UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Diseño de un banco de ensayos de máquinas de imanes permanentes de técnica senoidal (servoaccionamiento PMSM Brushless AC) de 200W con interfaz HTML basada en ESP32.

Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

AUTOR/A: Ferrer Vayá, Ismael

Tutor/a: Martínez Román, Javier Andrés

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

DOCUMENTOS DEL PROYECTO

MEMORIA

PRESUPUESTO

ANEXOS DE LA MEMORIA

Resumen del proyecto:

El objetivo principal del trabajo es el diseño de una interfaz de operación para un servomotor de 200W de potencia mediante una red local de conexión Wifi.

Para ello es necesario la programación de un servidor local, el cual gestionará el intercambio de datos entre el usuario cliente del servidor y el servomotor, y cuyo hosting será el microcontrolador ESP32 de la marca Espressif. Paralelamente se debe diseñar y construir la electrónica encargada de alimentar al microcontrolador y todos los componentes auxiliares que permiten la comunicación cableada entre éste y el servomotor. También se debe realizar específicamente la interfaz entre el usuario humano y el sistema, con este objetivo se ha programado una página web, que aporta la interfaz a todo aquel dispositivo que se conecte a la dirección indicada, y que hace la función del tablero de mandos del servomotor.

Con esto queda patente la implicación de diversas ramas de la técnica en este proyecto, como la electrónica, la informática o la automática. Disciplinas que están en una evolución constante y veloz y que han permitido en gran medida la revolución tecnológica de los últimos 75 años.

El resultado final del proyecto consiste en la creación de un prototipo de producto que permite la comunicación remota con un servomotor, empleando como intermediario un microcontrolador, lo cual lo hace extremadamente barato y fácilmente modificable.

Palabras clave: Interfaz de usuario WEB, interfaz electrónica, servoaccionamiento, ESP32, banco de ensayos de motores.

Project summary:

The main objective of this work is the design of an operating interface for a 200W servomotor through a local Wi-Fi connection network.

For this purpose, it is necessary to program a local server that will manage the data exchange between the server's client user and the servomotor. The hosting of this server will be the ESP32 microcontroller. Simultaneously, it is required to design and build the electronics responsible for powering the microcontroller and all the auxiliary components that enable the wired communication between it and the servo motor. Furthermore, the interface between the human user and the system must be specifically developed. To achieve this, a web page has been programmed, which provides the interface to any computer that connects to the specified address and functions as the control panel for the servo motor.

This clearly demonstrates the involvement of various technical fields in this project, such as electronics, computer science, and automation. These disciplines are constantly and rapidly evolving and have greatly contributed to the technological revolution of the last 75 years.

The result of the project consists of creating a prototype that allows remote communication with a servomotor, using a microcontroller as an intermediary. This approach makes it extremely affordable and easily modifiable.

Keywords: Web user interface, electronic interface, servo drive, ESP32, motor testing bench.

Resum del projecte:

L'objectiu principal del treball és el disseny d'una interfície d'operació per a un servomotor de 200W de potència mitjançant una xarxa local de connexió Wifi.

Per a això és necessària la programació d'un servidor local, el qual gestionarà l'intercanvi de dades entre l'usuari client del servidor i el servomotor, sent el hosting del servidor el microcontrolador ESP32 de la marca Espressif. Paral·lelament s'ha de dissenyar i construir l'electrònica encarregada d'alimentar el microcontrolador i tots els components auxiliars que permeten la comunicació cablejada entre aquest i el servomotor. També s'ha de realitzar específicament la interfície entre l'usuari humà i el sistema, amb aquest objectiu s'ha programat una pàgina web, que aporta la interfície a tot aquell dispositiu que es connecta a l'adreça indicada, i que fa la funció del tauler de comandament del servomotor.

Amb això queda patent la implicació de diverses branques de la tècnica en aquest projecte, com l'electrònica, la informàtica o l'automàtica. Disciplines que estan en una evolució constant i ràpida i que han permés en gran mesura la revolució tecnològica dels últims 75 anys.

El resultat final del projecte consisteix en la creació d'un prototip que permet la comunicació remota amb un servomotor, utilitzant com a intermediari un microcontrolador, la qual cosa el fa extremadament barat i fàcilment modificable.

Paraules clau: interfície de usuari web, interfície electrònica, servoaccionament, ESP32, taulell d’assajos de motors.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

DISEÑO DE UN BANCO DE ENSAYOS DE MÁQUINAS DE IMANES PERMANENTES DE TÉCNICA SENOIDAL (SERVOACCIONAMIENTO PMSM BRUSHLESS AC) DE 200W CON INTERFAZ HTML BASADA EN ESP32

MEMORIA

Autor: Ismael Ferrer Vayá

Tutor: Javier Andrés Martínez Román

Universidad politécnica de Valencia

Curso académico: 2022 – 2023

Índice de la memoria

1. OBJETIVO DEL TRABAJO 1
2. INTRODUCCIÓN AL PROBLEMA 2
   2.1 Antecedentes *2*
   2.2 Motivación y justificación *2*
   2.3 Relación del proyecto con los objetivos de desarrollo sostenible *3*
   2.4 Estructura de la memoria *4*
3. NORMATIVA 6
4. DISEÑO GLOBAL DEL SISTEMA 7
   4.1 Componentes propios del servomotor *7*
      4.1.1 Servopack 7
      4.1.2 Servomotor 8
   4.2 Funcionamiento del sistema *9*
   4.3 Selección del microcontrolador *10*
      4.3.1 Microcontroladores disponibles 11
      4.3.2 Microcontrolador seleccionado 13
   4.4 Otros dispositivos empleados *13*
      4.4.1 Interfaz electrónica 13
      4.4.2 Router Wifi 14
5. DISEÑO Y ESPECIFICACIONES DEL SERVIDOR 15
   5.1 Introducción *15*
   5.2 Cliente y servidor *15*
   5.3 Servidor asíncrono *16*
   5.4 Protocolo de envío de página (HTTP) *16*
   5.5 Protocolo de intercambio de información (Websocket) *17*
   5.6 Paquetes de información (JSON) *18*
6. INTERFAZ HTML (USUARIO-SERVIDOR) 20
   6.1 Introducción *20*
   6.2 Presentación del sitio web *20*
      6.2.1 Entradas de control 21
      6.2.2 Salidas de control 22
      6.2.3 Indicadores de comunicación con el servidor 23
   6.3 Lenguajes empleados *24*
      6.3.1 HTML 24
      6.3.2 CSS 25

6.3.3 JavaScript 26
7. PROGRAMACIÓN DEL MICROCONTROLADOR 27
7.1 Introducción 27
7.2 Arduino 27
7.3 Características internas del microcontrolador 28
7.3.1 Patillaje empleado 28
7.3.2 Capacidad multinúcleo 28
7.3.3 Soporte para interrupciones 29
7.3.4 Generación de señal analógica mediante PWM 29
7.3.5 Puerto micro-USB 30
7.3.6 Antena integrada 30
7.4 Librerías empleadas 31
7.4.1 Wifi.h 31
7.4.2 Ticker.h 31
7.4.3 ESPAsyncWebServer.h 31
7.4.4 WebSocketsServer.h 32
7.4.5 ArduinoJson.h 32
8. INTERFAZ ELECTRÓNICA 33
8.1 Introducción 33
8.2 Diseño de filtros para las señales analógicas 33
8.2.1 Objetivos deseados 33
8.2.2 Fundamentos teóricos empleados 34
8.2.3 Selección del tipo de filtro base 35
8.2.4 Ajuste de la tensión de desbalance en el caso de la velocidad 37
8.2.5 Cálculos realizados para el filtrado y amplificación de la velocidad 38
8.2.6 Ajuste de la tensión de desbalance en el caso del límite de par 41
8.2.7 Cálculos realizados para el filtrado y amplificación del límite de par 42
8.2.8 Modelo empleado de amplificador operacional 45
8.3 Diseño de divisores de tensión para las salidas digitales 45
8.3.1 Objetivos deseados 45
8.3.2 Disposición de los divisores de tensión. 46
8.3.3 Cálculos de las resistencias a emplear 47
8.4 Diseño de transistores de potencia para las entradas digitales 48
8.4.1 Objetivos deseados 48
8.4.2 Circuito electrónico empleado 48
8.4.3 Modelo de transistor empleado 49
8.4.4 Cálculos para el diseño 50

8.5 Diseño de etapas de alimentación ... *51*
8.5.1 Alimentación de los amplificadores operacionales ... 51
8.5.2 Fuente de alimentación de 5V ... 53
8.5.3 Fuente de alimentación de 24V ... 53
8.6 Otros elementos implementados y distribución de pines empelados ... *54*
8.6.1 Zócalo del microcontrolador ... 54
8.6.2 Bus de datos paralelo ... 54
8.6.3 Recopilatorio de GPIOs, pines y señales eléctricas ... 55
8.7 Esquema de conjunto ... *56*
8.8 Construcción del prototipo ... *58*
8.8.1 Herramientas y materiales empleados en el proceso de construcción ... 58
8.8.2 Proceso constructivo ... 59
8.8.3 Prototipo finalizado ... 63
9. MANUAL DE USUARIO ... 64
9.1 Configuración de Arduino IDE, selección de red y dirección IP ... *64*
9.2 Configuración del servopack ... *65*
9.3 Montaje del conjunto ... *67*
9.4 Funcionamiento normal del sistema ... *68*
10. CONCLUSIÓN Y PROPUESTAS DE MEJORA ... 69
11. BIBLIOGRAFÍA Y WEBGRAFÍA ... 71

Índice de ilustraciones

Ilustración 1. Imagen del servopack. 8
Ilustración 2. Imagen del servomotor 9
Ilustración 3. Esquema de funcionamiento del conjunto 10
Ilustración 4. ESP32 11
Ilustración 5. ESP8266 12
Ilustración 6. Arduino MKR Wifi 1010 13
Ilustración 7. Interfaz electrónica 14
Ilustración 8. Ajax Vs Websocket 18
Ilustración 9. Sitio Web 21
Ilustración 10. Entradas de control 22
Ilustración 11. Salidas de control 23
Ilustración 12. Comunicación con el servidor 24
Ilustración 13. Modulación PWM 30
Ilustración 14. Esquema de entrada analógica al servopack 34
Ilustración 15. Esquema de filtro inversor 35
Ilustración 16. Esquema de filtro no inversor 36
Ilustración 17. Ajuste de offset para el control de la velocidad 37
Ilustración 18. Filtrado de la señal analógica de control de la velocidad 40
Ilustración 19. Ajuste de offset para el límite de par 41
Ilustración 20. Representación de `VAjuste` *en función de la regulación x* 42
Ilustración 21. Filtrado de la señal analógica de límite de par 44
Ilustración 22. Esquema TL082 45
Ilustración 23. Esquema de salida digital del servopack 46
Ilustración 24. Divisor de tensión 46
Ilustración 25. Entradas digitales del servopack 48
Ilustración 26. Esquema del transistor 49
Ilustración 27 . Conector paralelo 54
Ilustración 28. Distribución de los GPIO del ESP32 56
Ilustración 29. Esquema de la interfaz electrónica 57
Ilustración 30. Herramientas de soldadura 59
Ilustración 31. Zócalo del microcontrolador 59
Ilustración 32. Implementación de los divisores de tensión 60
Ilustración 33. Etapa de filtrado de las señales analógicas 60
Ilustración 34. Transistores de potencia 61
Ilustración 35. Fuentes de alimentación 61
Ilustración 36. Bus paralelo 62

Ilustración 37. Puerto bus paralelo *62*
Ilustración 38. Prototipo de interfaz electrónica finalizado *63*
Ilustración 39. Selector de parámetros *66*
Ilustración 40. Conjunto montado *67*

1. OBJETIVO DEL TRABAJO

El objetivo principal de este trabajo es el diseño, programación y construcción de un sistema de control de la velocidad y límite de par de un servomotor de 200W a través de la comunicación inalámbrica que ofrece una red local Wifi. Todo aquel dispositivo conectado a esta red tendrá acceso a la página web, que hace de tablero de mandos del servomotor, y con la que se puede interactuar en tiempo real enviando señales de control y recibiendo información del sistema.

El dispositivo encargado de gestionar el sistema es un microcontrolador. Siendo el hosting del servidor, que carga y gestiona todo lo relacionado con la página web, y también el que envía y recibe las señales eléctricas de control al servopack. El servopack es el controlador del servomotor y la etapa de potencia que permite su funcionamiento, por lo que realmente las señales se envían al servopack y no al servomotor directamente. El microcontrolador es la pieza clave del proyecto y también su punto más fuerte, pues es un dispositivo barato, pero con una potencia informática y unas prestaciones muy elevadas.

Sin embargo, el microcontrolador no dispone de circuitos de una potencia suficiente para comunicarse directamente con el servopack, por lo que se debe implementar una etapa de adaptación entre ambas partes que adecue los niveles de tensión y filtre las señales para permitir el intercambio de información. Así que además se ha diseñado y construido un prototipo de placa base en el que se inserta el microcontrolador y se conecta por cable el servopack.

Al ser un proyecto con gran variedad de disciplinas implicadas las partes en las que se divide son bastante diferenciables, siendo las siguientes:

- Programación de la página web (interfaz entre usuario y servidor) principalmente en los lenguajes HTML, CSS y JavaScript.
- Programación del servidor, codificado en lenguaje de Arduino.
- Diseño y construcción del prototipo de la placa base electrónica que permite la comunicación entre microcontrolador y servomotor.
- Montaje y cableado del conjunto placa base, servopack y servomotor.

Con todo ello también se pretende conocer las capacidades de la microelectrónica y la informática de redes y servidores como elementos de control de máquinas y accionamientos eléctricos. Teniendo estas ramas de la técnica un coste muy competitivo y unas altas capacidades de evolución y mejora.

2. INTRODUCCIÓN AL PROBLEMA

2.1 Antecedentes

Los microcontroladores llevan largo tiempo en la industria electrónica ya que surgieron pocos años después del primer microprocesador, resultaron ser elementos útiles para sistemas embebidos y para la creación de prototipos. Son el inicio de la filosofía "System on a chip" que persigue integrar todos los elementos de un computador en un único circuito integrado; memoria, interfaces de comunicación y unidad de procesamiento. Por otro lado, rara vez son utilizados para control de accionamientos o sistemas industriales que requieran cierta complejidad o potencia.

El control de los servomotores en la industria es realizado habitualmente por Ordenadores Lógicos Programables (PLC de sus siglas en inglés), ordenadores fácilmente programables y muy robustos que suelen llevar a cabo la gran mayoría de procesos de automatización y robótica industrial. Suelen ser dispositivos grandes y aparatosos, pues están diseñados para operar en entornos con muchas interferencias electromagnéticas y a controlar máquinas que poseen elevadas potencias. Además de ser elementos caros, es costoso ajustar sus características tanto de software como de hardware de forma profunda para conseguir que realicen algunas tareas para las que no han sido diseñados.

En el laboratorio de máquinas eléctricas hay implementado un sistema de control con PLC del servomotor de 200W que se va a utilizar en este proyecto. Todas las conexiones de control han sido cableadas y dispone de un tablero de mandos físico con ciertas entradas y salidas de control que le envían las señales por cable al PLC. En principio cumple con el mismo objetivo que el de este proyecto, no obstante, presenta ciertos inconvenientes. En primer lugar, se requiere un gran espacio para realizar todo el cableado y para ubicar el tablero de mandos, lo que contrasta totalmente con el poco espacio que requiere el servomotor. Por otro lado, no dispone de interacción remota y todas las acciones se deben realizar en contacto con el sistema, lo que en algunas ocasiones podría comprometer la seguridad del usuario si no se hace un buen uso del conjunto. Además, tampoco dispone de realimentación de la velocidad en el propio tablero de mandos. Por último, resultaría muy costoso realizar algún cambio o actualización al sistema, pues ello requeriría de sustituir el tablero de mandos físicamente y de realizar el cambio correspondiente en el cableado.

2.2 Motivación y justificación

La principal motivación de este proyecto es académica y del ámbito de la investigación, pues pretende dotar al servomotor de 200W mencionado anteriormente de una interfaz de control remota y fácil de utilizar desde cualquier tipo de dispositivo. Que sea capaz de unificar tanto el tablero de mandos como la realimentación de señales y el valor real de la velocidad, sin necesidad de cables ni de controles físicos y con una gran capacidad de adaptabilidad y personalización de su código. Tras la realización del proyecto, el estudio de este servomotor será más sencillo y económico, y también dotará a su control de una elevada accesibilidad.

Al tratarse de un proyecto con gran cantidad de disciplinas interactuando entre sí es relativamente sencillo relacionarlo con los estudios realizados a lo largo de la titulación. Está relacionado principalmente con la electricidad, la electrónica y la informática, y son bastantes los

conocimientos adquiridos los que han sido necesarios para llevar a cabo el objetivo del trabajo como se verá más adelante.

El resultado del proyecto es un prototipo de interfaz de operación que se ha construido físicamente y que será empleado en el laboratorio de máquinas eléctricas para el control de servomotores. Al tratarse de un prototipo tiene un interés meramente tecnológico y de estudio, no se ha realizado con el objetivo de ser un producto que pueda tener una aplicación directa en la industria o una salida comercial. Se trata de un nuevo dispositivo para el departamento de ingeniería eléctrica que en los últimos años ha realizado varios proyectos con microcontroladores para el control de diversos accionamientos.

Por último, cabe destacar que la industria se encamina hacia un modelo en el que el intercambio de datos, la informática y el control a distancia van a tener un papel fundamental, lo que se conoce como industria 4.0. Este trabajo sigue esta tendencia, en especial en lo que se conoce como "Internet de las cosas" (IoT de sus siglas en inglés), que pretende que los dispositivos mediante una conexión a la red puedan informar acerca de su operación y de actuar sobre ella en tiempo real de forma telemática.

2.3 Relación del proyecto con los objetivos de desarrollo sostenible

Los proyectos de ingeniería tienen la capacidad de provocar grandes cambios en su entorno de mayor o menor envergadura, por ello se deben realizar siempre con la motivación de brindar una mejoría y un avance tanto de la propia técnica como del mundo que los rodea. En este aspecto destacan los objetivos de desarrollo sostenible, promulgados por la ONU en 2015 que pretenden establecer una hoja de ruta para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todas las personas como parte de una nueva agenda de desarrollo sostenible. [1]

Este trabajo se vincula principalmente con 2 de estos objetivos: "Una educación de calidad" (Objetivo N.º 4) e "Industria, innovación e infraestructura" (Objetivo N.º 9) . Por ser un trabajo de investigación académica.

En relación con el cuarto objetivo, este proyecto busca facilitar el empleo de bancos de ensayo para el uso de servomotores, abaratando las instalaciones y permitiendo a una mayor cantidad de personas el estudio de este tipo de máquinas eléctricas. Además de adaptar su operación para que pueda ser operado por los estudiantes de forma segura. En especial se enfoca en las metas 4.4 y 4.a de este objetivo, que pretenden aumentar la accesibilidad al conocimiento mediante unas instalaciones seguras y sencillas.

En relación con el noveno objetivo, este trabajo consiste en la investigación acerca de un sistema novedoso de control de máquinas eléctricas que sea barato y eficiente, adoptando nuevas tecnologías y modernizando los sistemas. Basado fuertemente en la innovación y buscando la mejora de los sistemas productivos industriales, tanto de países industrializados como en vías de desarrollo. Principalmente conectando con las metas 9.4 y 9.5 que se enfocan en la mejora continuada de la industria y en la investigación para llevarla a cabo.

