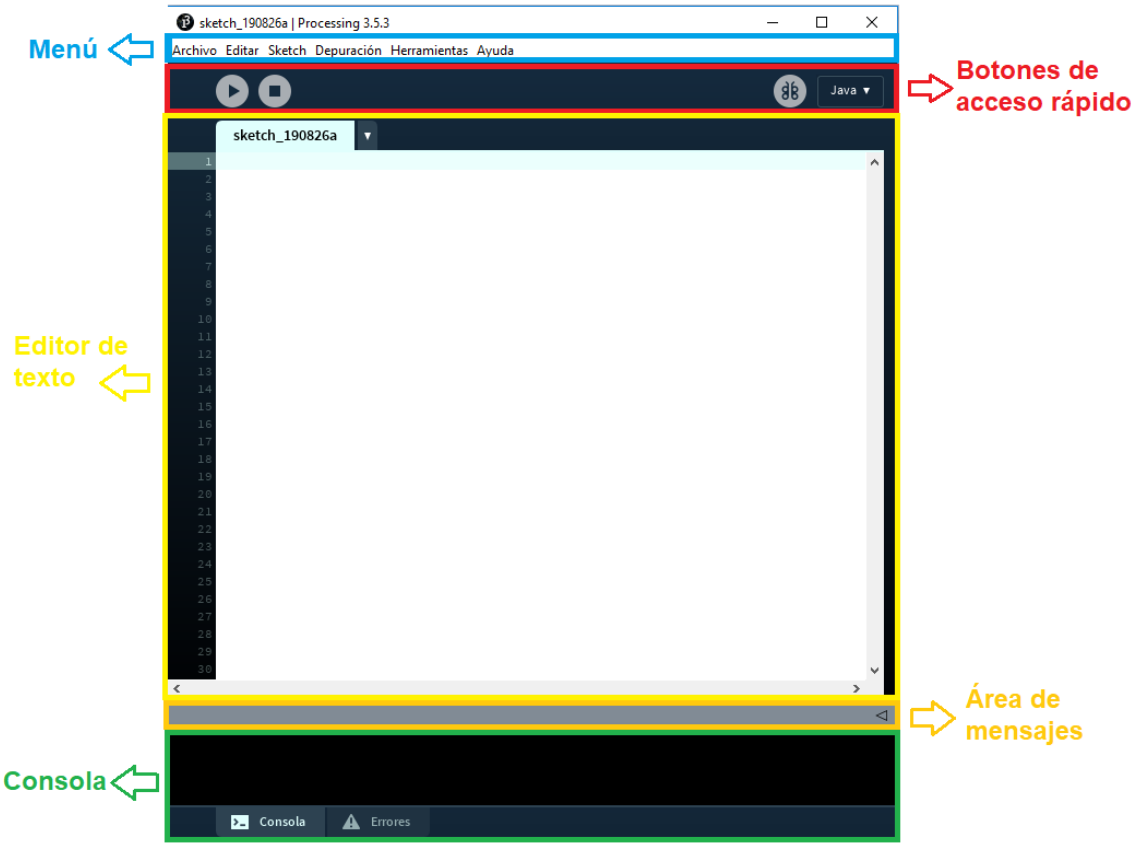


Figura 8.2 Elementos de la interfaz Processing

8.1.2 Monitor visual

Una vez generado un sketch, este se puede ejecutar y mostrar su resultado mediante el monitor visual (Fig 8.3), donde podremos ver contenido multimedia e incluso interactuar con el mediante el teclado o ratón.

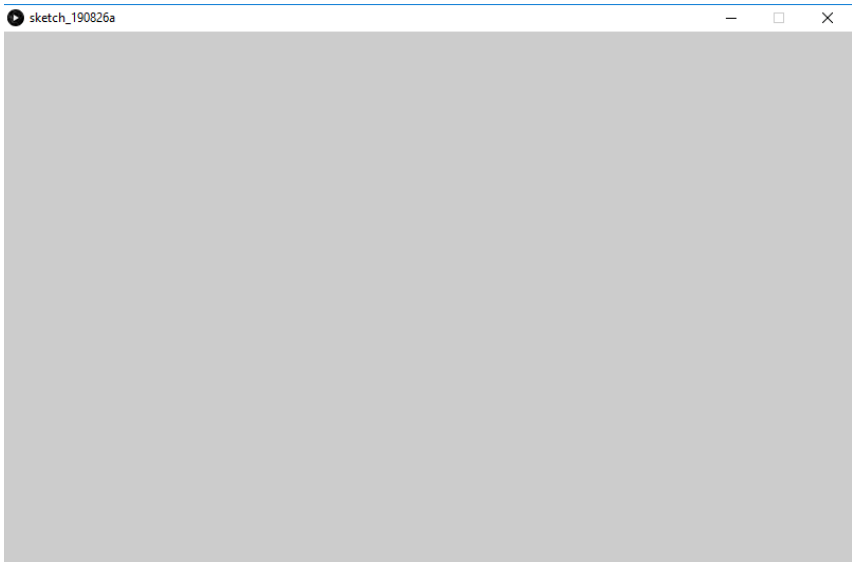


Figura 8.3 Monitor visual

8.2 ELEMENTOS

El marcador interactivo (Fig. 8.4) realizado mediante Processing está compuesto por diferentes elementos que proporcionan información al pasajero en la conducción, como las que nos puede mostrar un vehículo convencional en el marcador de este.

8.2.1 Cuentakilómetros

En la parte central se muestra una esfera en la que se muestra la velocidad actual del vehículo.

8.2.2 Batería

Para conocer la carga de las baterías, se ha integrado un elemento el cual en función de la carga de las baterías se muestra el nivel de estas, distinguiéndose el nivel de carga por tres colores:

- Rojo: La carga se encuentra por debajo del 20 %
- Naranja: La carga se encuentra entre el intervalo [20,30) %
- Verde: La carga es superior al 30%

Por otro lado, si el vehículo se encuentra cargándose, se mostrará un rayo en la parte central de la batería para indicarlo.