Se presenta a continuación una tabla que expone las relaciones de este proyecto con dichos objetivos:

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.				X
ODS 4. Educación de calidad.	X			
ODS 5. Igualdad de género.				X
ODS 6. Agua limpia y saneamiento.				X
ODS 7. Energía asequible y no contaminante.			X	
ODS 8. Trabajo decente y crecimiento económico.			X	
ODS 9. Industria, innovación e infraestructuras.	X			
ODS 10. Reducción de las desigualdades.			X	
ODS 11. Ciudades y comunidades sostenibles.				X
ODS 12. Producción y consumo responsables.			X	
ODS 13. Acción por el clima.				X
ODS 14. Vida submarina.				X
ODS 15. Vida de ecosistemas terrestres				X
ODS 16. Paz, justicia e instituciones sólidas.				X
ODS 17. Alianzas para lograr objetivos.				X

Tabla 1. Relación del proyecto con los objetivos de desarrollo sostenible

2.4 Estructura de la memoria

La memoria se divide en la explicación de la funcionalidad, diseño y construcción de cada uno de los sistemas que componen el conjunto del trabajo y que realizan una tarea determinada y distinta a la de los demás elementos. Aun así, se ha considerado necesario dedicar un capítulo a la explicación global de todo el sistema para establecer la correlación entre todos los elementos, así como para mostrar las decisiones de diseño que afectan a todo el conjunto. Se podrían enumerar estos elementos de la siguiente forma:

- El servidor: Programa encargado de la gestión de la comunicación inalámbrica entre usuario y microcontrolador. En su apartado se pretende explicar las diferentes tecnologías así cómo los estándares y protocolos que se han empleado en su diseño, pero no la explicación de su código.

- La interfaz HTML: Establece el canal de comunicación entre el usuario y el servidor, se trata de una página web que se cargará en el dispositivo del usuario y desde la cual podrá controlar el servomotor. Su apartado consta de una explicación de la interfaz creada, de su funcionamiento y de los diferentes lenguajes de programación empleados en su

codificación. La explicación del código HTML, así como el código completo aparecen en los anexos de la memoria.

- Programación del microcontrolador: Éste debe ser el encargado de soportar el servidor y de comunicarse mediante señales eléctricas con los demás dispositivos. En su apartado se detallan los diferentes elementos del microcontrolador que se utilizan, tanto para el servidor como para la gestión de entradas y salidas de señales eléctricas, una breve introducción al lenguaje de Arduino y las librerías que se emplean en su código. La explicación del código de programa, así como el código completo se encuentran en los anexos de la memoria.

- Interfaz electrónica: Prototipo de placa base electrónica de diseño y construcción propia cuya función es adaptar las señales eléctricas entre microcontrolador y servopack. Su apartado desarrolla tanto los cálculos eléctricos de su diseño, la elección de los componentes a emplear, así como la explicación de su construcción.

Adicionalmente habrá una serie de capítulos que contendrán el manual de usuario del dispositivo, propuestas de mejora y conclusiones y la bibliografía.

3. NORMATIVA

Debido a que el proyecto se centra en la construcción de un prototipo que no va a ser comercializado y cuyo principal objetivo es el uso académico, la normativa no tiene un papel fundamental en el trabajo. Sin embargo, sí que se ha aplicado en todos los aspectos en los que asegura un buen uso de los diversos elementos y la seguridad tanto de la instalación como de los usuarios.

Las normas y estándares aplicados en la realización del proyecto son los siguientes:

- Reglamento electrotécnico para baja tensión (REBT): Toda instalación eléctrica de baja tensión, así como los dispositivos conectados a ella, deben cumplir esta normativa para asegurar la seguridad y el correcto funcionamiento del sistema. Del mismo se obtienen los requisitos necesarios que debe cumplir la instalación en la que vaya a operar el prototipo.

- Norma IEC 61131-2:2017, acerca de controladores programables que entre otras indicaciones regula los valores de tensión de comunicación en valores seguros que son los empleados por el servopack.

- La norma UNE-EN 61010-031:2015/A11:2021 que indica los requisitos de seguridad para las mediciones realizadas en el laboratorio mientras se han llevado a cabo los ensayos del sistema.

Además de la normativa expuesta se seguirán las recomendaciones de los manuales de los dispositivos empleados en su montaje, cableado y alimentación, así como de las fichas técnicas de los elementos electrónicos que conforman el prototipo, asegurando la seguridad del usuario y del sistema y funcionando siempre dentro de los rangos de operación permitidos.

En el caso del diseño del servidor han sido empleados los protocolos y estándares habituales del desarrollo web con el objetivo de permitir su uso en la mayoría de los dispositivos, aumentando así la accesibilidad y facilitando la tarea a todo aquel interesado en modificar o implementar nuevas funcionalidades al sistema.

4. DISEÑO GLOBAL DEL SISTEMA

4.1 Componentes propios del servomotor

Los siguientes componentes del sistema son el punto de partida del proyecto, pues todos los demás elementos se han diseñado y elegido para comunicarse con ellos. Esto es debido a que es el servomotor y el servopack de que se dispone en el laboratorio y cuyo control es el objetivo del conjunto.

4.1.1 Servopack

Elemento encargado del control de movimiento y alimentación del servomotor y de indicar los datos sobre su operación, como velocidad o posición. Se empleará el modelo SGDH-02AE-OY producido por Yaskawa, diseñado para servomotores de 200W y alimentado a 230V monofásicos. Cuenta con distintos parámetros que se guardan en su memoria y que sirven para seleccionar desde el modo de funcionamiento hasta la relación de proporcionalidad entre una tensión de entrada y la velocidad demandada, dichos parámetros se seleccionan y modifican empleando los botones y la pantalla de cristal líquido de que dispone el servopack.

Para mover al servomotor utiliza un circuito de potencia formado por un inversor trifásico que se conecta a los bornes del servomotor y que está gobernado por la unidad central de cálculo y procesamiento para mantener la velocidad, posición o par correspondientes. El control lo realiza variando la frecuencia de estas ondas trifásicas generadas. Además, tiene varios conectores de entrada y salida para comunicación con PLC u otros dispositivos.

Hay que destacar por último el sistema interno del servopack encargado de recibir información del encoder del servomotor mediante transferencia serie y transformarlos en pulsos de tensión que serán salida del sistema y que se emplean para el control de la velocidad o la posición. En el punto siguiente se explica con más detalle qué es y para qué sirve un encoder.

En este proyecto no se emplean todas las características de este dispositivo, pero sí gran parte de ellas. Se pretende realizar un control de la velocidad y del par del servomotor, así que se empleará el modo de control de la velocidad con los ajustes para poder regular el límite de par, para la comunicación se utilizará el conector CN1 que se trata de un puerto paralelo de 50 pines, y el que mediante un bus se conecta a la placa base mencionada en el punto anterior. Finalmente se hará uso de los pulsos generados a partir del encoder para el cálculo de la velocidad.

Esta información, así como toda la empleada en el trabajo en relación con este dispositivo se han obtenido de su manual de usuario. [2]

Ilustración 1. Imagen del servopack. Fuente propia. Se muestra el dispositivo a emplear, ubicado en el banco de ensayos montado en el laboratorio.

4.1.2 Servomotor

Elemento cuyo control es el objetivo principal del conjunto. Un servomotor es un motor eléctrico, pero con una elevada capacidad de control de su velocidad, aceleración y posición, característica que lo hace especialmente útil en automatización y robótica. Usualmente está formado por un motor eléctrico y un sensor que lo retroalimenta en velocidad y posición, bien a él mismo o a aquel dispositivo que lo controla, como es el caso del servopack.

El modelo empleado en este proyecto es SGMAH-02AAA61D-OY, de una potencia de 200W, alimentación trifásica a 230V proporcionada por el servopack, con una velocidad nominal de 3000rpm (revoluciones/minuto) y un par nominal de 0,67 Nm. Dispone de un encoder incremental de 13 bits que envía mediante un cable serie datos acerca de su posición, velocidad y sentido al servopack.

Los encoder convierten el movimiento en una señal eléctrica que puede ser leída por algún dispositivo en un sistema de control de movimiento, tal como un controlador o un PLC. El encoder envía una señal de respuesta que puede ser utilizado para determinar la posición, velocidad o dirección. Un dispositivo de control puede usar esta información para enviar un comando para una función particular. En el caso de este modelo el encoder es incremental, lo que significa que no conoce la posición absoluta del servomotor, pero sí la diferencia entre posiciones para permitir el cálculo de la velocidad. [3]

Información obtenida de su manual de usuario así como de catálogos y sitios web de distribuidores del producto. [2]

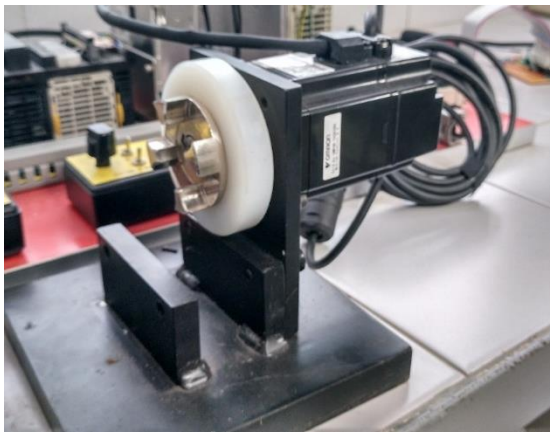

Ilustración 2. Imagen del servomotor. Fuente propia. Se muestra el motor a controlar, ubicado en el banco de ensayos montado en el laboratorio.

4.2 Funcionamiento del sistema

Debido a la gran cantidad de relaciones entre los distintos elementos que componen el conjunto es necesaria una explicación detallada y más visual de su funcionamiento. Pues de otra forma resulta abstracto y dificulta el entendimiento de las distintas decisiones de diseño adoptadas, así como de la función de cada dispositivo. Por ello se presenta una idea global del sistema que se va a emplear.

El objetivo primordial es el control del servomotor a través de cualquier dispositivo con capacidad de conexión a la red Wifi local. Aunque para el funcionamiento del conjunto hay gran cantidad de sistemas de comunicación trabajando, los más importantes, y en los que se centra este proyecto por su inexistencia en el mercado, son las dos siguientes interfaces de comunicación: Entre usuario y servidor y entre servidor y servopack. En ambas está implicado el microcontrolador pues es el hosting (máquina en la que se instala el programa) del servidor y el encargado de enviar señales eléctricas de control a través de la placa base. Por otro lado, el camino de la información es totalmente bidireccional pues el servomotor recibe órdenes del usuario y le devuelve los datos por la misma ruta, pues es necesaria una realimentación para el control. Esto facilita la explicación, pues se prestará atención a los canales en sí siendo indiferente el sentido de las señales. Una vez aclarado esto se presenta el funcionamiento del sistema, ordenado desde el usuario hasta el servomotor.

El usuario envía y recibe datos empleando un dispositivo que debe tener capacidad de conectarse a una red Wifi y que disponga de un navegador web, como puede ser Chrome, Firefox, Edge, etc. La red la genera y gestiona el router, asignando a cada dispositivo conectado una dirección IP, así que tanto el dispositivo del usuario como el servidor tendrán una. La conexión a la red es necesaria para poder intercambiar información en estado bruto con el servidor, y el navegador lo es para poder acceder a la dirección IP del servidor e interpretar el código HTML que éste le sirve. Dicho código es el que le indica al navegador qué debe aparecer en pantalla, los protocolos a seguir para la transferencia de datos y en definitiva todo aquello con lo que el usuario puede interactuar.

El código HTML programa la forma de conexión con el servidor, y el navegador es el que la lleva a cabo empleando la red Wifi local. Una vez establecida la conexión los datos viajan de forma inalámbrica hacia el router y de ahí a su destino, sea el servidor o el usuario.

El servidor es un programa informático encargado de enviar y recibir datos, así como de gestionar los clientes conectados o de servir y contener la propia página web. El servidor se carga en una computadora llamada hosting, siendo en este caso el microcontrolador. Sin embargo, en este proyecto tanto el programa servidor como el programa encargado de la gestión de entradas y salidas eléctricas están fusionados en uno solo. El mismo programa se encarga de mantener el intercambio de información con el cliente, de la recepción y envío de señales eléctricas a través de la placa base, y sobre todo de puente entre estos dos canales.

En el zócalo central de la placa base está conectado el microcontrolador, rodeado de diferentes circuitos electrónicos de diseño y construcción propios que adaptan las señales de comunicación entre servopack y microcontrolador. Los circuitos están soldados por un lado al zócalo del microcontrolador y por el otro conectados a los cables del bus paralelo de la placa. Cada uno de estos cables transmite la información de una determinada entrada o salida de control al puerto de entrada/salida del servopack. Éste se encarga de procesar las señales, de que el servomotor actúe y de devolver la realimentación de su operación.

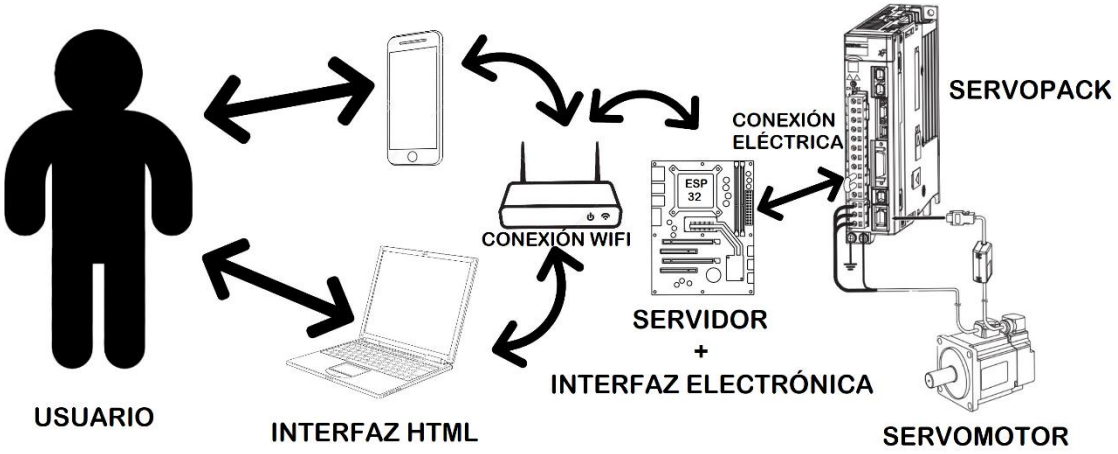

Ilustración 3. Esquema de funcionamiento del conjunto. Fuente propia. Se muestra un esquema del sistema con el objetivo de aclarar visualmente los diferentes canales de comunicación así como los elementos que los emplean.

4.3 Selección del microcontrolador

Un microcontrolador es un circuito integrado de elevada versatilidad, pues hay en su interior un microprocesador encargado de ejecutar los programas, la memoria dedicada a guardar datos e instrucciones y los módulos de entrada y salida para comunicarse y controlar el proceso en cuestión. Normalmente se han empleado para realizar tareas repetitivas y sencillas, pero en los últimos años han experimentado una mejora notable hasta ser capaces de controlar maquinaria y sistemas de mayor complejidad.

Una vez aclarado el funcionamiento del sistema ya se pueden conocer algunas de las características que serán necesarias para el microcontrolador. Ha de destacarse que, al ser el microcontrolador una pieza clave en el sistema, su elección condiciona a todo el conjunto y por ello la elección de uno u otro modelo aparece en este apartado de la memoria. A continuación, se muestran algunos de los microcontroladores con las características buscadas y que se han considerado para la selección.

4.3.1 Microcontroladores disponibles

4.3.1.1 ESP32

Este microcontrolador, diseñado por la marca Espressif Systems y lanzado al mercado a finales de 2016, pertenece a la familia de "System on a chip" de la marca, es decir, que incluyen todas las unidades funcionales en un único circuito integrado, lo cual lo hace barato y eficiente. Anclado a la placa de desarrollo ESP32 DEVKIT DO IT para facilitar su programación y alimentación. Las especificaciones más importantes se enumeran a continuación:

- Procesador Xtensa LX6 de 32 bits, con dos núcleos para multitarea, con una frecuencia de reloj máxima de 240 MHz. Con soporte para interrupciones.
- 520 KB de memoria SRAM para almacenar datos e instrucciones.
- 448 KB de memoria FLASH donde se almacena el código del programa, se puede aumentar hasta 4MB externos.
- Conexión Wifi con una velocidad de 150 Mbps, modo HT40 en la banda de frecuencia de los 2,4 GHz.
- Antena integrada en la placa de desarrollo.
- 25 entradas y salidas, algunas con capacidad de generar señales analógicas empleando la modulación por anchura de pulso (PWM) por sus siglas en inglés.
- Interfaz micro-USB para alimentación y comunicación serie.
- Alimentación: 5V de tensión continua.
- Programable empleando los lenguajes Arduino y C.
- Precio: 4 € – 10 €

Obtenidas de la ficha técnica del producto. [4]

Además, se trata de un módulo muy utilizado por la comunidad electrónica, sobre todo para aplicaciones relacionadas con conectividad inalámbrica, por lo que se dispone de mucha información y también de una gran cantidad de librerías con funciones especializadas.

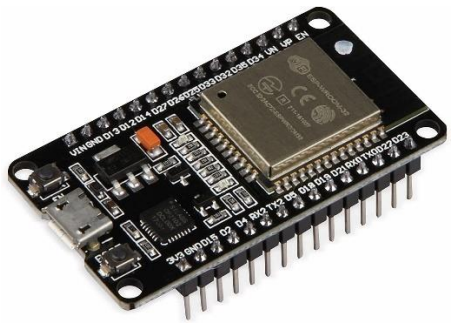

Ilustración 4. ESP32. Imagen obtenida de Amazon.es. Imagen del microcontrolador ESP32, insertado en su placa de desarrollo DEVKIT DO IT 1

4.3.1.2 ESP8266

Microcontrolador fabricado también por Espressif Systems, que apareció en el mercado el año 2014. Se trata de uno de los primeros productos de éxito de la compañía principalmente por ser de los primeros con capacidad Wifi. Se suele emplear con su módulo de desarrollo que le proporciona memoria FLASH adicional, una interfaz de comunicación y pines de entrada y salida más robustos conocida como NodeMCU. Sus principales características son:

- Procesador de 32 bit, Tensilica Xtensa LX106 a una frecuencia de reloj de 80 MHz. Un único núcleo, lo que no permite multitarea. Brinda soporte para interrupciones.

- Memoria RAM de instrucciones de 32KB y RAM de datos de 96 KB.
- Conexión Wifi con modo HT40 en las frecuencias de los 2,4 GHz. Velocidad hasta 72,2 Mbps.
- Antena integrada en la placa de desarrollo.
- 17 entradas y salidas de propósito general con una tensión de funcionamiento de 3,3V, con capacidad de generar PWM.
- Interfaz micro-USB para alimentación y comunicación serie.
- Alimentación: 5V de tensión continua.
- Programable empleando los lenguajes Arduino y C.
- Precio: 3 € – 6 €.

Datos obtenidos de la ficha técnica del producto. [5]

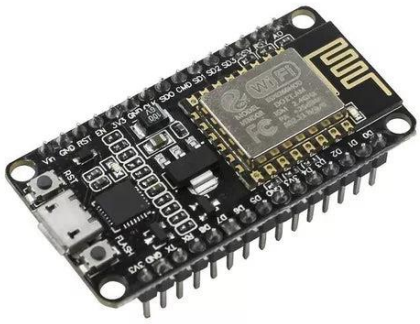

Ilustración 5. ESP8266. Imagen obtenida de Arca Electrónica.com. Imagen del microcontrolador ESP8266, insertado en su placa de desarrollo NodeMCU

4.3.1.3 Arduino MKR Wifi 1010

Desarrollado por Arduino.cc, compañía encargada al software y hardware libres. Esta placa de desarrollo dista bastante de los modelos más conocidos de Arduino, pero es la que más se adapta al proyecto. Sus especificaciones técnicas son:

- Procesador: SAMD21 Cortex®-M0+ 32bit low power ARM®. Con soporte para interrupciones y frecuencia de reloj de 48 MHz.
- Memoria FLASH de 256 KB y SRAM de 32 KB.
- Conectividad Wifi HT40 en las frecuencias de los 2,4 GHz, con antena integrada.
- Cuenta con 8 entradas y salidas digitales, así como 13 con capacidad para PWM, operando a 3,3 V.
- Interfaz micro-USB para comunicación y alimentación.
- Alimentación a 5V y soporte para baterías de 3,7 V.
- Programable empleando el lenguaje Arduino.
- Precio: 30 €.

Información obtenida del sitio web del fabricante. [6]

Al pertenecer a la marca Arduino cuenta con gran variedad de periféricos de la propia marca que dotan de mayor funcionalidad al producto, además de contar con una gran comunidad de desarrolladores de herramientas y programas.

Ilustración 6. Arduino MKR Wifi 1010. Imagen obtenida de BricoGeek.com.

4.3.2 Microcontrolador seleccionado

Una vez analizadas las diferentes opciones de microcontroladores que hay en el mercado y expuestas las características con más peso para este proyecto, se expone a continuación la justificación de la selección final.

En primer lugar, se ha descartado el modelo de Arduino debido a las inferiores capacidades técnicas que presenta y su elevado precio en comparación con las otras opciones. Su principal ventaja, ser un producto de Arduino, no se aprovecha en este caso, pues los otros microcontroladores también soportan el lenguaje de programación de Arduino así que también pueden hacer uso de su comunidad y de su entorno de desarrollo. Además, este modelo no ha tenido la aceptación que tuvo el primero por lo que muchos desarrolladores emplean el ESP32 o el ESP8266 para funcionalidad Wifi.

Entre los dos restantes se ha seleccionado el ESP32, pese a tener un precio un poco más elevado que el ESP8266, cuenta con mejores prestaciones tanto en memoria, en conectividad o velocidad de procesamiento y también por su mayor número de pines de entrada y salida, pues es siempre preferible contar con una buena cantidad de estos en especial si se trabaja con un puerto tan complejo como es el del servopack.

Finalmente se trabajará con el microcontrolador ESP32 en la placa de desarrollo ESP32 DEVKIT DO IT, para contar con todas las funcionalidades integradas en ella y para dotar al conjunto de mayor resistencia mecánica y facilidad de montaje.

4.4 Otros dispositivos empleados

4.4.1 Interfaz electrónica

Es la encargada de adaptar las señales eléctricas entre microcontrolador y servopack. Se trata de una placa base de prototipos de diseño y construcción propia, está formada por diferentes componentes electrónicos que deben permitir la conexión con el servopack ajustando los valores de tensión y corriente, además de tener un zócalo en el que conectar la propia placa de desarrollo del ESP32. La placa se alimenta con 230V de corriente alterna, y ella misma se encarga de regular la tensión a los valores necesarios que emplean sus diferentes circuitos y el microcontrolador.

Consta de 3 circuitos independientes donde cada uno adapta la señal de una entrada o salida concreta, pudiéndolos agrupar en:

- Filtros con amplificadores operacionales para adaptar las salidas analógicas a los rangos y especificaciones del servopack.
- Divisores de tensión para rebajar el voltaje de las señales digitales enviadas por el servopack de 24V a los 3,3V que requiere el ESP32.
- Transistores bipolares empleados para enviar al servopack las señales digitales del ESP32 pasando de 3,3V a 24V.
- Sistemas de alimentación de los distintos circuitos empleando convertidores y adaptadores electrónicos.

Una explicación más detallada tanto del diseño como la construcción de este elemento se expondrán más adelante en el apartado correspondiente de la memoria.

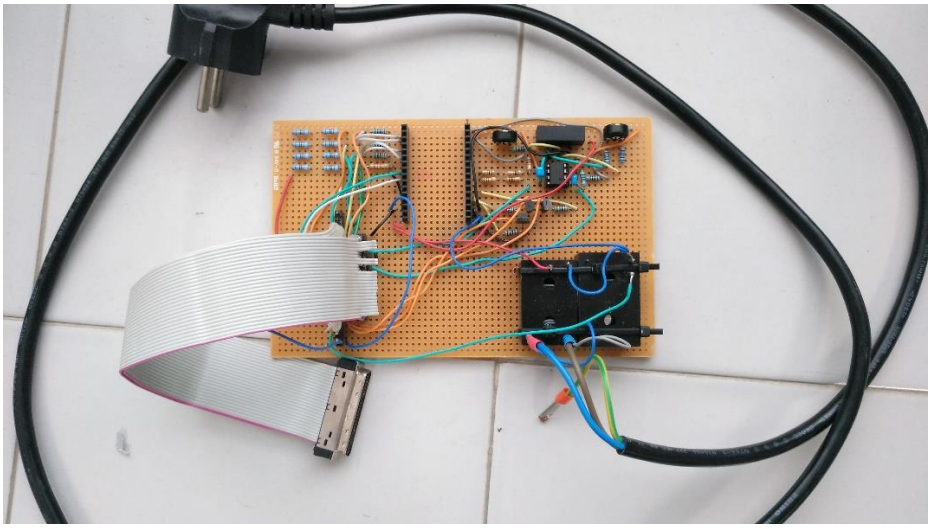

Ilustración 7. Interfaz electrónica. Fuente propia. Fotografía del prototipo de placa base sin el microcontrolador insertado.

4.4.2 Router Wifi

Elemento encargado de gestionar la conexión entre los dispositivos conectados a una red y de permitir que estos puedan enviarse datos de la forma más rápida posible. Su uso más habitual es la conexión a internet, pero en este proyecto es necesario por su funcionalidad como punto de acceso inalámbrico que permite la creación de una red Wifi local, de hecho, no es necesario que se contrate ningún servicio de Internet. Para la realización del proyecto no se usa ni exige ningún router en específico, pues no se van a emplear grandes redes ni altas tasas de envío de datos. Se han hecho pruebas en diferentes router domésticos e incluso con la funcionalidad de punto de acceso Wifi que permiten ciertos dispositivos como teléfonos inteligentes o tabletas dando en todos los casos buenos resultados.

5. DISEÑO Y ESPECIFICACIONES DEL SERVIDOR

5.1 Introducción

Un servidor en informática es un programa o dispositivo capaz de proporcionar funcionalidades o recursos a otros sistemas conectados a la misma red. Pueden desarrollar diversas funciones como alojar sitios web, gestionar accesos a dispositivos remotos, control de bases de datos, mensajería, entre otros. Normalmente se denomina servidor tanto al propio programa como al dispositivo que lo ejecuta, aunque si se quiere diferenciar, habitualmente el servidor es el programa informático y el "hosting" la máquina que lo contiene, el ESP32 en este caso.

Se trata de una tecnología con una evolución constante que dispone de gran cantidad de herramientas, protocolos y funciones preestablecidas que facilitan su implementación, gracias principalmente al papel fundamental que tiene en la red de internet. No se debe olvidar el objetivo del servidor en este proyecto, el cual es soportar el sitio web con los controles del servomotor y hacer que las señales de control indicadas en la página web lleguen al servopack.

Se exponen a continuación las tecnologías empleadas más importantes para el funcionamiento del servidor, así como la justificación de su uso, su utilidad y ciertas especificaciones.

5.2 Cliente y servidor

La arquitectura cliente-servidor es la más usada y la más simple de las diferentes arquitecturas con las que se puede diseñar un servidor. En ella hay básicamente dos entidades implicadas, el servidor y los clientes (pues puede haber varios por servidor). La comunicación se realiza a través de una estructura de peticiones por parte del cliente y de respuestas a estas por parte del servidor, aunque en este proyecto esto se ha alterado con el uso de Websocket y de un servidor de tipo asíncrono.

Sus principales ventajas para este proyecto son:

- Mejor eficiencia del conjunto al mantener el servidor la mayoría de la carga computacional, lo que mejora la respuesta del sistema.
- Facilita el mantenimiento y las actualizaciones, pues éstas sólo se tendrán que realizar en el servidor y no en cada cliente en específico.
- Facilita la integración entre varios sistemas, pues el servidor puede funcionar como medio de comunicación entre dispositivos de forma sencilla.

El servidor programado es capaz de soportar a varios clientes y de enviarles información acerca de la operación, sin embargo, bloquea las acciones demandadas por aquellos clientes que no sean el primero en conectarse por motivos de seguridad de operación. Esto se consigue con la capacidad de identificar a cada cliente por separado. En este proyecto cliente y usuario son sinónimos.

5.3 Servidor asíncrono

Los servidores asíncronos, al contrario de los síncronos, son capaces de manejar múltiples solicitudes de forma eficiente sin necesidad de respuesta inmediata. Sus principales características son: permitir la ejecución de nuevas solicitudes mientras se están procesando las anteriores de forma simultánea, esperar a completar la solicitud cuando sea más conveniente y mientras se realizan otras tareas y la capacidad de respuesta sin petición por parte del cliente. [7]

Es ésta última propiedad la que más ha influido en la elección de esta clase de servidores, pues junto con Websocket, no requiere la codificación explícita del protocolo de pregunta respuesta para su funcionamiento, si no que tanto cliente cómo servidor pueden enviar y recibir datos cuando sea necesario y sin necesidad de respuesta explícita.

Por otra parte, el servidor asíncrono suele ser la forma habitual de emplear Websocket cómo se verá más adelante. Pues permite el envío de pequeños paquetes de información de forma rápida sin que el cliente perciba ninguna demora y trabajar con eventos independientes.

5.4 Protocolo de envío de página (HTTP)

“Hypertext Transfer Protocol” es el nombre del protocolo de comunicación más empleado en la web. Su estructura está basada en la estructura cliente-servidor de forma rígida, pues en su caso la iniciativa en la comunicación siempre la tiene el cliente enviando una petición y posteriormente el servidor deberá enviar una respuesta con los recursos solicitados. [8]

Esto es un problema para el objetivo del trabajo si se usara también para el intercambio de información, pues se pretende un control en tiempo real y asíncrono. De este modo solo se emplea este protocolo para el envío al cliente del código HTML que contiene la página web, pero para el posterior intercambio de datos se emplean otros métodos más modernos que sirven para pequeños intercambios de datos.

Primero, se debe establecer la conexión, habitualmente mediante TCP (“Transfer control protocol”), pues en este se basa HTTP, además es necesario que tanto cliente como servidor tengan asignada una dirección IP. En este protocolo los servidores HTTP ocupan el puerto de conexión 80 por defecto y será el que se utilice en este proyecto para no tener que especificar el puerto cada vez que se acceda al servidor.

Un puerto es un punto virtual en el que comienzan y terminan las conexiones de red, están basados en software y los gestiona el sistema operativo de un ordenador. La mayoría de los puertos están reservados para ciertos protocolos, lo que le permite al ordenador diferenciar distintos tipos de tráfico. Mientras que las direcciones IP permiten que los mensajes vayan hacia y desde dispositivos específicos, los números de puerto permiten dirigirse a servicios o aplicaciones específicos en esos dispositivos. [9]

Segundo, el cliente debe enviar una petición mediante su navegador. Este es el motivo de porque aparece http//: antes de la dirección a buscar, pues indica el protocolo a seguir para el envío y recepción de la página. Hay diversos métodos de peticiones posibles como GET (solicita un recurso), POST (envío de datos al servidor), DELETE (elimina información), entre otros. En este caso solamente se utiliza el método GET pues el cliente pretende indicar al servidor que le proporcione un recurso. [10]

En tercer y último lugar, el servidor debe responder a la petición del cliente realizando la acción indicada por el método. El servidor del proyecto está preparado para en cuanto se conecte un cliente y solicite mediante el método GET la página, le proporcione todo el código HTML con la información del sitio web.

Finalmente se debe añadir que este protocolo lo realizan de forma autónoma el dispositivo del usuario y las librerías empleadas en el servidor, por lo que solo se han codificado funciones que dan ciertas indicaciones de la información a transmitir, o el método de transferencia. Además, tanto las peticiones como las respuestas están codificadas en ASCII y en ellas se indica también tamaños, opciones, versiones e información adicional estructurada tal y como indica el protocolo.

5.5 Protocolo de intercambio de información (Websocket)

El protocolo HTTP no es capaz de modificar en tiempo real ciertas partes de la página web si no es volviéndola a enviar al cliente cada vez que haya un cambio. Por lo que se hace necesario establecer un intercambio de datos con una herramienta capaz de modificar ciertos valores de la página web como pueden ser indicadores de velocidades o señales de control sin tener que cargar de nuevo la página web, lo que ralentiza en gran medida la operación. Además, se requiere de la capacidad del servidor de enviar valores al cliente sin que este los solicite, pues es así como debe funcionar un tablero de mandos donde en todo momento se deben poder ver las variables de operación.

Un conjunto de tecnologías que permite dicho objetivo es AJAX (Asynchronous JavaScript and XML), basada en HTTP y que mediante una serie de peticiones asíncronas al servidor y el uso de objetos XMLHttp de comunicación en segundo plano permiten recargar ciertas secciones de la página web, haciéndola más interactiva. XML es un formato de almacenamiento de información que utilizaba AJAX en un principio, aunque ha sido desplazado por otro formato llamado JSON (Java Script Object Notation). Sin embargo, el motivo por el que no se ha empleado AJAX es que es menos eficiente que Websocket para conseguir que el servidor envíe datos sin un requerimiento del usuario. Es posible realizar esta operación, pero requiere de peticiones automáticas del dispositivo del cliente para que el servidor le brinde dicha información, lo que aumenta la carga sobre el dispositivo cliente y se ocupa mayor ancho de banda para la comunicación, es en definitiva una solución menos óptima que la que se presenta a continuación.[11]

La opción empleada es el protocolo Websocket, una tecnología más moderna que AJAX, aunque también fundamentada desde HTTP. La principal característica de este protocolo es el establecimiento de un canal de comunicación persistente y bidireccional entre el cliente y el servidor, que permite la transferencia de datos de forma simultánea y en cualquier dirección sin una petición previa por parte del cliente. Además, los mensajes que se envían servidor y cliente están mucho más optimizados y ocupan un menor espacio, por ello es más eficiente y ocupa un menor ancho de banda que AJAX para pequeños trasvases de datos. Hay que añadir que Websocket también permite recargar sólo ciertos componentes de la página web sin cargarla entera otra vez. [12]

Su funcionamiento se inicia sobre el protocolo HTTP que será el que indicará tanto a servidor como cliente que se va a utilizar Websocket y ambas partes deben definir ciertos servicios y protocolos mutuos, esto se conoce como “handshake”, apretón de manos. Una vez realizado se abre el canal de comunicación que no se volverá a cerrar hasta la desconexión del cliente, y tanto usuario como servidor pueden comenzar a enviar y recibir información sin necesidad de peticiones empleando una única conexión. Por lo que a partir de ese momento pierde toda relación con HTTP. El puerto indicado por defecto para interactuar con Websocket es el 81, pues se está

creando un servidor secundario necesario para la implementación de este protocolo que se establecerá en paralelo con el servidor principal que contiene la página web y que está en el puerto 80 al ser HTTP. [13]

La forma de actuar de Websocket y de codificar su uso es mediante los siguientes eventos:

- onopen: Se ejecuta este evento cuando se realiza la conexión cliente-servidor.
- onmessage: Cuando se recibe un mensaje por parte del servidor o del cliente.
- onclose: Cuando se desconecta el cliente.
- onerror: Cuando sucede algún error en la conexión.

Estos eventos aparecen tanto en el servidor cómo en el cliente, facilitando la escritura del código que deberá responder a estos eventos de la forma deseada. Websocket es una tecnología soportada por todos los navegadores y dispositivos actuales, JavaScript tiene librerías incluidas para su control, así como el ESP32 tiene también esta capacidad con librerías externas, lo que facilita enormemente su programación. Pese a que Websocket requiere cierto esfuerzo computacional por parte del servidor, proporciona gran cantidad de ventajas, pues este proyecto tiene como objetivo la razón por la que se creó Websocket, la comunicación en tiempo real entre cliente y servidor. [14]

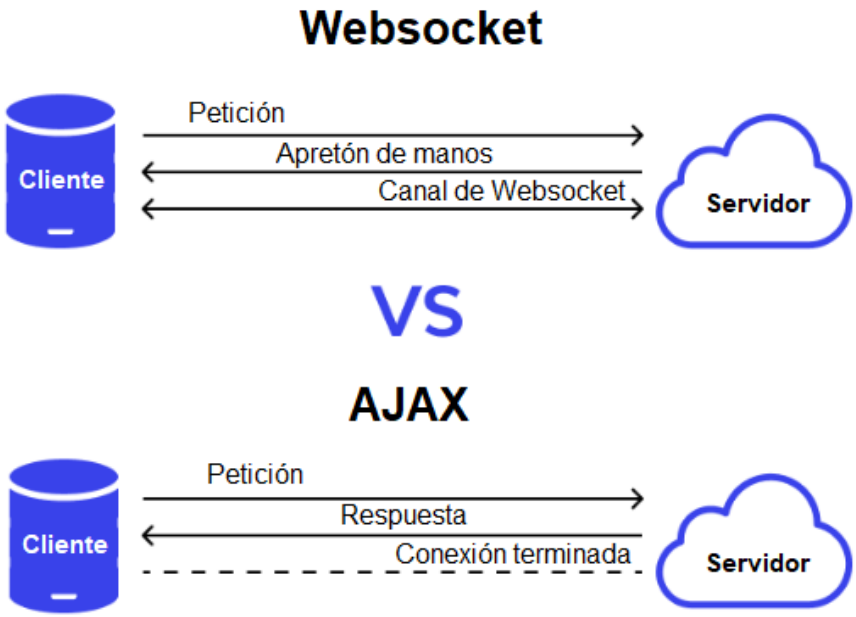

Ilustración 8. Ajax Vs Websocket. Imagen obtenida de "Geekflare". Comparativa de los diferentes esquemas de comunicación que emplean AJAX y Websocket.

5.6 Paquetes de información (JSON)

Por el canal de comunicación creado por el Websocket se envían paquetes de información estructurada bajo el formato JSON (JavaScript Object Notation). Se trata de un estándar que pretende organizar toda la información que hay en un paquete y de proveer de herramientas para su escritura o lectura. Su principal ventaja es que es igual de sencillo de interpretar y redactar por un humano que por una máquina.