8.2.3 Testigos de luz

Se ha incorporado una serie de testigos de luz informativos. De izquierda a derecha, luces cortas, luces largas, triángulo de emergencia, luces delanteras antiniebla y trasera antiniebla. Debajo de las marchas, se encuentra otro testigo para el freno de mano.

8.2.4 Intermitentes

En la parte superior, a cada lado del cuentakilómetros, se colocaron dos flechas para indicar si los intermitentes están activados o no. Si estos no lo están, se mostrará una flecha apagada y si está activo, la flecha se volverá verde.

8.2.5 Marchas

Como el coche dispone únicamente de marcha directa y marcha atrás, se incorporó al lado derecho del cuentakilómetros una señalización de la marcha en la que se encuentra el vehículo mediante dos letras que se encienden o se apagan, una “D” para la marcha directa y una “R” para la marcha atrás.

8.3 RESULTADO Y PRUEBAS DE FUNCIONAMIENTO

Una vez creados los diferentes elementos por separado, se juntaron todos únicamente en un sketch.

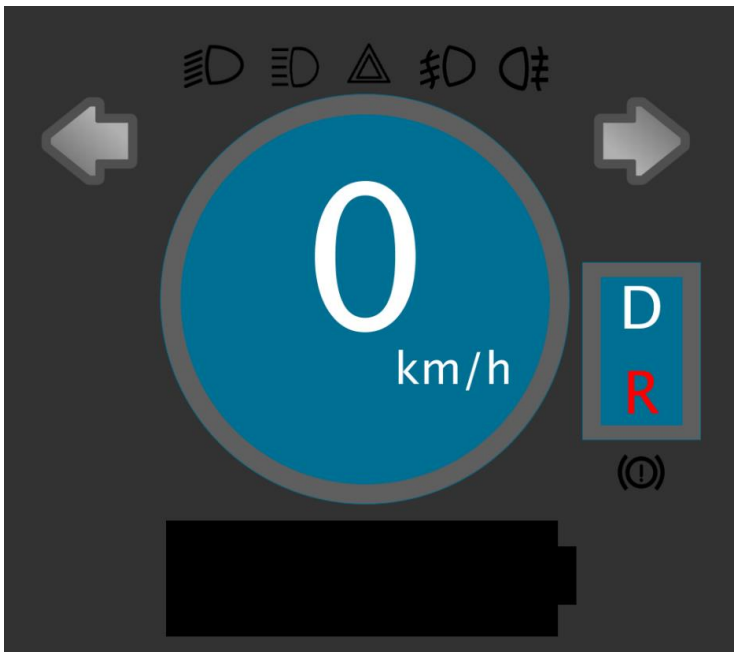


Figura 8.4 Marcador interactivo

A continuación, se muestran una serie de pruebas con diferentes valores para los distintos elementos que lo componen:

Prueba 1

- Intermitente izquierdo encendido
- Luces cortas y largas
- Batería al 50%
- Velocidad actual de 37 km/h
- Marcha directa

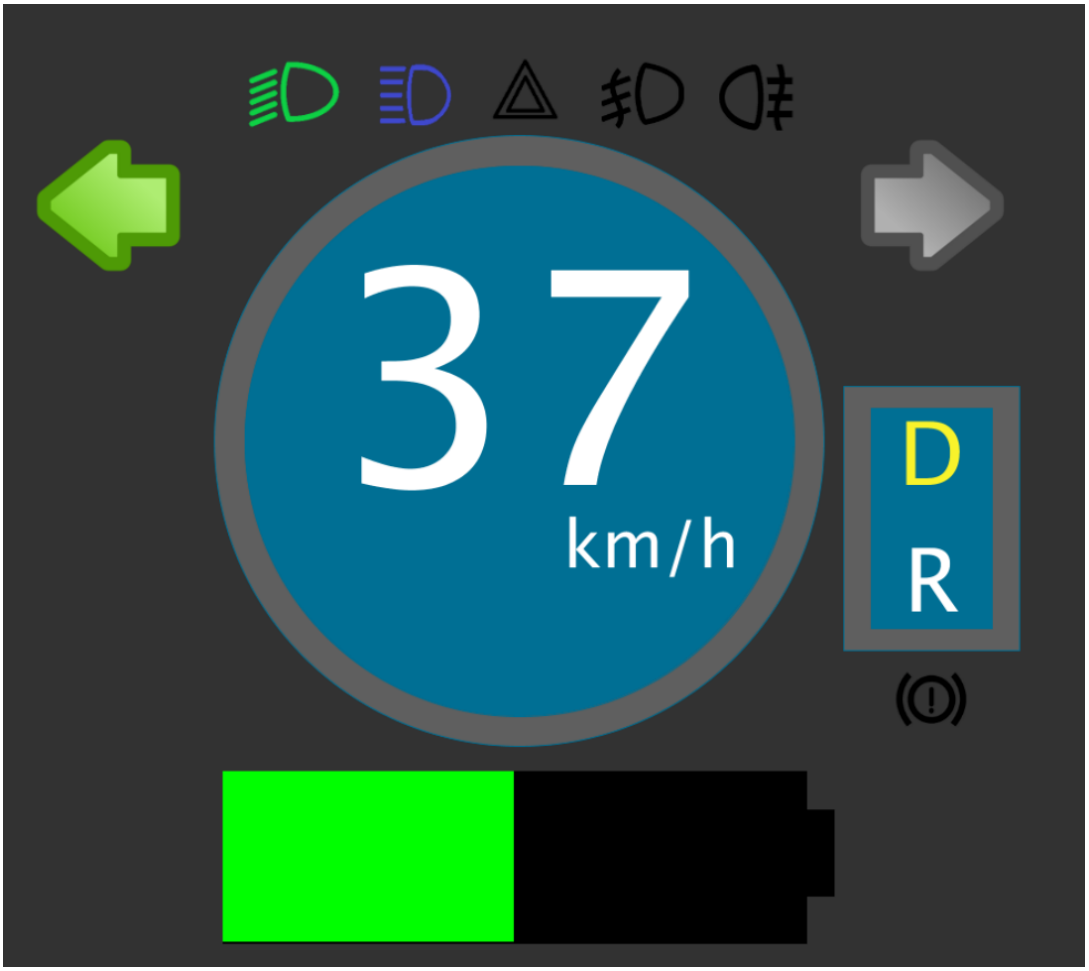


Figura 8.5 Prueba marcador 1

Prueba 2

- Triángulo de emergencia (intermitentes activos)
- Batería al 25%
- Velocidad actual de 15 km/h
- Marcha atrás