Permite la creación de una estructura organizativa en la que se pueden agrupar los datos en conjuntos y nombrarlos de una forma determinada, para ello se emplea lo que se conoce como objetos, que es el conjunto del nombre asignado a ese dato (se le suele llamar clave) y de su valor. Estos dos campos suelen ir entre comillas, el de nombre habitualmente suele ser un texto, no obstante, el de valor puede contener tanto texto como números e incluso vectores de otros objetos, creando así la jerarquía. [15]

Tanto cliente como servidor crearán un archivo JSON en el cual introducirán los valores de las variables de control con el nombre indicado y lo enviarán mediante el Websocket, una vez recibido, se deberá hacer referencia a cada variable usando el nombre dado para conocer lo que contiene. El motivo principal de su uso en este proyecto es por su capacidad de envío de todas las variables de control de forma simultánea y por su facilidad de escritura y lectura por la parte del código tanto en JavaScript con funciones propias, como en la parte del ESP32 con librerías externas. Se trata de un formato compacto y rápido de procesar, lo cual es importante para conseguir una comunicación fluida y, además, lo soportan todos los navegadores y dispositivos actuales.

6. INTERFAZ HTML (USUARIO-SERVIDOR)

6.1 Introducción

Puesto que el objetivo del trabajo es controlar el servomotor a través de una conexión Wifi, la forma más idónea es la creación de un servidor de funcionamiento similar a los que ofrecen servicio en Internet. Éste tendrá almacenado el código de una página web que enviará al usuario, y establecerá la interfaz de comunicación entre ambas partes. Se ha empleado el lenguaje o estándar HTML en su quinta versión para construir la estructura de la página, así como otros lenguajes que se basan en lo indicado por el HTML y permiten modificar el diseño gráfico o el comportamiento del sitio web, estos son CSS y JavaScript. El código se envía al dispositivo del usuario y es el navegador que tenga instalado el que interpretará la programación y lo mostrará por pantalla.

El software empleado para la codificación del sitio web es Visual Studio Code de Microsoft, se trata de un editor de texto muy inclinado hacia la programación y de acceso gratuito, que brinda ciertas ayudas como autocompletado o indicaciones de errores en la sintaxis. Posteriormente este código se almacenará en una variable del servidor, pero esto se tratará en el apartado correspondiente.

Este apartado tiene la finalidad de presentar el sitio web, su funcionalidad e introducir los lenguajes de programación involucrados. La explicación del código, así como el código completo aparecen en los anexos de la memoria.

6.2 Presentación del sitio web

Se presenta en este apartado el aspecto definitivo de la página web, así como la explicación de sus funcionalidades y controles. Será lo que aparezca en la pantalla del navegador del dispositivo al buscar la dirección IP de conexión con el microcontrolador. Debido a que HTML es un estándar sometido a cambios y actualizaciones que deben seguir inmediatamente los navegadores para traducir de forma satisfactoria su información, es posible que el sitio web se muestre de forma ligeramente distinta en función del navegador empleado y de su versión. Además, debido a que cualquier dispositivo con navegador puede interactuar con la página web, surge un nuevo inconveniente que es adaptar los tamaños de controles y texto para que sea cual sea la resolución y tamaño de pantalla puedan ser utilizados y leídos.

Para los ejemplos de visualización de la página aquí mostrados se ha empleado un portátil con una resolución de 1920x1080, en disposición horizontal, visualizada desde el navegador Google Chrome en la versión 113.0.5672.93

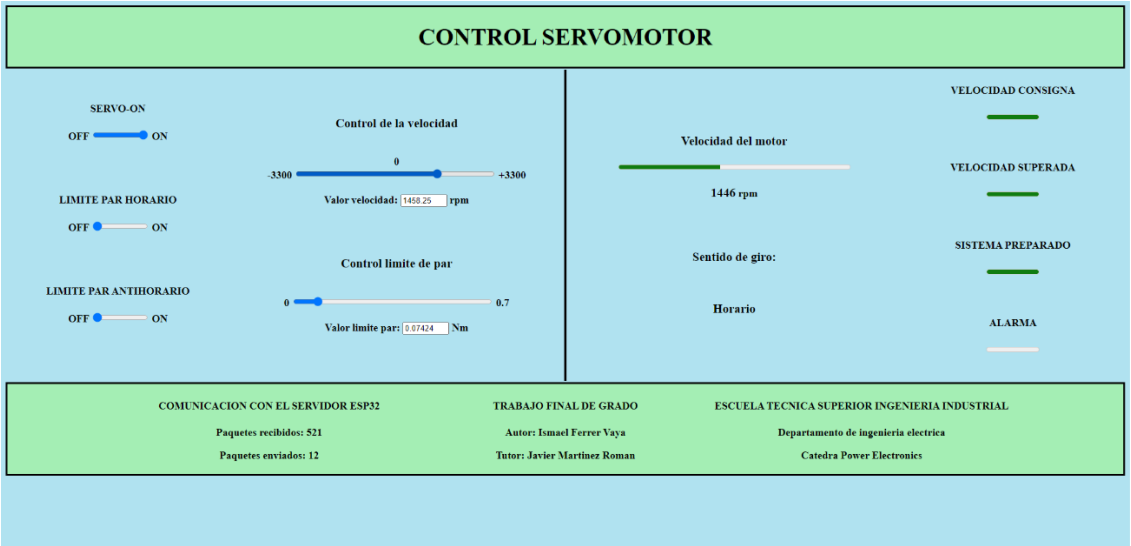

Ilustración 9. Sitio Web. Fuente propia. Captura de pantalla de la página web que aparece al buscar en el navegador la dirección IP del servidor mientras se operan sus controles.

6.2.1 Entradas de control

Se dividen en dos partes diferenciadas, las digitales y las analógicas, diferenciadas así por el tipo de señal eléctrica que representan y también por su rango de entrada de valores. Las digitales solo pueden representar dos estados lógicos: ON (1) entrada activa, OFF (0) entrada inactiva. Las analógicas debido a que se ha empleado en el servidor una codificación de 12 bits para ellas se pueden seleccionar 4096 valores distintos que pretenden representar una selección continua tanto de velocidad como de par. Esto se traduce a una señal eléctrica cuyo valor de continua a la salida del microcontrolador puede variar entre 0 y 3,3 voltios con la resolución indicada, se consigue empleando modulación por anchura de pulso (PWM).

Entradas digitales, implementadas con una barra deslizante entre dos estados a modo de interruptor:

- SERVO-ON: Debe activarse antes de realizar cualquier acción. Avisa al servopack que va a comenzar la operación del servomotor.
- LÍMITE PAR HORARIO: Al activarse limita automáticamente el límite de par máximo exigible en el sentido horario de giro al valor de un parámetro del servopack elegido por el usuario y guardado en su memoria. Aunque la entrada analógica de par solicite un límite de par mayor nunca podrá superar el indicado por esta señal de control en el caso de estar activa.
- LÍMITE PAR ANTIHORARIO: Funcionamiento idéntico a la entrada anterior, pero en el sentido de giro antihorario.

El sentido de giro horario o antihorario se considera el visto desde la parte frontal del servomotor, es decir, desde donde se ve el eje.

Entradas analógicas:

- Control de la velocidad: Dispone de dos selectores para modificar su valor, un control deslizante que se desplaza arrastrándolo bien con el ratón o con el dedo en caso de una pantalla táctil, y un cuadro de texto donde se puede escribir, interactuando sobre él, el valor requerido para la velocidad, medida en revoluciones por minuto. Los límites de

velocidad son 3300 rpm en los dos sentidos, pues el valor en negativo indica sentido de giro antihorario y en positivo horario. El valor solicitado se imprimirá en el cuadro de texto en el caso de que la variación se solicite con la barra deslizante.

- Control del límite de par: Funcionamiento similar a la entrada anterior, con un control deslizante y un cuadro de entrada de texto. En este caso el límite de par se aplica en los dos sentidos de giro simultáneamente que son considerados siempre como valores positivos. Su límite de operación va desde 0 hasta 0.70053 Nm.

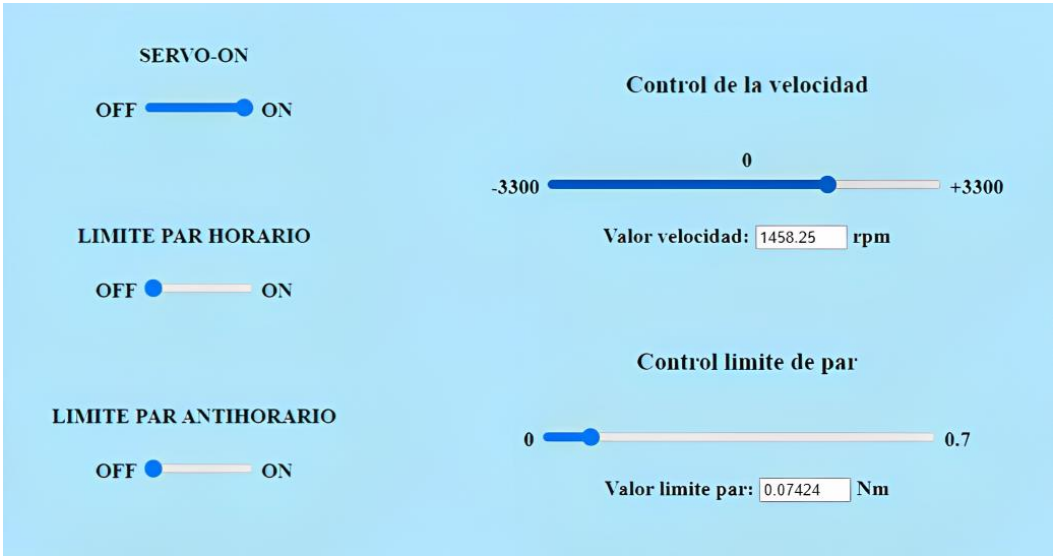

Ilustración 10. Entradas de control. Fuente propia. Captura de pantalla de la sección de entradas digitales y analógicas.

6.2.2 Salidas de control

Divididas en dos partes, las señales recibidas debidas a los pulsos generados por el encoder y las que representan salidas digitales del servopack. Las señales del encoder indican la velocidad real de giro del servomotor, así como su sentido de giro, representan valores enviados directamente desde el servidor donde se ha procesado toda la información de los pulsos recibidos. Las salidas digitales sólo pueden indicar dos estados, activa e inactiva, y cada una de ellas representará una variable de operación diferente del sistema.

Salidas debidas al encoder:

- Velocidad del motor: Indica en revoluciones por minuto la velocidad real a la que está girando el servomotor, sin importar el sentido de giro. Se representa de dos formas: con una barra de progreso, que aparece más o menos llena en función de la velocidad, y con un cuadro de texto que muestra el valor de la velocidad. Cabe destacar que este número aparecerá de color rojo cuando se supere la velocidad nominal de 3000 rpm.
- Sentido de giro: Indica el sentido de giro del motor: horario, antihorario o parado si la velocidad es 0. Se imprime la palabra correspondiente en pantalla debajo del título de esta salida.

Salidas digitales, implementadas con una barra de progreso entre dos estados que actúa a modo de indicador luminoso cuando están activas:

- VELOCIDAD CONSIGNA: Se activa cuando la velocidad demandada y la obtenida coinciden según la medición del servopack.
- VELOCIDAD SUPERADA: Se activa cuando se supera cierta velocidad indicada por el usuario en un parámetro del servopack.
- SISTEMA PREPARADO: Activa cuando todo el sistema está listo para recibir órdenes de control, cuando se ilumine dicha salida es cuando se debe activar la entrada SERVO-ON.
- ALARMA: Activa cuando sucede algún problema en el sistema, una vez éste es detectado el servopack detiene su operación y el servidor deshabilita las entradas de control indicando una velocidad y un límite de par nulos y desactivando la señal SERVO-ON.

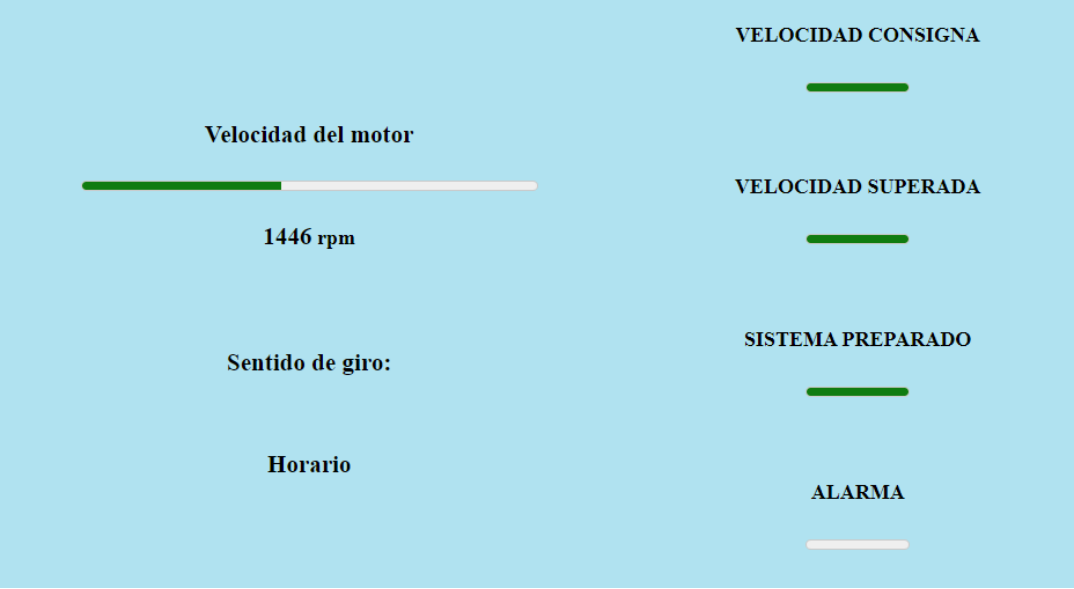

Ilustración 11. Salidas de control. Fuente propia. Captura de pantalla de la sección de salidas digitales y de las mediciones del encoder.

6.2.3 Indicadores de comunicación con el servidor

Estos dos valores indican la cantidad de paquetes de información que se han enviado al servidor y los que se han recibido de este desde que se ha iniciado la página web. El servidor los envía cada 0,1 segundos con todas las señales que recibe del servopack analizadas e interpretadas, mientras que la página envía datos cada vez que se produce algún cambio en alguna entrada. Su principal objetivo es mostrar el estado de la comunicación con el servidor.

Ambos indicadores funcionan de la misma forma, imprimiendo por pantalla la cantidad de paquetes en cada caso.

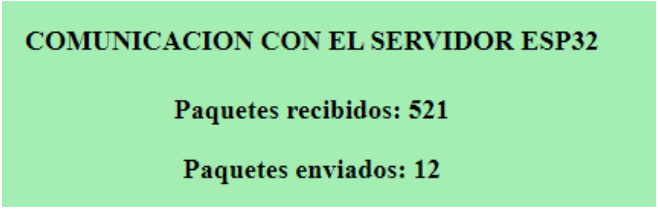

Ilustración 12. Comunicación con el servidor. Fuente propia. Captura de pantalla de la sección de comunicación y recuento de datos enviados y recibidos

6.3 Lenguajes empleados

6.3.1 HTML

HTML (HyperText Markup Language) es el lenguaje estándar utilizado para crear la base de cualquier página web. Fue creado en 1991 por Tim Berners-Lee, un científico de la computación que tenía como objetivo proporcionar una herramienta que facilitara la divulgación de textos e imágenes a través de la web, por lo que no se hizo para codificar las modernas y versátiles páginas que hoy son la norma. Por ese motivo se ha ido actualizando a lo largo de los años añadiendo nuevas funcionalidades hasta llegar a la versión que se usa en este proyecto, HTML 5, lanzada en 2014. Sin embargo, no se ha modernizado lo suficiente y dentro del propio HTML se deben utilizar otros lenguajes compatibles para codificar el diseño y las funcionalidades del sitio web, como son CSS y JavaScript. [16]

Se trata de un lenguaje de etiquetas, pues su principal objetivo es proveer la estructura de la página web, indicando títulos, texto, ventanas de interacción o tablas. Cada etiqueta tiene una o varias finalidades como puede ser delimitar el contenido, incluir tablas, títulos o texto, indicar la codificación en otros lenguajes o aplicar ciertas propiedades. La mayoría de las etiquetas están compuestas por la etiqueta de apertura como por ejemplo “<h1>” y por la de cierre “</h1>”, y entre ellas se incluye el contenido que suele ser habitualmente el texto que aparece en el sitio web. Otras etiquetas como la de salto de línea “
” o la que incluye entradas de interacciones “<input>” no requieren la de cierre.

Dentro de las propias etiquetas de apertura se pueden escribir ciertas palabras, llamadas atributos, que servirán para darle un identificador a la etiqueta y lo que contiene, definir valores a algunos tipos concretos o asignar funcionalidades que se deberán cumplimentar a parte con JavaScript. Se expone un ejemplo con la etiqueta <input>: <input onchange= "enviarDato(this)" id="control_P_CL" type="range" step ="1">

Los atributos de identificación como son id o class sirven para que desde otros puntos del programa u otros lenguajes se pueda hacer referencia a estas etiquetas con diversas finalidades, como puede ser cambiar su estilo gráfico o aplicarle una determinada funcionalidad. La diferencia entre ambas es que mientras id sólo hace referencia a una etiqueta, class se utiliza para agrupaciones de etiquetas que tengan un significado como conjunto.

Las etiquetas HTML básicas más utilizadas en el proyecto son las siguientes:

Etiquetas orientadas al texto, como “<h1>”, “<h3>” o “<h4>” cuya finalidad es aumentar el tamaño del texto que se escribe entre ellas a modo de título, o la etiqueta “<p>” para incluir texto en párrafos con tamaño normal de fuente.

Etiquetas para creación de tablas, siendo la que incluye la tabla “<table>”. Dentro de ella se pone la etiqueta que indica una nueva fila “<tr>” y la que indica una nueva columna “<th>”. Una vez establecidas filas y columnas, se escribe entre las etiquetas aquello que se vaya a incluir en cada celda de la tabla, normalmente esto se especifica entre las etiquetas de columna. Se suelen utilizar para ordenar el contenido del sitio web tanto en horizontal como en vertical.

La etiqueta “
” sirve para introducir una línea en blanco y “ ” para centrar en la página todo aquello que se escriba en su interior. Bastante usadas para gestión de espacio y separación entre elementos.

Etiquetas de entrada y salida, como son “<input>”, que permite definir varios tipos de entrada de datos por parte del usuario y “<meter>” que es una salida de datos a modo de barra de progreso en función del número que le corresponda indicar.

Cabe destacar que todas las etiquetas indicadas a excepción de “<input>” y “
” requieren de su respectiva etiqueta de cierre. Varias etiquetas de uso más específico serán explicadas a medida que lo requiera la presentación del código, pero se ha considerado oportuno introducir las más genéricas para agilizar la explicación posteriormente.

6.3.2 CSS

Las siglas de "Cascading Style Sheets", hojas de estilo en cascada. Se trata de un lenguaje encargado de aplicar sobre la página web todo aquello que esté relacionado con su estilo o su diseño visual o incluir determinados efectos que mejoren la presentación o la accesibilidad, y todo agrupado en un único lugar. Se complementa con el código HTML para actuar y aunque se puede aplicar de varias formas, la empleada en este trabajo es incluir el código CSS dentro de la etiqueta “<style>” en el código HTML.