Figura 8.6 Prueba marcador 2

Prueba 3

- Intermitente derecho encendido
- Luces cortas, largas, antiniebla trasero y delantero
- Freno de mano puesto
- Batería al 12 % y cargándose
- Velocidad actual de 0 km/h
- Marcha directa

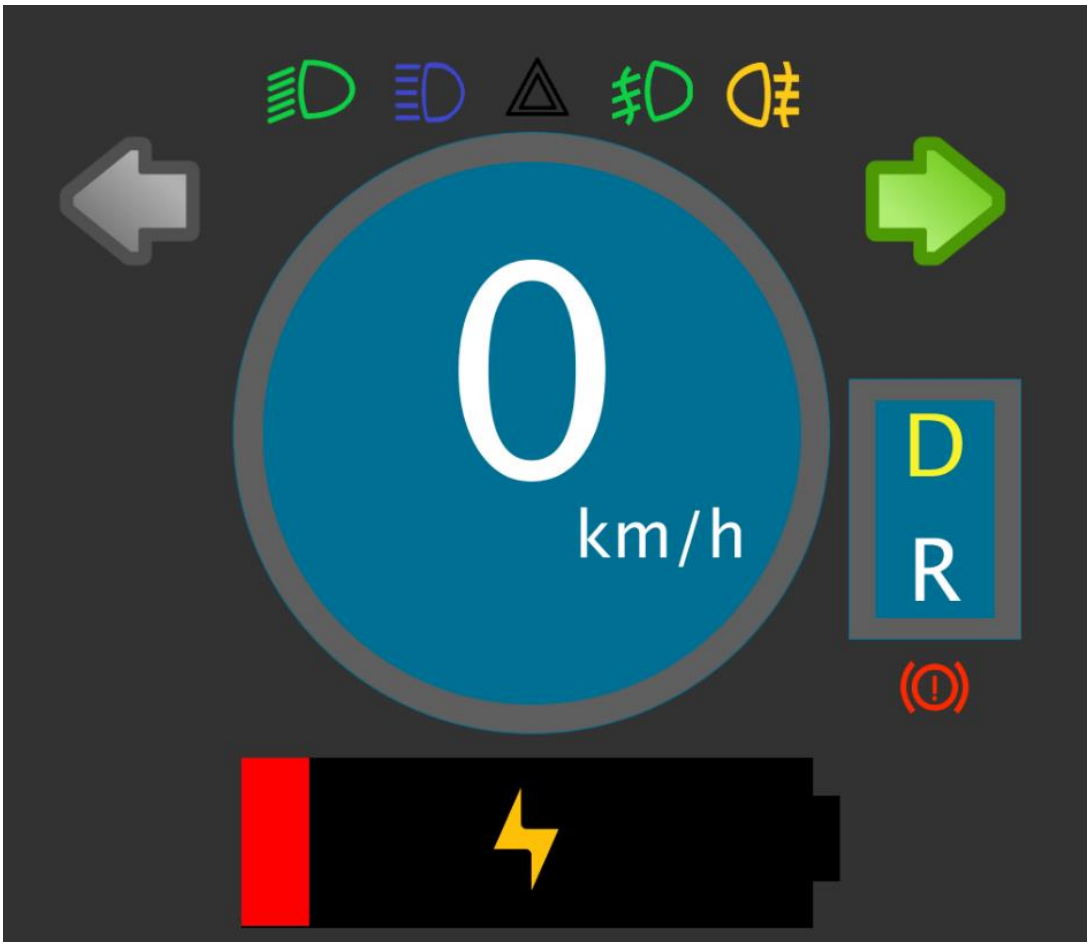


Figura 8.7 Prueba marcador 3

Actualmente, los valores mostrados por el monitor visual son ajustados manualmente mediante el código realizado, por lo que, para futuras investigaciones, se podría realizar un sistema de sensores y cableado para controlar los diferentes parámetros que lo componen.

9 POSIBLES FUTURAS FUENTES DE INVESTIGACIÓN

A continuación, se exponen una serie de posibles investigaciones para llevar a cabo futuros trabajos relacionados con el vehículo experimental para el objetivo final de la obtención de un vehículo eléctrico totalmente autónomo:

- Implementación un sistema de frenado ligado al presente trabajo mediante el uso de la central hidráulica.

- Instalación de un sistema de cableado para transmisión de los parámetros que conforman el marcador interactivo de Processing.

- Implementación de un sistema de giro autónomo del vehículo mediante del uso del motor conectado a un encoder ubicado en la parte trasera del volante.

- Implementación de un sistema de cámaras para la detección de señales, personas, aparcamiento, etc.

En paralelo al presente trabajo se desarrolló otro que consistió en la implementación de un sistema de navegación GPS con filtro Kalman mediante el uso de un módulo de GPS compatible con Arduino. En la Figura 9.1 se puede observar la fusión de ambos trabajos mediante el uso de dos placas Arduino.

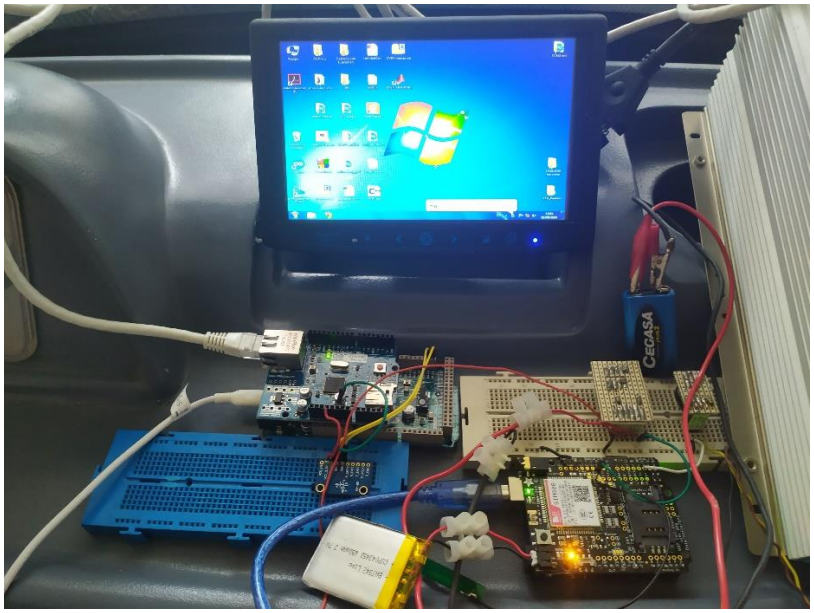


Figura 9.1 Montaje conjunto al sistema de navegación GPS

ANEXO 1. CÓDIGOS PARCIALES DE ARDUINO

PRUEBA DEL PWM

Para la utilización de este código es necesario realizar el montaje de la Figura 7.1.

Para comprobar su funcionamiento será necesario colocar los interruptores instalados en la posición autónomo y cambiar en el sketch el valor de la variable ‘acelerador’ por un valor comprendido entre 0 y 255.

```
/*********************************************************************/
/* PRUEBA PWM                                                        */
/*                                                                   */
/* IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE VELOCIDAD              */
/* PARA UN PROTOTIPO DE VEHÍCULO ELÉCTRICO AUTÓNOMO                  */
/* Grado en Ingenería Electrónica Industrial y Automática            */
/* Realizado por Diego Sánchez López // Septiembre 2019              */
/*********************************************************************/

const byte  acelerador  = 9   ;

void setup() {

  // Velocidad comunicación serie
  Serial.begin(115200);

  // Salida aceelerador por el pin 9
  pinMode(acelerador, OUTPUT);

  // Inicializar salida acelerador a 0
  analogWrite(acelerador,0);

}

void loop() {

  // Enviamos un valor entre [0,255] (*Cambiar valor*)
  analogWrite(acelerador,0);

}
```

PRUEBA DEL SENSOR HALL

Para la utilización de este código es necesario realizar el montaje de la Figura 7.2.

Para comprobar su funcionamiento será necesario colocar los interruptores instalados en la posición autónomo y cambiar en el sketch el valor de la variable ‘acelerador’ por un valor comprendido entre 0 y 255.

El programa mostrará por el monitor serie de Arduino IDE el número de pulsos recibidos cada 0.1 segundos en función del valor que hayamos introducido en la anterior variable, a mayor valor, mayor número de pulsos.

```
/*********************************************************************/
/* PRUEBA SENSOR HALL                                                */
/*                                                                   */
/* IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE VELOCIDAD              */
/* PARA UN PROTOTIPO DE VEHÍCULO ELÉCTRICO AUTÓNOMO                  */
/* Grado en Ingenería Electrónica Industrial y Automática            */
/* Realizado por Diego Sánchez López // Septiembre 2019              */
/*********************************************************************/

////////////////////////////////////////////////

// Entradas y salidas
const byte pinInterrupcion = 2 ;
const byte acelerador      = 9 ;

// Variables temporales.
unsigned long t  = 0       ; // Contador tiempo
unsigned long Ts = 100000  ; //Cada 0.1 segundos interrupción

// Contador de pulsos interrupción
volatile unsigned long pulsos=0;

////////////////////////////////////////////////

void setup( ) {

  // Salida acelerador por el pin 9
  pinMode(acelerador,OUTPUT);

  // Prescaler a 1 del Timer1;
  TCCR1B=TCCR1B & 0b11111000 | 0x01;

  // Velocidad comunicación serie
  Serial.begin(115200);

  // Activar interrupción
  attachInterrupt(digitalPinToInterrupt(pinInterrupcion),sensor,CHANGE);

  // Inicializar salida acelerador a 0
  analogWrite(acelerador,0);

}

////////////////////////////////////////////////

void loop(  ) {
```

```
47.
48.   // Cada 0.1 segundos ejecutamos el bucle.
49.   if (micros()-t >= Ts){
50.       t=micros();
51.
52.       // Salida del acelerador
53.       analogWrite(acelerador,0); //*Cambiar por un valor entre 0 y 255*
54.
55.       // Mostramos los pulsos por el monitor serie en función del acelerador
56.       Serial.print(pulsos);
57.       Serial.print("//");
58.
59.       // Reseteamos el contador de pulsos
60.       pulsos = 0;
61.       }
62. }
63.
64. ////////////////////////////////////////////////
65. // **INTERRUPCIÓN** //
66.
67. void sensor(){
68.
69.   // Aumentamos la variable pulsos cuando se activa la interrupción
70.   pulsos  ++ ;
71. }
```

ANEXO 2. CÓDIGO ARDUINO FINAL DEL SISTEMA DE CONTROL DE VELOCIDAD

Para la utilización de este código es necesario realizar el montaje de la Figura 7.3.