La forma de trabajar de CSS es hacer referencia a etiquetas empleadas en HTML y que hayan sido identificadas mediante los atributos de identificación id o class. A cada elemento se le podrán asignar unos valores a sus atributos de estilo como puede ser el tamaño, color, posición o forma indicándolos entre corchetes después del identificador. La forma de codificar el color RGB es mediante hexadecimal de 6 dígitos, en los que los dos dígitos de la izquierda indican la cantidad en rojo, los del medio la cantidad en verde y los dos últimos en azul.

Cabe destacar que para lograr una buena adaptabilidad del sitio web a los diferentes dispositivos que pueden acceder se ha hecho uso de las herramientas que permiten modificar el tamaño de los componentes de la página de forma relativa a las dimensiones del elemento que las contiene. Permitiendo que los elementos mayores se adapten a las dimensiones del dispositivo, pero sin perder su ordenación interna.

Se expone un ejemplo de código en este lenguaje, cuyo objetivo es actuar sobre la class piepagina, dándole un fondo de color verdoso y unos bordes negros sólidos de tamaño 3 píxeles.

```
.piepagina{
    background: #a4eeb4;
    border: 3px solid #000000;
}
```

6.3.3 JavaScript

Se trata de un lenguaje de alto nivel que se ejecuta en el navegador del cliente de forma nativa usado principalmente en desarrollo web para la creación de páginas dinámicas e interactivas. Su principal propósito es implementar funcionalidades en el sitio web como cálculos o animaciones, establecer canales de comunicación con el servidor o realizar la gestión de la comunicación. En definitiva, se encarga de hacer actuar a la página de la forma deseada empleando variables, funciones, estructuras de control y operaciones aritméticas y lógicas. En este aspecto es bastante más similar a los lenguajes clásicos de programación pues está fuertemente inspirado en C++ y Java.

Presenta una gran compatibilidad con los lenguajes de los apartados anteriores, y su código se escribe dentro de la etiqueta de "<script>" en el HTML. La forma de interactuar con las etiquetas es también mediante los identificadores id y class, teniendo la capacidad de conocer o modificar tanto su diseño como el contenido de esta. Para ello hay ciertas funciones de lectura o escritura incluidas en JavaScript que permiten realizar estas acciones indicando el identificador de etiqueta y diferenciando de si se trata de texto o de números.

Su uso es fundamental para este proyecto pues es el encargado de gestionar los datos que llegan desde el servidor y de enviar todas las acciones realizadas por el usuario, por lo que se han codificado en JavaScript funciones dedicadas a su control. Además, establece el canal y el modo de comunicación Websocket con el servidor facilitando esta tarea al programador con ciertas funciones, variables y eventos incluidos en el propio lenguaje. Por último, brinda soporte y facilita el uso de JSON ("JavaScript Object Notation"), el formato de intercambio de datos ligero y rápido empleado en este trabajo.

Puesto que este lenguaje es el más complejo de los empleados en la parte del cliente del servidor (el usuario), se hará una explicación detallada de funciones, variables y operaciones a medida que se presente su código en los apartados correspondientes de los anexos.

7. PROGRAMACIÓN DEL MICROCONTROLADOR

7.1 Introducción

El microcontrolador es la pieza fundamental de unión entre el usuario y el servomotor, pues se encarga de gestionar el servidor y las señales eléctricas de comunicación con el servopack. Finalmente se ha optado por el ESP32 para la realización de este proyecto y el lenguaje de programación Arduino para programarlo. En este capítulo se muestra una breve explicación acerca del lenguaje Arduino, los elementos del microcontrolador implicados y las librerías de código externas empleadas. El código y su explicación se incluyen en los anexos de la memoria.

7.2 Arduino

Arduino es un proyecto de hardware y software de código abierto que pretende ser una opción versátil tanto para aficionados cómo para profesionales del mundo de la electrónica, la programación y la robótica. Son conocidas las placas Arduino por presentar una gran versatilidad con un enfoque muy amigable para iniciados en esta disciplina, incluso en este proyecto se ha tenido en cuenta la posibilidad de emplear una de sus placas como microcontrolador. La idea surge en 2005, cuando Massimo Banzi, junto con un grupo de ingenieros y diseñadores en el "Interaction Design Institute de Ivrea", Italia, deciden desarrollar una placa y un lenguaje más asequible económicamente para que los estudiantes tuvieran una menor barrera de entrada al mundo de la microelectrónica. [17]

Para este proyecto solamente es importante el lenguaje de programación, el cual está basado en C y C++, lenguajes muy comunes en informática e ingeniería. Su principal ventaja es que tiene una comunidad de desarrolladores, tanto aficionados como profesionales, muy implicada en la creación de proyectos, por lo que hay mucha información en la red y una amplia variedad de librerías de funciones que aportan una enorme versatilidad al código. Por otro lado, se trata de un lenguaje muy enfocado a los microcontroladores, así que la gestión de señales eléctricas es directa y dispone de muchas funciones propias que tienen gran interés en el mundo de la robótica. Adicionalmente dispone de un entorno de desarrollo integrado, el Arduino IDE. Se trata de un software para PC gratuito capaz de compilar el código en Arduino y cargarlo en el microcontrolador que se tenga configurado, no es necesario que se trate de una placa propia pues también permite el uso de los demás microcontroladores del mercado como es el caso del ESP32. También permite la comunicación por el puerto serie del microcontrolador, imprimiendo la información recibida por pantalla, lo que lo hace muy útil para detectar errores de operación o como comunicación directa.

Todas estas ventajas han propiciado su uso en este proyecto frente al otro lenguaje que soporta el ESP32, el lenguaje C. Que, pese a ser un lenguaje conocido y estudiado a lo largo de la carrera se hace notoria su inclinación hacia la informática más que a la electrónica, lo que dificulta y alarga la codificación del uso de señales eléctricas o la gestión de la interfaz Wifi del microcontrolador. Por lo que, aunque es el lenguaje seleccionado por el fabricante del microcontrolador para mostrar las posibilidades del ESP32, en este proyecto se ha optado por su programación en Arduino. Por otro lado, al ser el Arduino de código abierto y ampliamente utilizado se pretende que sea sencilla la modificación del código con el objetivo de mejorarlo o dotarlo de nuevas funcionalidades por parte del usuario.

7.3 Características internas del microcontrolador

Este apartado pretende indicar las características tanto de hardware como de software más importantes del microcontrolador orientadas a su programación o a su uso, como puede ser su capacidad de generar PWM, de interrupciones, los pines empleados, así como ciertas especificaciones que dan soporte a lo que requiere el código. Toda esta información se ha obtenido de su ficha técnica, de catálogos o de sitios web especializados.

7.3.1 Patillaje empleado

El ESP32 cuenta con 30 pines, de los cuales 1 es de alimentación (VIN), 2 son de referencia de masa (GND), uno da una tensión de referencia de 3,3V y otro es la entrada de reinicio por hardware (EN). Por lo que finalmente se dispone de 25 pines para entrada/salida, más conocidos como Entradas o Salidas de Propósito General o GPIO (General Purpose Input Ouput). En el capítulo de la interfaz electrónica se detallará su ubicación en la placa de desarrollo. Los GPIOs empleados han sido:

- 23: Con capacidad para PWM, genera la señal analógica de control de límite de par.
- 22: Con capacidad para PWM, genera la señal analógica de control de velocidad.
- 2: Genera la señal digital SERVO-ON.
- 4: Genera la señal digital de límite de par horario.
- 15: Genera la señal digital de límite de par antihorario.
- 26: Recibe los pulsos debidos al encoder y actúa mediante interrupciones.
- 25: Recibe los pulsos debidos al encoder y actúa mediante interrupciones.
- 35: Recibe la señal digital de velocidad de consigna.
- 34: Recibe la señal digital de velocidad superada.
- 39: Recibe la señal digital de sistema preparado.
- 36: Recibe las señal digital de alarma.

Se hace notar que lo que se considera entradas o salidas para el usuario está totalmente invertido desde el punto de vista del microcontrolador. Pues el ESP32 envía al servopack aquello que el usuario considera entradas, y recibe lo que el usuario considera salidas de control.

7.3.2 Capacidad multinúcleo

El ESP32 cuenta con dos núcleos en su procesador lo que le permite realizar multitarea. Esto en el código se traduce en que se pueden ejecutar simultáneamente dos funciones. Se trata de una funcionalidad que ha resultado ser imprescindible para lograr una optimización del proceso, pues al estar empleando el canal multidireccional de Websocket el servidor debe poder estar ejecutando el proceso de recopilación y envío de datos al mismo tiempo que está a la espera de recibir nueva información o incluso gestionando estos datos recibidos. Estas dos funciones de envío y recepción de información son las dos funciones principales del código y el hecho de que se ejecuten paralelamente facilita su control y permite una interacción casi inmediata entre usuario y servomotor.

7.3.3 Soporte para interrupciones

Las interrupciones son eventos o señales que interrumpen temporalmente la ejecución normal del programa o proceso en curso, al producirse ejecutan una función específica que ha debido ser codificada por el programador llamada rutina de interrupción.

Esta característica tiene mucha importancia para la función de medición de la velocidad a partir de los pulsos del encoder, pues estos llegan al microcontrolador con unas velocidades y frecuencias dependientes de la velocidad del servomotor, por lo que aquel debe estar preparado para recibirlas en el momento en el que lleguen y además deben ser tratadas de forma individual. La solución ideal es el uso de interrupciones del procesador, que se pueden activar de diversos modos como flanco de subida o bajada de tensión, y que al producirse permiten el recuento de pulsos empleando las rutinas de interrupción. [18]

7.3.4 Generación de señal analógica mediante PWM

Modulación por anchura de pulso, de sus siglas en inglés (PWM) es un método muy empleado en electrónica para la generación de señales analógicas. Para lograrlo hace uso de una señal cuadrada de una frecuencia elevada que varía su valor medio de tensión entre 0V (valor mínimo) y 3,3V (valor máximo). Para un período de señal constante, el tiempo que la onda cuadrada esté a nivel alto en relación con la duración total del período se denomina ciclo de trabajo o "duty cicle" siendo esta la variable de control fundamental de este proceso de modulación. Cuanto mayor sea el tiempo que la señal está a nivel alto (elevado valor de ciclo de trabajo) mayor será el valor medio de tensión del conjunto y viceversa. Es precisamente el valor de la tensión media el que interesa controlar, pues para valores elevados de la frecuencia de la señal, debido a los transitorios eléctricos, el dispositivo al que se le envía dicha señal tendrá un comportamiento similar al que le correspondería si se le hubiera enviado una señal de tensión continua de valor de voltaje igual al valor medio entregado mediante PWM. Adicionalmente se trata de un proceso muy barato, pues requiere solamente de una salida digital y de un generador de pulsos interno con capacidad de modulación. En algunas ocasiones se filtra esta señal para eliminar los armónicos de mayor frecuencia y obtener una tensión de señal continua real con cierto rizado, esto se lleva a cabo en la interfaz electrónica como se ve más adelante.

En este proyecto el ciclo de trabajo se ha codificado con 12 bits, lo que conlleva que se pueden codificar de 0 a 4095 valores diferentes, siendo el 0 la señal continua a nivel bajo, el 4095 la señal continua a nivel alto, y los valores intermedios diferentes anchuras de pulso de la señal cuadrada. La frecuencia empleada es de 19500 Hz, siendo esta la máxima permitida por el microcontrolador para 12 bits. Se ha seleccionado la mayor frecuencia posible para facilitar el posterior filtrado de la señal como se verá en el apartado correspondiente de la interfaz electrónica.

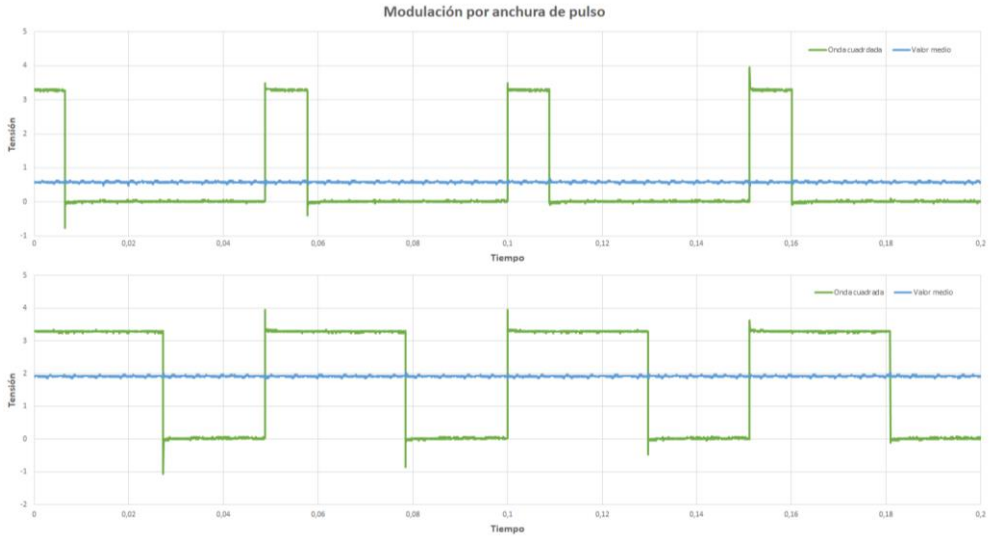

Ilustración 13. Modulación PWM. Fuente propia. Mediciones realizadas en el laboratorio, indicando la variación en el valor medio de tensión ante cambios en el ciclo de trabajo, con la tensión medida en voltios y el tiempo en segundos.

7.3.5 Puerto micro-USB

La placa de desarrollo lleva incorporado este puerto para la conexión del PC del usuario con el microcontrolador. A través de él se carga el programa en la memoria y permite la comunicación serie durante la ejecución. Se emplea un cable micro-USB a USB para conectarlo por un extremo al ordenador y por otro al controlador. Dentro del código del programa se emplea bastante esta funcionalidad de envío de información mediante un protocolo serie, no para el objetivo principal de control del servomotor pero sí para que el usuario pueda recibir información en tiempo real del funcionamiento del microcontrolador si conecta su PC, o incluso para modificar el código si lo requiere.

7.3.6 Antena integrada

Es una de las partes más importantes del microcontrolador pues es la que permite la comunicación mediante Wifi con los demás dispositivos sin necesidad de periféricos y controladores externos. Está diseñada para trabajar en la frecuencia habitual de los router de 2,4GHz, mediante el estándar HT40 que permite canales de transmisión de 40MHz y con una velocidad de transferencia máxima de 150Mbits por segundo, la cual es ampliamente suficiente para los requerimientos del proyecto. Su funcionamiento está controlado por las funciones de librería tanto externas como internas permitiendo la realización de tareas con múltiples subprocesos a bajo nivel empleando pocas líneas de código.

7.4 Librerías empleadas

En informática, las librerías son conjuntos de código predefinido y funciones que se utilizan para facilitar el desarrollo de software. Contienen una colección de rutinas, subrutinas y clases que realizan tareas específicas y proporcionan funcionalidades listas para usar en programas. Algunas de las librerías están incluidas en el propio entorno de desarrollo de Arduino, así como muchas funciones propias, sin embargo, para proyectos de mayor envergadura es común recurrir al uso de librerías externas desarrolladas por terceros y que requieren su instalación en el entorno de desarrollo por parte del usuario.

Gracias a la comunidad de aficionados y usuarios del lenguaje de programación Arduino y del microcontrolador ESP32, hay en la red una gran cantidad de librerías externas de funciones que otorgan gran cantidad de herramientas de elevado nivel para llevar a cabo cualquier tipo de proyecto sin tener que escribir el código completo. Son de código abierto y en su creación suelen participar varias personas. Estas librerías han sido descargadas de GitHub, un sitio web muy conocido en informática utilizado para compartir código y la creación de proyectos conjuntos.

7.4.1 Wifi.h

Se trata de una librería propia del entorno de desarrollo de Arduino, su principal función es proveer de herramientas que faciliten el uso del Wifi. En el código del proyecto se ha empleado para realizar la conexión a la red wifi deseada y para el establecimiento de una dirección IP estática.

7.4.2 Ticker.h

Librería propia del entorno de desarrollo, cuyo principal uso es el control de temporizadores avanzados y la ejecución temporizada de funciones. Su uso en el trabajo es el control temporizado de la función encargada de enviar los datos al dispositivo del usuario, haciendo que este tenga un refresco de información cada décima de segundo. Se relaciona fuertemente con la característica multitarea, pues es independiente el cálculo de tiempo y la ejecución de esta función de las demás partes del programa.

7.4.3 ESPAsyncWebServer.h

Librería externa desarrollada en el repositorio del usuario de GitHub “me-no-dev”. [19]

Su función principal es la creación de un servidor web asíncrono compatible con ESP32, capaz de establecer los protocolos de comunicación necesarios y llevar a cabo el funcionamiento del servidor. Se ha empleado principalmente para la definición del puerto del servidor, la creación e inicio de este y finalmente para el envío mediante el protocolo HTTP del código HTML de la página web ante el requerimiento de conexión por parte del usuario.

7.4.4 WebSocketsServer.h

Librería externa desarrollada en el repositorio de GitHub del usuario “Links2004”. [20]

Creada para permitir el uso de Websocket en proyectos de Arduino, con control de su protocolo y sus eventos. En el proyecto se emplea para la definición del puerto de Websocket, inicializar el protocolo, el servidor, codificar la respuesta a eventos dentro de la función que incluye la librería y permitir el envío de datos al usuario mediante Websocket.

7.4.5 ArduinoJson.h

Librería externa desarrollada en el repositorio de GitHub del usuario “bblanchon”. [21]

Creada para permitir el uso del formato JSON en el lenguaje de Arduino. En este trabajo se emplea con el objetivo de convertir vectores de datos a formato JSON, para así poder comunicarse con el usuario, así como decodificar los datos recibidos en este mismo formato a variables entendidas por Arduino. Para ello se emplean funciones, variables y clases incluidas en dicha librería.

8. INTERFAZ ELECTRÓNICA

8.1 Introducción

La interfaz electrónica es un dispositivo, más concretamente una placa base, cuyo principal objetivo es adaptar las señales que se envían y reciben el microcontrolador y el servopack a sus valores respectivos de operación. Sus circuitos deben filtrar las señales analógicas enviadas por el ESP32, reducir la tensión de las señales enviadas por el servopack y elevarla en el caso de las enviadas por el microcontrolador. Sumado a esto debe disponer de etapas de adaptación de la tensión de red para la alimentación de sus componentes, así como de un zócalo para el microcontrolador y un puerto de entrada/salida para el servopack.

En este capítulo se pretende mostrar de forma detallada el diseño de los diferentes circuitos implicados, los componentes que los forman y el proceso de construcción del prototipo.

8.2 Diseño de filtros para las señales analógicas

Como se ha comentado anteriormente, las señales analógicas generadas a través de PWM requieren su posterior filtrado mediante filtros paso bajo para la obtención de un valor de continua coincidente con su valor medio y que el servopack pueda comprender. Para dicho propósito se han diseñado unos filtros basados en amplificadores operacionales que cumplen con dicho objetivo.

8.2.1 Objetivos deseados

Se toma como punto de partida una señal (PWM), cuadrada de amplitud variable a una frecuencia de 19500 Hz y con unos valores máximos y mínimos de 3,3V y 0V respectivamente. Esta señal se debe filtrar antes de su envío al servopack para que reciba una señal de tensión continua como indican sus recomendaciones de uso.