Para comprobar su funcionamiento será necesario colocar los interruptores instalados en la posición autónomo y cambiar en el sketch el valor de la variable 'velFin' por un valor comprendido entre 0 y 40, siendo esta última la velocidad que se impuso en el código como máxima permitida.

```
1.  /*****************************************************************/
2.  /* SISTEMA DE CONTROL DE VELOCIDAD                               */
3.  /*                                                               */
4.  /* IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE VELOCIDAD          */
5.  /* PARA UN PROTOTIPO DE VEHÍCULO ELÉCTRICO AUTÓNOMO              */
6.  /* Grado en Ingenería Electrónica Industrial y Automática        */
7.  /* Realizado por Diego Sánchez López // Septiembre 2019          */
8.  /*****************************************************************/
9.
10. ////////////////////////////////////////////////
11.
12. // Entradas y salidas
13. const byte pinInterrupcion = 2 ;
14. const byte acelerador      = 9 ;
15.
16. // Variables temporales.
17. unsigned long t  = 0       ; // Contador tiempo
18. unsigned long Ts = 100000  ; //Cada 0.1 segundos interrupción
19.
20. // Variables constantes
21.
22. // Pulsos por vuelta
23. const byte  pulsosvuelta = 80 ;
24.
25. // Número pi
26. const float pi = 3.14159 ;
27.
28. // Velocidad máxima limitada a 40
29. const float velmax = 40 ;
30.
31. // Loongitud rueda
32. const float longr = 1.76 ;
33.
34. // Ecuación cte velocidad actual
35. const float k = longr/pulsosvuelta/(1e-6*Ts)*3.6 ;
36.
37. // Factor correción
38. const float f = 0.8 ;
39.
40. // Contador de pulsos interrupción
41. volatile unsigned long pulsos=0;
42.
43. // Variables de velocidad.
44. float velAct  = 0.0 ; // Velocidad actual
45. float velFin  = 0.0 ; // Velocidad final deseada *Cambiar valor*
46.
47. // Variables del controlador PID.
```

```
48. const float kp    = 0.4,    ki        = 0.1,    kd    = 0.0 ;
49. float       e     = 0.0,    ie        = 0.0,    de    = 0.0 ;
50. float       eAnt  = 0.0,    velActAnt = 0.0,    sPID  = 0.0 ;
51.
52. ////////////////////////////////////////////////
53.
54. void setup( ) {
55.
56. // Salida acelerador por el pin 9
57. pinMode(acelerador,OUTPUT);
58.
59. // Prescaler a 1 del Timer1;
60. TCCR1B=TCCR1B & 0b11111000 | 0x01;
61.
62. // Velocidad comunicación serie
63. Serial.begin(115200);
64.
65. // Activar interrupción.
66. attachInterrupt(digitalPinToInterrupt(pinInterrupcion),sensor,CHANGE);
67.
68. // Inicializar salida acelerador a 0.
69. analogWrite(acelerador,0);
70.
71. }
72.
73. ////////////////////////////////////////////////
74.
75. void loop(  ) {
76.
77.   // Cada 0.1 segundos ejecutamos el bucle.
78.   if (micros()-t >= Ts){
79.       t=micros();
80.
81.       // Si la velocidad final es igual a la actual, mantenemos la salida cte
82.       if (velAct >= velFin){
83.           analogWrite(acelerador,(byte)(sPID*255));
84.       }
85.
86.       else{
87.
88.       // Ecuación para calcular la velocidad actual
89.       velAct=f*velAct+(1-f)*pulsos*k;
90.
91.       Serial.print(velAct); // Mostrar velocidad actual por monitor serie
92.       Serial.print("//");
93.
94.      // Activar acelerador si sPID >=0
95.      if( sPID >= 0){
96.          analogWrite(acelerador,(byte)(sPID*255));
97.       }
98.
99.       else  {
100.          analogWrite(acelerador,0);
101.      }
102.
103.     //*Descomentar si se desea ver los pulsos por monitor serie*
104.     //Serial.print(pulsos);
105.     //Serial.print("//");
106.
107.     // Reseteamos el contador de pulsos
108.     pulsos = 0;
109.     }
110.  }
111.
112. }
113.
```

```
114. ////////////////////////////////////////////////
115. // **INTERRUPCIÓN** //
116.
117. void sensor(){
118.    // Aumentamos la variable pulsos cuando se activa la interrupción
119.    pulsos  ++ ;
120. }
121.
122. ////////////////////////////////////////////////
123. // **PID** //
124.
125. void  controladorPID(){
126.
127.   // Salvar error y velocidad anterior
128.   eAnt = e;
129.   velActAnt = velAct;
130.
131.   // Calculamos el error
132.   e = (velFin - velAct)/velmax;
133.
134.   // Calculamos la integral del error
135.   ie += (e + eAnt)* 0.5 * Ts * 1e-6;
136.
137.   if ((ie*ki)> 1 ){
138.      ie=1/ki;
139.   }
140.   else if ((ie*ki) < -1 ){
141.      ie=(-1)/ki;
142.   }
143.
144.   // Derivada del error (*No se utiliza Kd*)
145.   //de = (e-eAnt)/(Ts * 1e-6);
146.
147.   // Suma de los elementos que forman el PID.
148.   sPID  = kp * e + ki  * ie  + kd  * de ;
149.
150.   // Limitamos salida del PID entre 1 y -1.
151.   sPID  = max( -1.0 , min( 1.0 , sPID)) ;
152.
153. }
```

ANEXO 3. CÓDIGO PROCESSING MARCADOR DE VELOCIDAD INTERACTIVO

El siguiente código muestra mediante el monitor un marcador cuentakilómetros interactivo mediante el software Processing.

Para comprobar su funcionamiento únicamente es necesario cambiar los valores de las diferentes variables de los elementos que componen el marcador.

```
/*************************************************************************/
/* MARCADOR CUENTAKILÓMETROS INTERACTIVO CON PROCESSING                  */
/*                                                                       */
/* IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE VELOCIDAD                  */
/* PARA UN PROTOTIPO DE VEHÍCULO ELÉCTRICO AUTÓNOMO                      */
/* Realizado por Diego Sánchez López // 2019                             */
/*************************************************************************/


// DEFINICIÓN DE VARIABLES DE LOS ELEMENTOS DEL MARCADOR //

// Velocidad del vehículo de 0 a 45 km/h
int velocidad = 0;

// Intermitente derecho encendido --> 1
int intder = 0;
// Intermitente izquiero encendido --> 1
int intizq = 0;

// Indicar porcentaje de batería
int porcentajeBateria = 0;
// Variable para cálculos de la batería
int cargaBateria = 0;

// Cargando el coche --> 1
int cargando = 0;

// Marcha directa --> 1
// Marcha atrás   --> 0
int marcha = 1;

// Luces cortas encendidas -->1
int lucesc = 0;

// Luces largas encendidas -->1
int lucesl = 0;

// Luces antiniebla delanteras/traseras encendidas -->1

int lucesantd = 0;
int lucesantt = 0;

// Triangulo intermitentes encendido -->1

int triangulo = 0;

// Freno de mano puesto -->1
int frenomano = 0;

```

```
50. ////////////////////////////////////////////////////////////////
51.
52. // IMAGENES //
53.
54. PImage  IntermitenteIzqON;
55. PImage  IntermitenteIzqOFF;
56.
57. PImage  IntermitenteDerON;
58. PImage  IntermitenteDerOFF;
59.
60. PImage trueno;
61.
62. PImage bateria;
63.
64. PImage warningON;
65. PImage warningOFF;
66.
67. PImage lucescortasON;
68. PImage lucescortasOFF;
69.
70. PImage luceslargasON;
71. PImage luceslargasOFF;
72.
73. PImage lucesantiniebladON;
74. PImage lucesantiniebladOFF;
75.
76. PImage lucesantinieblatON;
77. PImage lucesantinieblatOFF;
78.
79. PImage trianguloON;
80. PImage trianguloOFF;
81.
82. ////////////////////////////////////////////////////////////////
83. // SETUP //
84.
85. void setup() {
86.
87.   // Resolución
88.   size(1920,1080);
89.   // Imagenes por segundo (FPS)
90.   frameRate(60);
91.   noStroke();
92.
93.   // CARGAR IMAGENES DEL DIRECTORIO
94.   IntermitenteIzqON=loadImage("IntermitenteIzqON.png");
95.   IntermitenteIzqOFF=loadImage("IntermitenteIzqOFF.png");
96.
97.   IntermitenteDerON=loadImage("IntermitenteDerON.png");
98.   IntermitenteDerOFF=loadImage("IntermitenteDerOFF.png");
99.
100.   bateria=loadImage("bateria.png");
101.
102.   trueno=loadImage("trueno.png");
103.
104.   warningON=loadImage("warningON.png");
105.   warningOFF=loadImage("warningOFF.png");
106.
107.   lucescortasON=loadImage("lucescortasON.png");
108.   lucescortasOFF=loadImage("lucescortasOFF.png");
109.
110.   luceslargasON=loadImage("luceslargasON.png");
111.   luceslargasOFF=loadImage("luceslargasOFF.png");
112.
113.   lucesantiniebladON=loadImage("lucesantiniebladON.png");
114.   lucesantiniebladOFF=loadImage("lucesantiniebladOFF.png");
115.
```

```
116.   lucesantinieblatON=loadImage("lucesantinieblatON.png");
117.   lucesantinieblatOFF=loadImage("lucesantinieblatOFF.png");
118.
119.   trianguloON=loadImage("trianguloON.png");
120.   trianguloOFF=loadImage("trianguloOFF.png");
121.
122.   //Regla de 3 porcentaje/pixeles para calcular la carga de la batería
123.   cargaBateria=(porcentajeBateria*530)/100;
124.
125. }
126.
127. ////////////////////////////////////////////////////////////////
128. // DRAW //
129. void draw (){
130.
131.   // FONDO
132.   background(190);
133.   textAlign(CENTER);
134.
135.   // TABLERO
136.   fill(50);
137.   rect(100,100,1700,880);
138.
139.   // ESFERA MARCADOR
140.   fill(95);
141.   ellipse(960,500,550,550);
142.   stroke(#006F94);
143.   ellipse(960,500,555,555);
144.   fill(#006F94);
145.   ellipse(960,500,500,500);
146.
147.
148.   // PANEL MARCHAS
149.   fill(95);
150.   rect(1255,450,160,240);
151.   fill(#006F94);
152.   rect(1280,470,110,200);
153.
154.   // MARCHA
155.   if (marcha == 1) {
156.     textSize(80);
157.     //Amarillo
158.     fill(248,243,43);
159.     text("D",width/2+375,height/2);
160.
161.     textSize(80);
162.     //Rojo
163.     fill(255);
164.     text("R",width/2+375,height/2+115);
165.   }
166.
167.    else if (marcha == 0) {
168.
169.
170.     textSize(80);
171.     fill(255);
172.     text("D",width/2+375,height/2);
173.
174.     textSize(80);
175.     //Rojo
176.     fill(255,0,0);
177.     text("R",width/2+375,height/2+115);
178.   }
179.
180.   // FRENO DE MANO
181.
```

```
182.   if (frenomano == 1){
183.   pushMatrix();
184.     warningON.resize(90,90);
185.     image(warningON,1290,690);
186.   popMatrix();
187.   }
188.   else {
189.   pushMatrix();
190.     warningOFF.resize(90,90);
191.     image(warningOFF,1290,690);
192.   popMatrix();
193.   }
194.
195.
196.   // LUCES CORTAS
197.
198.   if (lucesc == 1){
199.   pushMatrix();
200.     lucescortasON.resize(90,90);
201.     image(lucescortasON,710,140);
202.   popMatrix();
203.   }
204.   else {
205.   pushMatrix();
206.     lucescortasOFF.resize(90,90);
207.     image(lucescortasOFF,710,140);
208.   popMatrix();
209.   }
210.
211.   // LUCES LARGAS
212.
213.     if (lucesl == 1){
214.   pushMatrix();
215.     luceslargasON.resize(90,90);
216.     image(luceslargasON,820,140);
217.   popMatrix();
218.   }
219.   else {
220.   pushMatrix();
221.     luceslargasOFF.resize(90,90);
222.     image(luceslargasOFF,820,140);
223.   popMatrix();
224.   }
225.
226.
227.   // LUCES ANTINIEBLA DELANTERAS
228.
229.     if (lucesantd == 1){
230.   pushMatrix();
231.     lucesantiniebladON.resize(75,75);
232.     image(lucesantiniebladON,1035,147);
233.   popMatrix();
234.   }
235.   else {
236.   pushMatrix();
237.     lucesantiniebladOFF.resize(75,75);
238.     image(lucesantiniebladOFF,1035,147);
239.   popMatrix();
240.   }
241.
242.   // LUCES ANTINIEBLA TRASERAS
243.
244.     if (lucesantt == 1){
245.   pushMatrix();
246.     lucesantinieblatON.resize(120,90);
247.     image(lucesantinieblatON,1118,143);
```

```
248.   popMatrix();
249.   }
250.   else {
251.   pushMatrix();
252.     lucesantinieblatOFF.resize(120,90);
253.     image(lucesantinieblatOFF,1118,143);
254.   popMatrix();
255.   }
256.
257.
258.   // INTERMITENTES
259.
260.   // Izquierdo
261.   if (intizq == 1){
262.   pushMatrix();
263.     image(IntermitenteIzqON,500,200);
264.   popMatrix();
265.   }
266.   else {
267.   pushMatrix();
268.     image(IntermitenteIzqOFF,500,200);
269.   popMatrix();
270.   }
271.
272.
273.   // Derecho
274.   if (intder == 1){
275.   pushMatrix();
276.     image(IntermitenteDerON,1250,200);
277.   popMatrix();
278.   }
279.   else {
280.   pushMatrix();
281.     image(IntermitenteDerOFF,1250,200);
282.   popMatrix();
283.   }
284.
285.   // TRIANGULO INTERMITENTES
286.
287.   if (triangulo == 1){
288.   pushMatrix();
289.     trianguloON.resize(70,60);
290.     image(trianguloON,930,150);
291.     image(IntermitenteIzqON,500,200);
292.     image(IntermitenteDerON,1250,200);
293.   popMatrix();
294.   }
295.   else {
296.   pushMatrix();
297.     trianguloOFF.resize(70,60);
298.     image(trianguloOFF,930,150);
299.   popMatrix();
300.   }
301.
302. ////////////////////////////////////////////////////////
303.
304.   // BATERIA
305.   pushMatrix();
306.     image(bateria,690,800);
307.   popMatrix();
308.
309.   // CARGA DE LA BATERIA
310.
311.   //Batería superior al 30%
312.   if (porcentajeBateria>=30) {
313.   pushMatrix();
```

```
    noStroke();
    //Color verde
    fill(0,255,0);
    rect(690,800,cargaBateria,155);
  popMatrix();
  }


  //Batería entre el 20% y el 30%
  else if (porcentajeBateria>=20){
  pushMatrix();
    noStroke();
    //Color naranja
    fill(255,128,0);
    rect(690,800,cargaBateria,155);
  popMatrix();

  }


  //Batería baja, inferior al 20%
  else {
  pushMatrix();
    noStroke();
    //Color rojo
    fill(255,0,0);
    rect(690,800,cargaBateria,155);
  popMatrix();

  }

  //BATERIAS CARGANDO

  if (cargando==1) {

  pushMatrix();

    // Escalar la imagen
    trueno.resize(150,150);
    // Mostrar trueno de carga sobre la batería
    image(trueno,880,800);

  popMatrix();

  }

////////////////////////////////////////////////////////

  //VELOCÍMETRO DIGITAL

  // Velocidad actual
    textSize(270);
    fill(255);
    text(velocidad,width/2,height/2);

  // km/h
    textAlign(100);
    textSize(60);
    fill(255);
    text("km/h",width/2+40,height/2+75);

}
```