Los valores de tensión que deben llegar al servopack deben estar incluidos entre [-6,6 ; 6,6]V en el caso de la señal de velocidad y [-3,3 ; 0]V para la entrada de límite de par. Esto es debido a los parámetros establecidos de fábrica en el servomotor, los cuales indican que los valores nominales de velocidad y par se alcanzan en 6V y -6V en cada sentido, y en -3V para el límite de par, siendo lineal en todo momento la correlación. No obstante, se permite superar el valor nominal en ambas señales, pero su uso se debe limitar a situaciones extraordinarias y no es recomendable. Se expone que, en el caso de la velocidad, la señal además de filtrada debe ser amplificada y se le debe dotar de una tensión de desbalance artificial que permita la generación de tensiones tanto positivas como negativas.

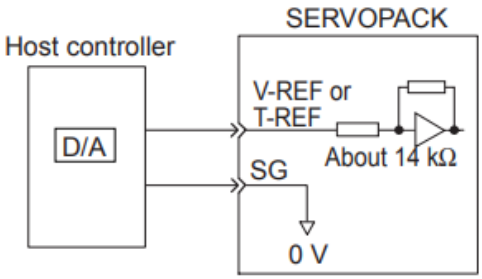

Ilustración 14. Esquema de entrada analógica al servopack. Dibujo obtenido del Manual de usuario del servopack y servomotor. Esquema de la electrónica interna de recepción de señales analógicas, e indicación de su impedancia de entrada.

El servopack cuenta con un filtrado interno que elimina parte del ruido generado y ayuda al filtrado. Como objetivo de diseño se exige una atenuación del 1‰ del primer armónico de la señal en su forma de onda cuadrada perfecta.

Un problema inherente de la electrónica analógica es el ruido y los desbalances internos, por lo que se debe construir un sistema capaz de realizar un ajuste a la tensión de salida cuando esta no se corresponda con la deseada. Este fenómeno se llama tensión de desbalance o de "offset" y debe ser eliminado en el caso del par y ajustado de forma adecuada en el caso de la velocidad para obtener la tensión de 0V de salida ante la señal de entrada correspondiente a 0rpm o 0Nm. Para ello ha instalado un conjunto de resistencias y potenciómetros para ajuste manual del offset en caso de desajustes.

8.2.2 Fundamentos teóricos empleados

En los filtros la variable fundamental es la frecuencia, pues ella determina cual será el valor de la atenuación de la señal de entrada. En este proyecto se persigue como señal de salida una tensión casi continua con un pequeño rizado, en el valor de tensión que corresponde al valor medio de la señal de entrada.

Según los desarrollos de Fourier, toda señal periódica de forma arbitraria se puede descomponer en una suma de señales de tipo sinusoidal de una amplitud y frecuencia determinados. Este hecho es fundamental para la teoría de los filtros, pues permite considerar cada componente en frecuencia de una señal de forma individual. Las señales PWM tienen una mayor cantidad de armónicos cuando su ciclo de trabajo se encuentra en el 50%, es decir cuando está el mismo tiempo a alto valor que a bajo. Por ello mismo se realiza todo el diseño de filtrado como si en todo momento se estuviera dando este tipo de señal más desfavorable, llamada señal cuadrada.

Se sabe que la onda cuadrada tiene su primer armónico en su frecuencia fundamental, en este caso 19500Hz, por lo que esta será la frecuencia estudiada. Además, solo tiene armónicos impares, situados en los múltiplos impares de esta frecuencia indicada, el tercero en 58500Hz, por ejemplo. Es esta señal senoidal a 19500Hz la que se pretende que tenga una atenuación del 1‰ con respecto a la tensión senoidal de entrada, pues lo que quede a la salida será ruido en la señal. Los armónicos a medida que se alejan de 0Hz en este tipo de señales disminuyen su amplitud, por lo que normalmente el primero de ellos suele ser el objetivo de diseño, mientras que los demás suponen un problema menor, además por el funcionamiento de los filtros será mayor la atenuación producida por lo que se desprecian.

Como se pretende eliminar las altas frecuencias el filtro a usar se denomina paso bajo, pues permite el paso de las frecuencias bajas y atenúa las señales de alta frecuencia. Como se pretende obtener a la salida un valor de tensión continua correspondiente al valor medio, este valor medio se considera un componente de continua en la señal y se trata como si fuera un armónico de frecuencia 0Hz, la más baja posible.

Por otro lado, este es un estudio exclusivamente teórico, así que no se han considerado tensiones inducidas o interferencias externas como puede ser la interferencia generada por la red eléctrica, pues el propio servopack está preparado para eliminarlas.

8.2.3 Selección del tipo de filtro base

En la etapa de diseño se han barajado dos posibles disposiciones de filtro paso bajo. Conocidos como filtro inversor y filtro no inversor. Cuyos esquemas se muestran a continuación.

Las funciones de transferencia en frecuencia se emplean para calcular la relación en amplitud entre la tensión de salida y la de entrada provocada por un filtro ante una entrada de una frecuencia determinada, a frecuencia 0 no hay actuación del filtro y se puede observar su efecto sobre la componente de continua (El valor medio). Las de las dos opciones estudiadas son las siguientes:

Filtro inversor:

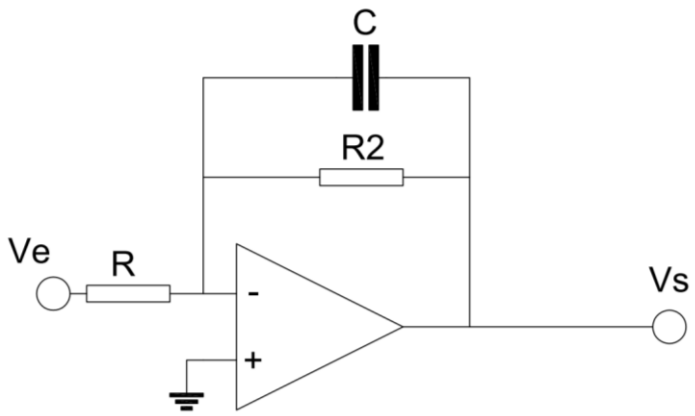

Ilustración 15. Esquema de filtro inversor. Fuente propia.

$$\left|\frac{V_s}{V_e}\right| = \frac{1}{\sqrt{\left(\frac{R}{R_2}\right)^2 + (RC2\pi f)^2}} \qquad Para\ f = 0\text{: } \left|\frac{V_s}{V_e}\right| = \frac{R_2}{R} \tag{1}$$

Filtro no inversor:

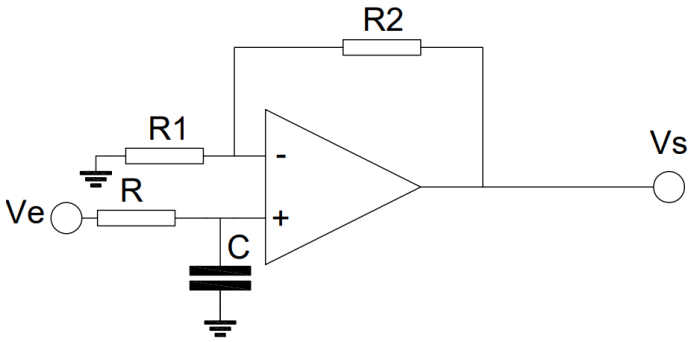

Ilustración 16. Esquema de filtro no inversor. Fuente propia.

$$\left|\frac{V_s}{V_e}\right| = \frac{1+\frac{R_2}{R_1}}{\sqrt{1+(RC2\pi f)^2}} \qquad Para\ f = 0\text{: } \left|\frac{V_s}{V_e}\right| = 1+\frac{R_2}{R_1} \tag{2}$$

Donde R es el valor de cada resistencia en Ohmios (Ω), C el valor de la capacidad en Faradios (F) y f la frecuencia en Hercios (Hz). Estas funciones de transferencia se han obtenido de aplicar el teorema de Millman entre la entrada positiva y negativa del amplificador operacional, considerado ideal, y posteriormente calculando su módulo considerando señal senoidal.

A la vista de los esquemas se puede apreciar que la disposición no inversora requiere de una resistencia más que la disposición inversora, aumentando así su coste y su complejidad de montaje. Sin embargo, su mayor desventaja se muestra a continuación.

Para el caso de la velocidad que requiere de una amplificación del valor medio (f=0Hz) de valor 4 y una atenuación del primer armónico (f=19500Hz) del 0,001 se requiere en función de la disposición de los siguientes componentes similares a los finalmente usados:

Para filtro inversor: $R_2 = 4K\Omega; R = 1K\Omega$; Se requeriría un valor de capacidad de 8µF aproximadamente.

Para filtro no inversor: $R_2 = 3K\Omega; R = 1K\Omega;\ R_1 = 1K\Omega$; Se requiere un valor de capacidad de 30µF aproximadamente.

Para el caso del par que requiere de una amplificación del valor medio (f=0Hz) de valor 1, y una atenuación igual del primer armónico se obtiene:

Para filtro inversor: $R_2 = 4{,}5K\Omega; R = 4{,}5K\Omega$; Se requeriría un valor de capacidad de 2µF aproximadamente.

Para filtro no inversor: $R_2 = 0\Omega; R = 4.5K\Omega;\ R_1 = 4{,}5K\Omega$; Se requiere un valor de capacidad de 2µF aproximadamente.

Ante estos cálculos realizados durante la selección de dichos filtros se observó que el filtro inversor, si se requiere de una amplificación del valor medio, requiere de un condensador de bastante menor capacidad que el filtro no inversor para obtener la atenuación deseada del primer armónico, pero para una relación unitaria ambos requieren el mismo condensador.

Finalmente teniendo en cuenta estas dos desventajas del no inversor y con el objetivo de unificar el diseño tanto para par como para velocidad se decide emplear la disposición inversora.

8.2.4 Ajuste de la tensión de desbalance en el caso de la velocidad

En el caso de la velocidad el ajuste del offset tiene dos motivaciones, por un lado, el ajuste de las posibles desviaciones generadas por el amplificador operacional, y por otro lado permitir la generación de una tensión de salida que pueda tener valores positivos y negativos, es decir, desplazar el rango de salida. Para la realización de los dos objetivos se conecta una resistencia entre la entrada negativa y la entrada de alimentación negativa del amplificador operacional. Esta resistencia deberá ser variable para poder realizar el ajuste manual para poder regular la tensión de desbalance no deseada.

El esquema electrónico es el siguiente:

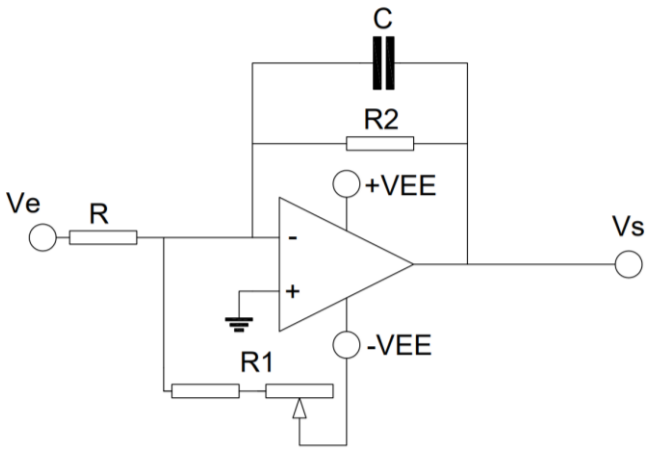

Ilustración 17. Ajuste de offset para el control de la velocidad. Fuente propia. Esquema de la electrónica capaz de eliminar la tensión de desbalance a la salida de forma manual.

Para la obtención de la ecuación que regula el comportamiento del offset se ha realizado la siguiente simplificación. Se va a realizar el cálculo como si la tensión de entrada fuera el valor medio en continua de la señal PWM que realmente va a entrar, esto se puede asumir, pues lo que se pretende es desplazar dicho valor medio que es el que lleva la información de la tensión de salida que se desea en el filtro. Al realizar dicha simplificación se ignoran los armónicos, así como la respuesta en frecuencia, lo que permite simplificar sobremanera la expresión y sus cálculos.

Adicionalmente esta simplificación no incluye casi error, pues este desplazamiento no afecta a la amplitud ni a la atenuación de los armónicos que aparecen a la salida del filtro, pues estos van a ser filtrados de la misma forma que en el apartado anterior, lo único que ocurrirá es el desplazamiento en tensión de estas señales senoidales que forman los armónicos, pero a la vista de las funciones de transferencia este hecho no influye en absoluto.

Finalmente, la expresión que gobierna el comportamiento para el valor medio de tensión es:

$$V_s = \frac{R_2}{R_1} V_{EE} - \frac{R_2}{R} V_e \tag{3}$$

Donde V_e es el valor medio de la señal de entrada, V_{EE} es la tensión de alimentación en valor absoluto, V_s la tensión de salida y las diferentes R el valor en Ohmios de cada resistencia.

Esta expresión se ha obtenido de aplicar el teorema de Millman entre la entrada positiva y negativa del amplificador operacional considerando solo tensión continua, por lo que el condensador genera un circuito abierto y deja de influir en los cálculos.

8.2.5 Cálculos realizados para el filtrado y amplificación de la velocidad

En este apartado se pretende obtener los valores requeridos, o las relaciones entre ellos, de los diferentes componentes pasivos que componen el filtro, es decir de las resistencias y del condensador. En primer lugar, se exponen los requerimientos a cumplir.

Atenuación del 0,001 para el primer armónico en disposición de onda cuadrada.

Partiendo de unos valores de tensión media de entrada en un rango de [0 ; 3,3]V se requiere una tensión de salida de rangos [-6,6 ; 6,6]V. De tal forma que ante una entrada de 0V a la salida se obtengan 6,6V, a la entrada de 1,65V se obtengan 0V, y a la entrada de 3,3V se obtengan -6,6V, pues no se debe olvidar que la disposición inversora invierte la polaridad de la tensión.

Para el primer requisito se empleará la función de transferencia del filtro inversor, en su disposición como si no hubiera ajuste de offset, por lo explicado en el apartado anterior. Para el segundo requisito se hará uso de la expresión obtenida en el apartado anterior, pues va a permitir conocer las relaciones entre resistencias para lograr el rango de salida deseado.

8.2.5.1 Cálculo de resistencias para la obtención del rango deseado

De los tres puntos de control que se obtienen del segundo requerimiento, empleando la expresión presentada en el apartado de ajuste del offset, se puede realizar el siguiente sistema de ecuaciones.

$$\begin{cases} 6{,}6 = \dfrac{R_2}{R_1}V_{EE} - \dfrac{R_2}{R} * 0 \\ 0 = \dfrac{R_2}{R_1}V_{EE} - \dfrac{R_2}{R} * 1{,}65 \\ -6{,}6 = \dfrac{R_2}{R_1}V_{EE} - \dfrac{R_2}{R} * 3{,}3 \end{cases} \tag{4}$$

Como es de esperar se trata de un sistema compatible indeterminado, pues nos ofrece una relación de resistencias, siendo estas:

$$\frac{R_2}{R_1} = \frac{6{,}6}{V_{EE}} \quad ; \quad \frac{R_1}{R} = \frac{2 * V_{EE}}{3{,}3} \quad ; \quad \frac{R_2}{R} = 4 \tag{5}$$

Donde V_{EE} es la tensión de alimentación del AO (Amplificador operacional) en valor absoluto. Es destacable la última relación, pues además de ser muy útil para el siguiente apartado, coincide con lo que se obtendría de la función de transferencia del filtro sin offset a una frecuencia de 0 Hz en el caso de que se quisiera amplificar el rango de la señal cuatro veces, como se realiza en este caso. Lo que indica que la simplificación está bien fundamentada.

8.2.5.2 Cálculo de la capacidad necesaria para el filtrado

Para este caso se hará uso exclusivamente de la función de transferencia del filtro inversor base y de la relación de amplificación obtenida en el apartado anterior $\frac{R_2}{R} = 4$, quedando finalmente esta expresión:

$$\left|\frac{V_s}{V_e}\right| = \frac{1}{\sqrt{\left(\frac{1}{4}\right)^2 + (RC2\pi f)^2}} \tag{6}$$

De esta, para una frecuencia de 19500Hz y una atenuación de esta frecuencia de $\left|\frac{V_s}{V_e}\right| = 0{,}001$ va a obtenerse el valor de la relación RC. Siendo $RC = 0{,}00816$, para la resistencia en Ohmios y la capacidad en Faradios.

8.2.5.3 Valores de los elementos pasivos finalmente utilizados

En este punto ya se deben tomar ciertas decisiones de diseño más relacionadas con la utilidad y la facilidad del montaje y más alejadas de la teoría, debido en parte a la necesidad de adecuarse a los valores comerciales de los componentes.

En primera instancia la alimentación del amplificador operacional será de ±15V, es decir tendrá un valor de 15V en su entrada de alimentación positiva y de -15V en la negativa, siendo su valor para V_{EE} en las fórmulas de 15V.

$$\frac{R_2}{R_1} = \frac{6{,}6}{15} \;\; ; \;\; \frac{R_1}{R} = \frac{2 * 15}{3{,}3} \;\; ; \;\; \frac{R_2}{R} = 4 \;\; ; \; RC = 0{,}00816$$

Finalmente empleando los valores de resistencias de que dispone el laboratorio de máquinas eléctricas y comprando el condensador se obtiene los siguientes resultados, que han sido los definitivos para el montaje.

$$R = 1K\Omega; \; R_2 = 4K\Omega \; ; \; R_1 = 9K\Omega \; (variable) \; ; C = 10\mu\text{F}$$

R_2 se construirá a base de 4 resistencias de 1 KΩ en serie, y R_1 con una resistencia de 4,7 $K\Omega$ en serie con una resistencia variable de 10 KΩ de recorrido total, para poder regular la tensión de desbalance perjudicial de forma sencilla.

Con estos valores los requerimientos indicados tendrían los siguientes valores teóricos:

Atenuación del primer armónico de 0,0008161.

Ante entrada de 0V de valor medio una salida de 6,6006V, ante entrada de 1,65V una salida de 0,0006V y ante una entrada de 3,3V una salida de -6,5993V.

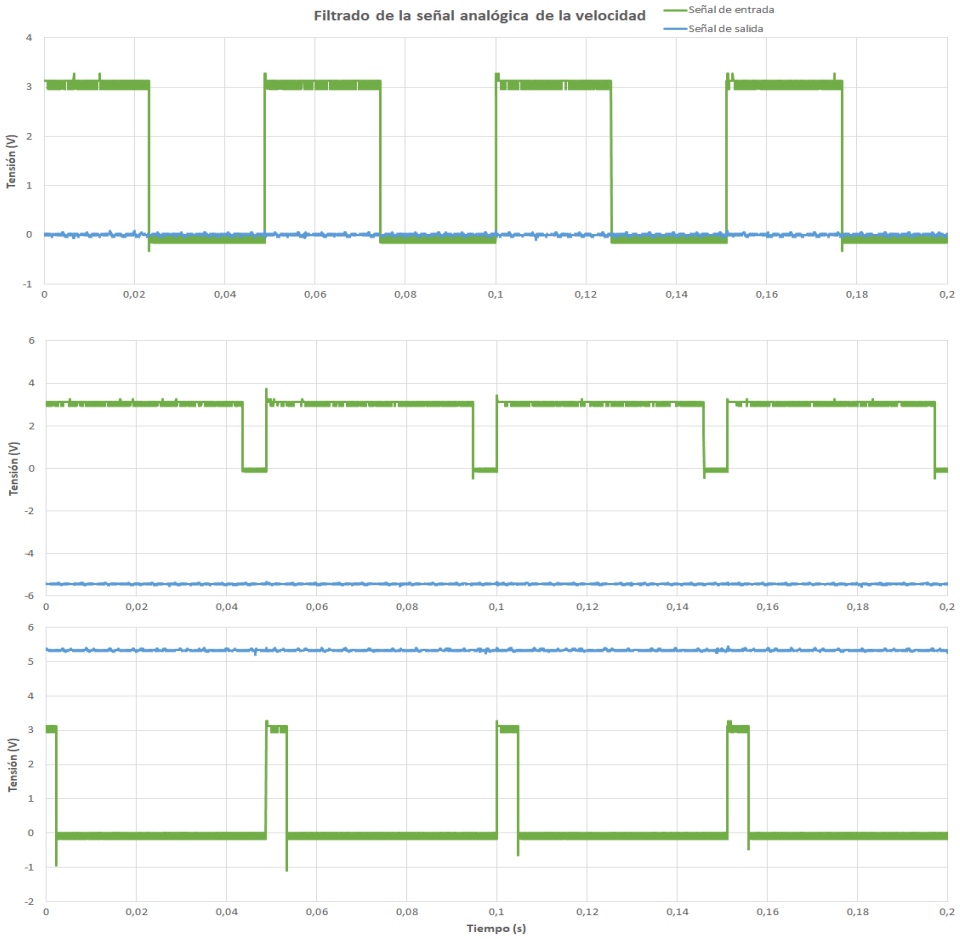

Ilustración 18. Filtrado de la señal analógica de control de la velocidad. Fuente propia. Mediciones efectuadas en el laboratorio con tres señales de entrada cuadradas de diferente ciclo de trabajo.

Paralelamente, la capacidad de ajuste del offset mediante la regulación de la resistencia variable puede evaluarse mediante cambios en los valores de R_1empleando las mismas expresiones pero obteniendo las tensiones de salida. Se hace notar que siempre que no se varíen las otras propiedades, entre el valor de salida centrado y las tensiones límite habrá una diferencia positiva o negativa de 6,6V. Por lo que se evalúa para su valor centrado con una tensión de entrada de 1,65V.

$$R_1 = 4{,}7\ K\Omega + 0(variable);\ V_{s,centrado} = 6{,}166V$$

$$R_1 = 4{,}7\ K\Omega + 10K\Omega(variable);\ V_{s,centrado} = -2{,}518V$$

Siendo este su rango de regulación para valores centrados.

8.2.5.4 Cálculo de la precisión en velocidad y de la intensidad de entrada

El parámetro del servopack que regula la relación entre la tensión de entrada y la velocidad generada establece una relación de 0,002V/rpm. También se sabe que al codificar la señal PWM con 12 bits se tienen 4096 valores para codificar la tensión de entrada, cuyo rango de valores es de 0 a 3,3V, lo que indica que la relación entre valor de tensión y ciclo de trabajo decimal es de 0,0008056V/división. Sin embargo, al aumentar dicho rango por 4 para adecuarse a los rangos de salida, esta precisión se ve multiplicada el mismo factor, quedando de 0,003222V/división.

Finalmente relacionando el valor del servopack con el de la tensión de salida se obtiene la siguiente precisión: 1,611rpm/división. Siendo la división los valores que puede indicar el usuario a la modulación PWM que regula la velocidad.

El cálculo de la corriente máxima que debe suministrar el microcontrolador se realiza para el caso en el que la señal de entrada adquiere su valor máximo, siendo este de 3,3V, y aplicando la Ley de Ohm entre los puntos de tensión de entrada y la entrada negativa del amplificador operacional, que al ser este ideal y tener su entrada positiva conectada a masa, tendrá potencial 0V. Para este caso la señal será una tensión continua, pues será una PWM siempre a valor alto, así que el cálculo se realiza como:

$$i_e = \frac{V_e - V^-}{R} = \frac{3{,}3 - 0}{1000} = 3{,}3mA \tag{7}$$

Siendo muy inferior a los 40mA, la cual es la corriente máxima absoluta con la que pueden trabajar los GPIOs del ESP32 como indica la sección “Condiciones de uso recomendadas” en su hoja de características.

8.2.6 Ajuste de la tensión de desbalance en el caso del límite de par

En este caso la tensión de desbalance es perjudicial, pues no queremos desplazar el intervalo de salida. Para su eliminación se ha empleado una disposición muy habitual para dicho propósito, consistente en una agrupación de 4 resistencias y una resistencia variable para que pueda ser ajustada manualmente.

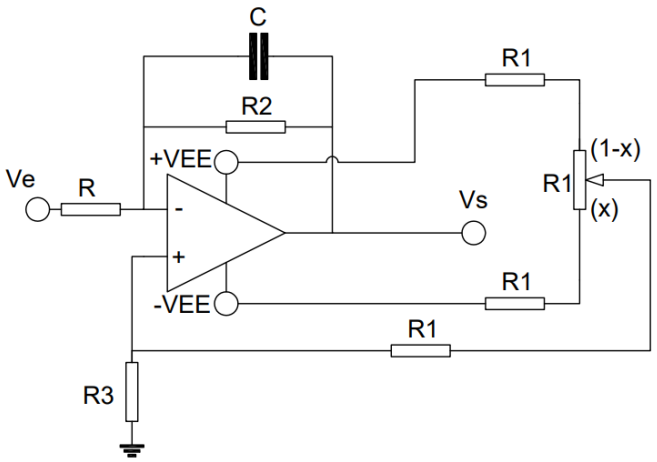

Ilustración 19. Ajuste de offset para el límite de par. Fuente propia. Esquema de la electrónica capaz de eliminar la tensión de desbalance a la salida de forma manual.

El funcionamiento de esta disposición es más intuitivo que la del caso de la velocidad, pues en este caso se influye sobre la entrada positiva del AO, la cual tiene una relación menos directa con la señal de entrada. A grandes rasgos, lo que hace es regular la tensión que aparece en la entrada positiva, y que por lo tanto aparecerá también en la negativa al ser considerado ideal, lo que provoca un cierto desbalance a la entrada que compense el “offset” por defecto.

La expresión que muestra la tensión en la entrada positiva es la siguiente:

$$V^+ = \frac{2x-1}{\left(\frac{R_1}{R_3}\right)(-x^2+x+5)+3} V_{EE} = f(x)V_{EE} \tag{8}$$

Donde x es un valor en el rango de [0 , 1] que indica la regulación de la resistencia variable, para x=0 toda la resistencia del potenciómetro estará en la rama de alimentación positiva, y para x=1 en la negativa, en el caso x=0,5 tendrá la misma en ambas. V_{EE} es el valor de la tensión de alimentación en valor absoluto para ambas alimentaciones y V^+el valor de la tensión en la entrada positiva del amplificador.

En esta disposición se recomienda que R_1 sea 10 veces superior a R_3, pues esta tensión no debe tomar valores muy elevados, solamente el necesario para la eliminación del offset. Se destaca que esta tensión puede tomar valores positivos o negativos en función de la x, esto es debido a que la tensión de desbalance puede tener cualquier polaridad y V^+ se debe oponer a ella.

Finalmente obteniendo la tensión de salida V_s en función de V^+ y V_e:

$$V_s = -\frac{R_2}{R}V_e + f(x)V_{EE}\left(\frac{R_2}{R}+1\right) = -\frac{R_2}{R}V_e + V_{Ajuste} \tag{9}$$

Siendo el segundo sumando la tensión a la salida que deberá encargarse de compensar la tensión de desbalance, y demostrando así que depende de la posición adoptada por el potenciómetro.

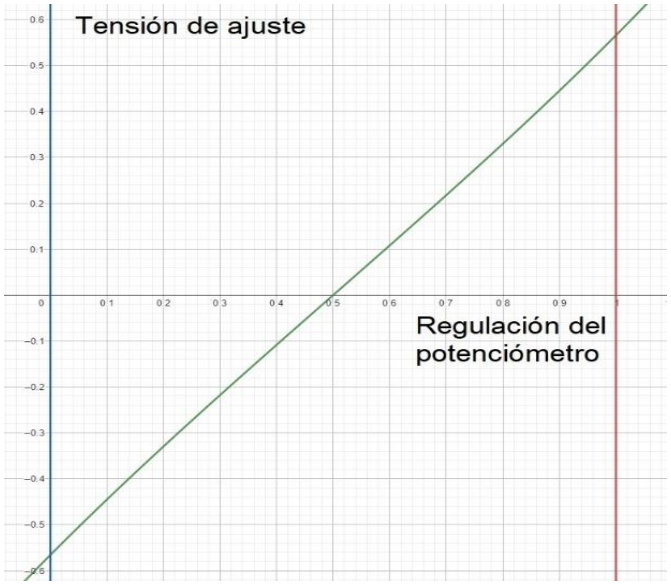

Ilustración 20. Representación de V_{Ajuste} en función de la regulación x. Fuente propia. Mapeado de la tensión de ajuste en voltios para una tensión de alimentación de 15V y una relación $R_1 = R_3 * 10$; $R_2 = R$ en resistencias.

8.2.7 Cálculos realizados para el filtrado y amplificación del límite de par

En este apartado se pretende obtener los valores requeridos, o las relaciones entre ellos, de los diferentes componentes pasivos que componen el filtro, es decir de las resistencias y del condensador. En primer lugar, se exponen los requerimientos a cumplir.

Atenuación del 0,001 para el primer armónico en disposición de onda cuadrada.

Partiendo de unos valores de tensión media de entrada en un rango de [0 ; 3,3]V se requiere una tensión de salida de rangos [-3,3 ; 0]V, pues la disposición inversora invierte la polaridad de la salida. Lo que indica que no se requiere de amplificación, si no exclusivamente de un filtrado.

En este caso se realizarán todos los cálculos empleando exclusivamente la función de transferencia del filtro inversor, para su valor medio a f=0Hz y para su primer armónico a f=19500Hz. Es decir que no se tendrá en cuenta el ajuste de offset, esto es debido a que los cálculos que se van a realizar a continuación son puramente teóricos, mientras que los realizados para el ajuste de offset tienen un objetivo experimental. Por otro lado, la tensión de ajuste debe tener un valor reducido, que además debe eliminar el "offset" perjudicial haciendo que la realidad se acerque más a los cálculos ideales sin influir mucho en ellos. No obstante, se eligen también en estos apartados los valores de las resistencias de regulación de "offset".

Por otro lado, la tensión de alimentación del amplificador operacional será de 15V y -15V de igual forma que en el caso de la velocidad.

8.2.7.1 Cálculo de resistencias para la obtención del rango deseado

A la vista de los requerimientos, y empleando la función de transferencia para el valor medio de tensión se tiene.

$$\left|\frac{V_s}{V_e}\right| = \frac{R_2}{R} \tag{10}$$

Esta relación debe ser igual a 1, pues no se debe amplificar ni atenuar la tensión de salida con respecto a la de entrada, si no que la debe mantener. Por lo tanto, se requiere que $R_2 = R$.

8.2.7.2 Cálculo de la capacidad necesaria para el filtrado

Empleando la relación anteriormente obtenida, y la función de transferencia del filtro para conseguir una atenuación de 0,001 en los 19500Hz se tiene:

$$0{,}001 \ = \frac{1}{\sqrt{(1)^2 + (RC2\pi 19500)^2}} \tag{11}$$

Despejando la relación RC se obtiene finalmente: RC=0,00816178, estando la resistencia en Ohmios y la capacidad en Faradios.

8.2.7.3 Valores de los elementos pasivos finalmente utilizados

Empleando los valores que mejor se ajustan a los objetivos deseados y siguiendo las siguientes relaciones:

$$R_2 = R;\ RC = 0{,}00816178;\ Para\ la\ regulación\ del\ offset:\ R_1 = 10R_3(recomendado)$$

Se obtiene los valores de diseño del filtro:

$R_2 = R = 4{,}7K\Omega$; $C = 2{,}2\mu F$;

$R_1 = 10K\Omega\ (También\ para\ la\ resistencia\ variable)$; $R_3 = 1\ K\Omega$

Obteniéndose los siguientes resultados:

Una regulación de offset entre los [-0,566 ; 0,566]V de capacidad de regulación a la salida.

Una atenuación de 0,0007893 para el primer armónico, y una ganancia unitaria para el valor medio.

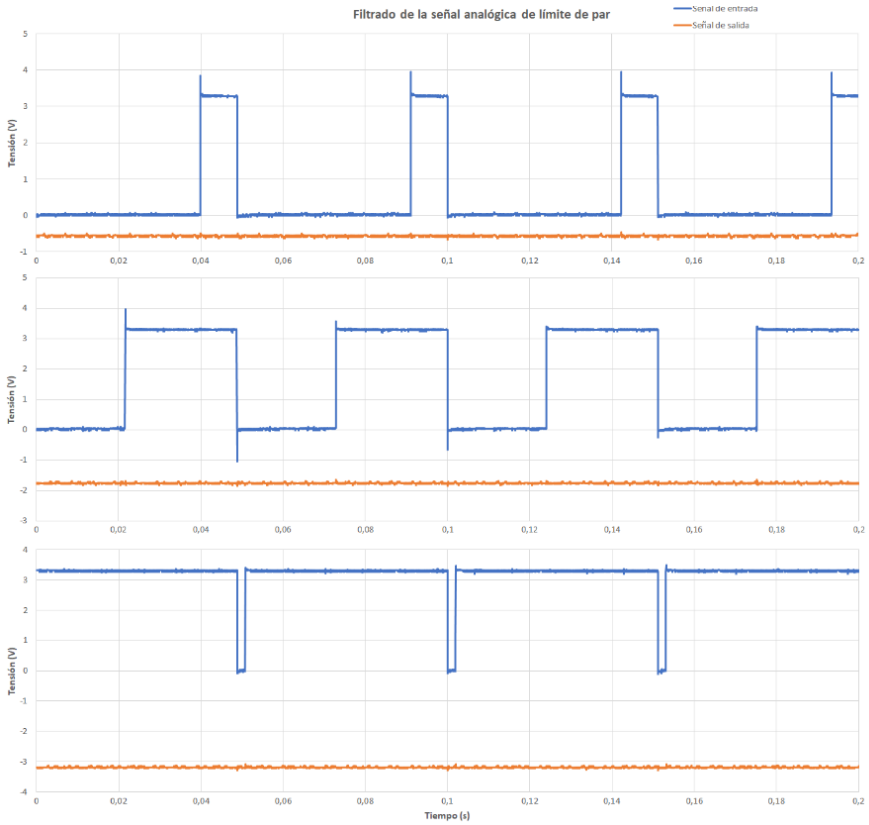

Ilustración 21. Filtrado de la señal analógica de límite de par. Fuente propia. Mediciones efectuadas en el laboratorio con tres señales de entrada cuadradas de diferente ciclo de trabajo.

8.2.7.4 Cálculo de la precisión en límite de par y de la intensidad de entrada

El parámetro del servopack que regula la relación entre la tensión de entrada y el límite de par establece una relación de 4,7095V/Nm, esto es así debido a que el par nominal no llega a la unidad, quedándose en 0,67Nm. También se sabe que al codificar la señal PWM con 12 bits se tienen 4096 valores para codificar la tensión de entrada, cuyo rango de valores es de 0 a 3,3V, lo que indica que la relación entre valor de tensión y ciclo de trabajo decimal es de 0,0008056V/división. En este caso al no haber amplificación dicha precisión en tensión se mantiene a la salida del filtro.

Finalmente relacionando el valor del servopack con el de la tensión de salida se obtiene la siguiente precisión: 0,0001710693Nm/división. Siendo la división los valores que puede indicar el usuario a la modulación PWM que regula la velocidad.

El cálculo de la corriente máxima se realiza para el caso en el que la señal de entrada adquiere su valor máximo siendo este de 3,3V y aplicando la Ley de Ohm entre los puntos de tensión de entrada y la entrada negativa del amplificador operacional. En este cálculo se ha despreciado el valor de la tensión de ajuste de offset por ser muy reducido en comparación con la tensión de entrada.

$$i_e = \frac{V_e - V^-}{R} = \frac{3{,}3 - 0}{4700} = 0{,}702mA \qquad (12)$$

Siendo muy inferior a los 40mA, la cual es la corriente máxima absoluta con la que pueden trabajar los GPIOs del ESP32 como indica la sección “Condiciones de uso recomendadas” en su hoja de características.

8.2.8 Modelo empleado de amplificador operacional

Para la realización de ambos filtros se va a requerir del uso de un único circuito integrado, el cual contiene dos amplificadores operacionales. Su modelo es TL082 CP, fabricado por Texas Instruments. Su patillaje es el siguiente:

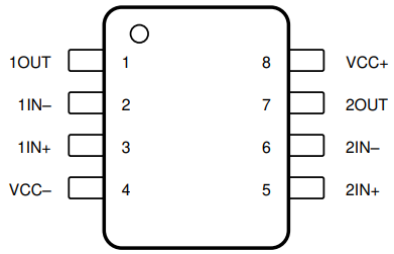

Ilustración 22. Esquema TL082. Obtenido de su ficha técnica. Distribución de pines del circuito integrado que contiene los dos amplificadores operacionales. Siendo VCC las tensiones de alimentación, IN las entradas y OUT las respectivas salidas.

Disponiendo de 4 pines para las entradas (2 para cada AO), 2 pines de salida (1 por AO), un pin para la entrada de alimentación positiva y el último para la alimentación negativa.

A la vista de sus especificaciones, los valores máximos permitidos son los siguientes:

Tensión de alimentación máxima: +18V y -18V, en el proyecto se emplean +15V y -15V, además las empleadas aparecen dentro de los valores recomendados por el fabricante.

Tensión de entrada máxima, +15V y -15V, en el proyecto el valor máximo es 3,3V a la entrada, llegando a la salida a valores máximos de -6,6V y +6,6V.

Valores obtenidos de su ficha técnica. [22]

8.3 Diseño de divisores de tensión para las salidas digitales

Para adaptar los valores de tensión de las 4 salidas de control empleadas del servopack se requieren un conjunto de resistencias en disposición de divisor de tensión para lograr adaptar la señal a los 3,3V del ESP32 y que el microcontrolador la pueda interpretar de forma segura.

8.3.1 Objetivos deseados

Se toma como punto de partida el esquema eléctrico de la salida del servopack.

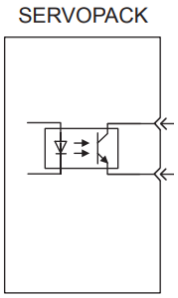

Ilustración 23. Esquema de salida digital del servopack. Obtenida del Manual de usuario del servopack y servomotor.

Se ve que se trata de un optoacoplador, un dispositivo de actuación similar al de un relé, que se emplea para desacoplar el circuito interno con el externo al no haber conexión eléctrica. Funciona como un interruptor interno, por lo que cuando la salida se activa cierra el circuito, y cuando no está activa lo abre. Esto se puede traducir a los valores de tensión buscados empleando una fuente de alimentación y resistencias, de tal forma que cuando se cierre el circuito se tenga a la entrada del ESP32 un valor de 3,3V, y cuando se abra de 0V.