ANEXO 4. HOJAS DE CARACTERÍSTICAS (DATASHEETS) Y MANUALES

- Hoja de características de la controladora Curtis 1268. Disponible en: <https://cdn.curtisinstruments.com/products/datasheets/1268_datasheet_en.pdf> [Consulta en: 5 de septiembre de 2019]

- Hoja de características del chip ATMmega328P. Disponible en: <http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf> [Consulta en: 5 de septiembre de 2019]

- Hoja de características del Arduino Ethernet Shield. Disponible en: <http://wizwiki.net/wiki/lib/exe/fetch.php?media=products:w5500:w5500_ds_v106e_141230.pdf> [Consulta en: 5 de septiembre de 2019]

- Hoja de características del amplificador operacional LM358. Disponible en: <http://www.ti.com/lit/ds/symlink/lm358-n.pdf> [Consulta en: 5 de septiembre de 2019]

- Hoja de características del osciloscopio Fluke 199C. Disponible en: <https://docs-emea.rs-online.com/webdocs/00ca/0900766b800ca78b.pdf> [Consulta en: 5 de septiembre de 2019]

- Manual del software Arduino IDE. Disponible en: <https://arduinobot.pbworks.com/f/Manual+Programacion+Arduino.pdf> [Consulta en: 5 de septiembre de 2019]

- Manual de la controladora Curtis 1268. Disponible en: <https://cdn.curtisinstruments.com/products/manuals/1268_manual_en.pdf> [Consulta en: 5 de septiembre de 2019]

PRESUPUESTO

A continuación, se expone una tabla que recoge los precios de los componentes necesarios para la implementación del sistema en el vehículo.

Tabla 5. Presupuesto

PRODUCTO	Cantidad	Precio unitario (€/ud.)	Subtotal (IVA incluido)
Arduino Uno Rev3	1	20	20
Arduino Ethernet Shield	1	17.50	17.50
Interruptor de 3 polos	3	3.20	9.6
Pila Cegasa de 9V	1	1.54	1.54
Amplificador operacional LM358	1	0.36	0.36
Potenciómetro 10kΩ	2	0.41	0.82
Condensador 1uF	1	0.10	0.10
Condensador 1nF	1	0.10	0.10
		TOTAL	50,02 €

BIBLIOGRAFÍA

[1] Motor pasión: La Jamais Contente. Disponible en:<https://www.motorpasion.com/coches-hibridos-alternativos/el-jamais-contente-el-electrico-que-supero-los-100-kmh>
[Consulta en: 5 de septiembre de 2019]

[2] E-ducativa Catedu: Motor de cuatro tiempos ciclo de Otto. Disponible en: <http://e-ducativa.catedu.es/44700165/aula/archivos/repositorio/4750/4932/html/2_motor_de_cuatro_tiempos_ciclo_de_otto.html>
[Consulta en: 5 de septiembre de 2019]

[3] E-ducativa Catedu: Motor de cuatro tiempos ciclo diésel. Disponible en: <http://educativa.catedu.es/44700165/aula/archivos/repositorio/4750/4932/html/3_motor_de_cuatro_tiempos_ciclo_diesel.html>
[Consulta en: 5 de septiembre de 2019]

[4] Toyota: ¿Qué es un híbrido? Disponible en: <https://www.toyota.es/hybrid-innovation/preguntas-hibridas/que-es-un-hibrido>
[Consulta en: 5 de septiembre de 2019]

[5] Diario motor: F3DM un sedán híbrido plug in a la venta en china. Disponible en: <https://www.diariomotor.com/2008/11/24/byd-f3dm-un-sedan-hibrido-plug-in-a-la-venta-en-china/>
[Consulta en: 5 de septiembre de 2019]

[6] Auto GLP Madrid: Historia del GLP. Disponible en: <http://autoglpmadrid.es/que-es-el-glp/historia-del-glp/>
[Consulta en: 5 de septiembre de 2019]

[7] Race: Cómo son las baterías en los coches eléctricos. Disponible en: <https://www.race.es/como-son-baterias-coches-electricos>
[Consulta en: 5 de septiembre de 2019]

[8] Peugeot: ¿Qué es un coche eléctrico? Disponible en: <https://www.peugeot.es/que-es-un-coche-electrico.html>
[Consulta en: 5 de septiembre de 2019]

[9] La Vanguardia: Google prueba su coche pilotada con un invidente. Disponible en: <https://www.lavanguardia.com/motor/20120329/54278548114/google-prueba-invidente-coche-piloto-automatico.html>
[Consulta en: 5 de septiembre de 2019]

[10] Iberdrola: El coche autónomo. ¿Podría un 'hacker' conducir mi coche autónomo? Disponible en: <https://www.iberdrola.com/innovacion/coche-autonomo>
[Consulta en: 5 de septiembre de 2019]

[11] DGT: Distintivo ambiental. Disponible en: <http://www.dgt.es/es/seguridad-vial/distintivo-ambiental/>
[Consulta en: 5 de septiembre de 2019]

[12] Arduino Store: Uno Rev3. Disponible en: <https://store.arduino.cc/arduino-uno-rev3>
[Consulta en: 5 de septiembre de 2019]

[13] Arduino Store: Due. Disponible en: <https://store.arduino.cc/due>
[Consulta en: 5 de septiembre de 2019]

[14] Arduino Store: Mega 2560. Disponible en: < https://store.arduino.cc/mega-2560-r3>
[Consulta en: 5 de septiembre de 2019]

[15] Promotec: Guía a la programación de Arduino III–Librerías. Disponible en: <https://www.prometec.net/funciones-iii/>
[Consulta en: 5 de septiembre de 2019]

[16] Wikibooks: Protocolos TCP y UDP en el nivel de transporte. Disponible en: <https://es.wikibooks.org/wiki/Redes_inform%C3%A1ticas/Protocolos_TCP_y_UDP_en_el_nivel_de_transporte>
[Consulta en: 5 de septiembre de 2019]

[17] Tecnología de control. Sistema de control de lazo abierto y lazo cerrado. Disponible en: <https://sites.google.com/site/tecnologiadecontrol2016/sistema-de-control-manual>
[Consulta en: 5 de septiembre de 2019]

[18] Electrónica Industrial: Sistema de control. Lazo abierto/cerrado. Disponible en: <http://eet602ei.blogspot.com/2012/05/sistemas-de-control-lazo-abiertocerrado.html>
[Consulta en: 5 de septiembre de 2019]

[19] Picuino: Control PID. Disponible en: <https://www.picuino.com/es/arduprog/control-pid.html>
[Consulta en: 5 de septiembre de 2019]

[20] Motor pasión: Que es un LIDAR y cómo funciona. Disponible en: <https://www.motorpasion.com/tecnologia/que-es-un-lidar-y-como-funciona-el-sistema-de-medicion-y-deteccion-de-objetos-mediante-laser>
[Consulta en: 5 de septiembre de 2019]

[21] Luis Llamas: Salidas analógicas PWM en Arduino. Disponible en: <https://www.luisllamas.es/salidas-analogicas-pwm-en-arduino/>
[Consulta en: 5 de septiembre de 2019]

[22] Smar: Hall Sensor. La tecnología de los Posicionadores Inteligentes de última generación Disponible en: <http://www.smar.com/espanol/articulos-tecnicos/hall-sensor-la-tecnologia-de-los-posicionadores-inteligentes-de-ultima-generacion>
[Consulta en: 5 de septiembre de 2019]

[23] Promotec: Interrupciones. Disponible en: <https://www.prometec.net/interrupciones/>
[Consulta en: 5 de septiembre de 2019]

[24] Wordpress: Aprendiendo Arduino. Interrupciones Disponible en: <https://aprendiendoarduino.wordpress.com/2016/11/13/interrupciones/>
[Consulta en: 5 de septiembre de 2019]

[25] Programar fácil: Processing, el lenguaje para gráficos. Disponible en: <https://programarfacil.com/podcast/80-processing-lenguaje-para-graficos/>
[Consulta en: 5 de septiembre de 2019]