Las 4 salidas empleadas (Velocidad de consigna alcanzada, Velocidad indicada superada, Servo preparado y Señal de alarma) funcionan todas de la misma forma, a excepción de la de alarma que es activa a nivel bajo, sin embargo, esto se invertirá por software en el código del microcontrolador. Por lo tanto las 4 emplearán el mismo circuito electrónico.

Según las especificaciones del servopack, entre los dos bornes del optoacoplador no pueden haber más de 30V de tensión, y por su interior pueden circular un máximo de 50mA de corriente. [2]

8.3.2 Disposición de los divisores de tensión.

El divisor de tensión es un elemento muy empleado en electrónica para rebajar tensiones en un determinado punto del circuito, empleando únicamente dos resistencias. La disposición para cumplir con los objetivos deseados debe ser la siguiente:

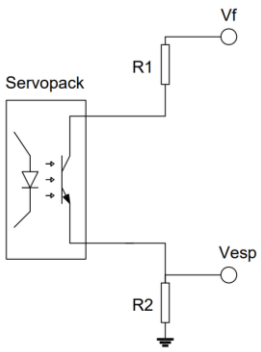

Ilustración 24. Divisor de tensión. Fuente propia. Esquema electrónico de la disposición de componentes a emplear para realizar el divisor de tensión y adaptar la señal.

De esta forma, cuando la salida esté activa, habrá un flujo de corriente entre la fuente de alimentación externa y la masa del sistema, debido a esta intensidad, habrá una caída de tensión en cada resistencia que debe ser calculada de tal modo que en el punto en el que se encuentra la entrada al microcontrolador aparezcan 3,3V. Por el contrario, cuando la señal esté desactivada, no habrá flujo de corriente y por lo tanto tampoco caída de potencial en las resistencias, lo que implica que el punto de entrada al ESP32 estará a la misma tensión que masa, 0V.

Para los cálculos en circuito cerrado se desprecia la corriente de entrada al microcontrolador, pues esta es muy inferior debido a la gran impedancia de entrada de los GPIO, sin embargo, para asegurar el menor error, no debe seleccionarse una resistencia de alto valor entre la entrada del microcontrolador y masa. Por otro lado, tampoco se tienen en cuenta las caídas de tensión dentro del servopack, al ser estas minúsculas en circuito cerrado.

8.3.3 Cálculos de las resistencias a emplear

La fórmula de los divisores de tensión es la siguiente:

$$V_{esp} = \frac{R_2}{R_2 + R_1} V_f \tag{13}$$

Para el cálculo de la corriente que discurrirá por el servopack se emplea la siguiente fórmula, obtenida de aplicar la Ley de Ohm.

$$I = \frac{V_f}{R_1 + R_2} \tag{14}$$

Donde V_{esp} es la tensión a la entrada del microcontrolador, V_f la tensión de la fuente de alimentación, R_2 y R_1 los valores en Ohmios de las resistencias e I el valor de la corriente en Amperios.

Por requisitos de diseño se tiene que V_{esp} ≈3,3V y que I≤0,05A.

Por otro lado, la fuente de tensión empleada es de 24V, lo cual hace imposible que se supere el límite de 30V entre los bornes del optoacoplador del servopack y además otros circuitos de la placa emplean este valor de tensión, así que se escoge el mismo para aprovechar la misma fuente. Por lo tanto $V_f = 24V$.

Con todas estas restricciones, finalmente se han escogido los siguientes valores:

$$R_2 = 2200\,\Omega;\;\; R_1 = 13600\Omega$$

R_1 se forma mediante dos resistencias de 6800 Ω en serie.

De este modo los valores objetivo son: V_{esp}=3,3417V; I=0,00152A.

Ajustándose a las especificaciones, pues las entradas del ESP32 soportan y detectan valores en estado ALTO hasta los 5V, y teniendo R_2 un valor lo suficientemente bajo comparado con los elevados valores de impedancia de entrada de los GPIO del ESP32.

8.4 Diseño de transistores de potencia para las entradas digitales

Para amplificar la tensión de las señales digitales que van a ser entrada del servopack se emplea una disposición muy típica de la circuitería de potencia, que es el empleo de transistores como amplificadores de esta señal, a un valor de tensión que el servopack pueda comprender. Esto se consigue empleando dichos transistores como interruptores que permitan el paso o no de corriente, y con una fuente de alimentación de tensión bastante superior a los 3,3V que es capaz de entregar el microcontrolador.

8.4.1 Objetivos deseados

Partiendo de las especificaciones requeridas para las entradas del servopack se tiene:

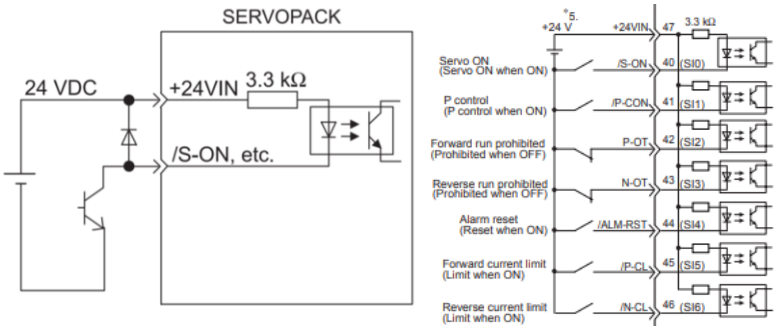

Ilustración 25. Entradas digitales del servopack. Obtenida del manual de usuario del servopack y servomotor. Esquema electrónico recomendado para las entradas, a la derecha el conjunto de entradas generales donde se muestra la alimentación común.

Las entradas están construidas dentro del servopack empleando optoacopladores, estos activarán la entrada cuando se cierre el circuito y se dé un flujo de corriente. Se exige una tensión de entrada de 24V, dicha entrada de tensión será común para todas las entradas, como se ve en la imagen global, para que junto con la resistencia interna de 3300 Ω se produzca un flujo de corriente capaz de activar al optoacoplador. Se pretende que, mediante un transistor, se regule la apertura y cierre del circuito de entrada al servopack, dicho transistor estará gobernado por la salida correspondiente del microcontrolador, cuyo valor de tensión es de 0V o 3,3V.

De todas las entradas mostradas, en el trabajo solamente se emplean tres: SERVO-ON, límite de par horario y límite de par antihorario. Todas ellas emplearán el mismo circuito y los mismos componentes.

En este apartado los valores seguros o de operación no aparecen debido a que el servopack ya exige unos determinados valores de funcionamiento, por lo que el microcontrolador solamente debe encargarse de la apertura o cierre del transistor.

8.4.2 Circuito electrónico empleado

Los transistores disponen de 3 estados de funcionamiento: comportamiento lineal, corte y saturación. Son estos dos últimos los que permiten su funcionamiento como un interruptor que

abre o cierra el circuito, controlando valores de tensión y corriente mucho mayores que las que controlan su actuación. En corte no permite el paso de corriente, y en saturación cierra el circuito oponiendo una caída de tensión muy baja. El esquema eléctrico es:

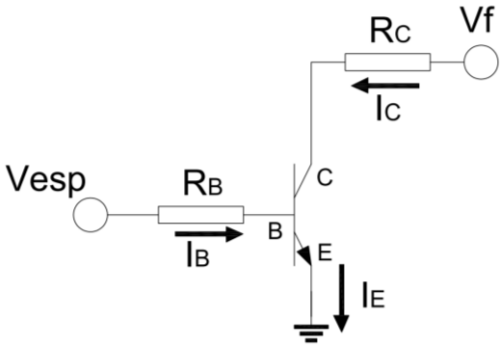

Ilustración 26. Esquema del transistor. Fuente propia. Esquema electrónico que indica la posición de los diferentes elementos que influyen en el funcionamiento del transistor. Las flechas indican el flujo de corriente eléctrica, la cual tiene gran importancia en los cálculos de transistores.

Donde I_B es la corriente de base, I_C la de colector e I_E la corriente de emisor. Siempre se cumple que I_E=$I_B + I_C$.

En la zona de funcionamiento lineal, la cual está en medio de corte y saturación se cumple que: $I_C = \beta * I_B$; Siendo β la ganancia en la zona lineal.

Por lo tanto, se sabrá que se ha alcanzado la saturación cuando $I_C \neq \beta * I_B$, adquiriendo un valor menor. Pues en este momento la I_C dependerá de otro factor muy importante de los transistores, la tensión de saturación entre colector y emisor $V_{CE,sat}$. Pues cuando un transistor entra en saturación, independientemente del valor de corriente de base, mantendrá dicha tensión de saturación, la cual implica una caída de tensión entre colector y emisor.

Por último, hay que destacar otra magnitud, la tensión entre base y emisor V_{BE}. Esta tensión aparecerá siempre que se de conducción en el transistor, tanto en la zona lineal como en saturación. Es debida a la unión de semiconductores en la zona de base a emisor, cuyo salto en potencial debe ser superado por la señal de tensión de entrada, de modo que se inicie el paso de corriente. Si la tensión dada por el microcontrolador es inferior a esta, no habrá conducción y por lo tanto el transistor estará en corte.

8.4.3 Modelo de transistor empleado

El diseño debe comenzar por la selección de transistor, pues es su modelo el que indica los valores de las tensiones y ganancias mencionadas en el apartado anterior. Para la realización de este proyecto se ha seleccionado el transistor BC337hfe-25. Se trata de un transistor de tipo NPN, que cumple con las características deseadas y se emplea habitualmente en el laboratorio de máquinas eléctricas.

Todos los datos se han obtenido de su hoja de especificaciones. [23]

Sus datos más importantes para los cálculos requeridos son:

$$V_{BE} = 1{,}2V;\ V_{CE,sat} = 0{,}7V\ ;\ \beta = [100\ ;400]$$

El rango de valores de β es muy amplio, esto es debido a que la regulación en fabricación de la ganancia es muy compleja y dependiente de múltiples factores, así que el fabricante no asegura un valor si no un rango.

Los valores máximos permitidos:

Tensión máxima entre colector y emisor: $V_{CE,max} = 50V$

Corriente máxima de colector: $I_{C,max} = 0{,}8A$

Máxima potencia permitida: $P_{max} = 0{,}625W$

La potencia será calculada con sus condiciones más desfavorables de este modo:

$$P = V_{CE} * I_E \tag{15}$$

8.4.4 Cálculos para el diseño

La única variable de diseño de que se dispone es R_B, pues R_c es la resistencia interna del servopack, cuyo valor es 3300 Ω y esta no debe ser modificada. Por lo tanto, el objetivo es obtener un valor de resistencia para la base, tal que provoque el estado de corte en el transistor ante una entrada de tensión del ESP32 de 0V, y la saturación ante la entrada de 3,3V. Se va a considerar cada uno de estos dos estados por separado, revisando además si cumple los límites de operación del transistor.

8.4.4.1 Cálculo del estado de corte

Se dará siempre que el microcontrolador emita una señal de 0V, pues al no superarse la tensión $V_{BE} = 1{,}2V$ de la unión semiconductora entre base y emisor, no habrá paso de corriente por el colector y el circuito del servopack quedará abierto. Al no haber paso de corriente por el colector, tampoco habrá una caída de tensión en R_c lo que significa que entre el colector y el emisor del transistor aparecerá la tensión de la fuente externa, es decir, $V_{CE} = 24$V. Siendo este valor inferior al máximo permitido $V_{CE,max} = 50V$.

Al no haber paso de corriente por el colector, $I_C = 0A$ la potencia disipada es $P = 0W$

A la vista de los resultados, el transistor escogido soporta eficientemente el estado de corte del sistema.

8.4.4.2 Cálculo del estado de saturación

En el cálculo de este estado se debe obtener el valor de R_B tal que consiga que, ante una señal de 3,3V del microcontrolador, el transistor trabaje en el estado de saturación. Se emplean las siguientes expresiones:

$$I_B = \frac{V_{esp} - V_{BE}}{R_B} > 0 \tag{16}$$

Donde V_{esp} es el valor de tensión de la señal del ESP32 (3,3V).

Obtenida del análisis de tensiones en la malla microcontrolador y base-emisor. Cuyo valor de I_B debe ser mayor que 0 para asegurar conducción por el colector.

$$I_C = \frac{V_f - V_{CE,sat}}{R_C} < \beta(\min) * I_B \quad (17)$$

Donde V_f es el valor de tensión de la fuente de alimentación externa (24V).

Obtenida del análisis de tensiones en la malla fuente de alimentación, colector-emisor. Cuyo valor de I_C debe ser menor que $\beta(\min) * I_B$ para demostrar que se ha abandonado la zona lineal y se encuentra en saturación. Por ello en la hipótesis la tensión entre colector y emisor es la de saturación. Se emplea el menor valor del rango posible para β para asegurar la saturación en el caso más desfavorable.

Finalmente se ha implementado con una resistencia R_B=22KΩ. Para dicho valor los resultados obtenidos son:

$$I_B = 0{,}09545mA;\ I_C = 7{,}06mA < \beta(\min) * I_B = 9{,}545mA$$

Lo que indica su estado de saturación.

$$I_E = I_B + I_C = 7{,}155\text{mA}$$

Se comprueban los valores máximos permitidos:

$$V_{CE} = V_{CE,sat} = 0{,}7V < V_{CE,max} = 50V$$

$$I_C = 0{,}00706A < I_{C,max} = 0{,}8A$$

$$P = I_E * V_{CE} = 0{,}005008W < P_{max} = 0{,}625W$$

Observándose que el transistor soporta el estado de saturación con el diseño empleado.

8.5 Diseño de etapas de alimentación

En este apartado se muestran las decisiones de diseño para la realización de las etapas de alimentación de la interfaz electrónica, encargadas de energizar los circuitos de entrada y salida, los amplificadores operacionales y el propio microcontrolador.

8.5.1 Alimentación de los amplificadores operacionales

Ambos amplificadores operacionales están incluidos en el mismo circuito integrado, como se ha indicado previamente, lo que implica que la alimentación se realice para el integrado y no para cada uno de los amplificadores por separado. Se ha decidido realizar la alimentación con ±15V, pues suele ser un valor estándar para la alimentación de este tipo de componentes, incluido en sus valores recomendados.

Para la obtención de estas tensiones se emplea el convertido continua-continua TMA0515D, fabricado por Traco Power, de cuya hoja de características se han obtenido los siguientes valores. [24]

Tensión de alimentación: +5V.

Corriente de alimentación: 30mA en vacío; 270mA a plena carga.

Tensiones de salida: +15V y -15V, además de la referencia de 0V.

Máxima corriente de salida: +34mA y -34mA

Según la ficha técnica del circuito integrado que contiene a los amplificadores operacionales el consumo de corriente por amplificador es de 2,8mA como valor máximo, medido en condiciones de vacío. Lo que indica su consumo estático. A modo de aproximación, pues la hoja de características no da más información al respecto, se puede sumar a dicha corriente la corriente de salida máxima en cada caso, asimismo se puede incluir la corriente consumida para el ajuste de "offset".

Para el cálculo de la corriente de salida se considera que la impedancia de entrada del servopack es, tal y como indica su manual, de 14 KΩ, tanto para la entrada de par como la de velocidad. [2]

En el caso de la velocidad la tensión de salida máxima es de ±6,6V, lo que implica empleando la Ley de Ohm, una corriente de salida de ±0,47mA.

En el caso del límite de par su tensión máxima es -3,3V, lo que implica una corriente de salida de -0,235mA.

En el caso del ajuste de "offset", la corriente demandada variará en función del tipo de ajuste realizado. En los cálculos se tendrá en cuenta el paso de corriente debido a las resistencias conectadas a las entradas de alimentación. Sin embargo, cabe destacar que dicha corriente variará en función del ajuste del potenciómetro.

En el caso de la velocidad, esta corriente será exclusiva de la alimentación negativa (-15V), pues es a la cual se ha conectado la resistencia de ajuste, el otro borne de la resistencia se conecta a la entrada negativa, la cual cuenta con un tensión de 0V. Considerando la resistencia teóricamente exacta para el ajsute de "offset" (9 KΩ), y aplicando la ley de Ohm, se obtiene un consumo de corriente por parte de la entrada negativa de -1,66mA. Bastante elevada teniendo en cuenta que debe desviar un valor elevado de tensión.

En el caso del par se va a considerar exclusivamente el conjunto de resistencias entre una entrada de alimentación y su homóloga. Esto debido principalmente a que el ajuste de tensión es muy reducido, lo que implica que el potenciómetro igualará las resistencias para ambas alimentaciones, y en ese caso, el consumo de corriente por la rama conectada a la entrada positiva del amplificador será muy reducida, pues tendrá una tensión cercana a 0V. Con estas hipótesis se tiene una resistencia total de 30 KΩ, con una diferencia de potencial de 30V (entre +15 y -15). Absorbiendo una corriente de 1mA, que tendrá valor positivo de salida de la alimentación positiva, y valor negativo de salida de la alimentación negativa.

Debido al par y al ajuste de "offset" de la velocidad, la alimentación negativa está más demandada, y sumando todas sus exigencias en corriente se obtiene:

$$Corriente\ de\ alimentación\ negativa: (-2{,}8) * 2 - 0{,}47 - 0{,}235 - 1{,}66 - 1 = -8{,}965mA$$

$$Corriente\ de\ alimentación\ positiva: (2{,}8) * 2 + 0{,}47 + 1 = 7{,}07mA$$

Siendo valores suficientemente inferiores a los máximos permitidos por el convertidor.

8.5.2 Fuente de alimentación de 5V

Esta fuente de alimentación se encarga de adaptar la tensión de red eléctrica doméstica de 230V de tensión alterna senoidal, a una tensión continua de 5V. Esta es empleada para la alimentación del microcontrolador y del convertidor encargado de alimentar a los amplificadores operacionales.

La fuente empleada es el modelo IRM-03-5, fabricado por Mean Well, los valores característicos importantes se han obtenido de la ficha técnica de esa serie.[25]

Tensión de alimentación: 230V de corriente alterna.

Corriente de alimentación: 40mA tipificada a dicha tensión.

Tensión de salida: 5V de tensión continua.

Corriente máxima de salida: 600mA.

La corriente máxima que debe suministrar la obtenemos de la ficha técnica del convertidor y del microcontrolador.

El microcontrolador consume una corriente en operación normal de 240mA en especial si emplea la funcionalidad de comunicación Wifi, sin embargo, se recomienda en su hoja de características una fuente de alimentación capaz de entregar 500mA.

En el caso del convertidor se especifica un consumo de 30mA en vacío y 270 mA a plena carga, teniendo en cuenta que este suministra un cuarto aproximado de la corriente máxima de salida, podemos extrapolar y aproximar de forma muy conservadora el consumo de corriente del convertidor en 100mA.

De esta forma, y con un amplio margen de seguridad, se puede estimar el consumo en 600mA, coincidiendo este valor con la corriente máxima que puede suministrar la fuente en condiciones nominales.

8.5.3 Fuente de alimentación de 24V

Esta fuente de alimentación es capaz de adaptar la tensión de red eléctrica doméstica de 230V de tensión alterna senoidal, a una tensión continua de 24V, por lo que se encarga de energizar los circuitos de entradas y salidas digitales.

La fuente empleada es el modelo IRM-03-24, fabricado por Mean Well, los valores característicos importantes se han obtenido de la ficha técnica de esa serie.[25]

Tensión de alimentación: 230V de corriente alterna.

Corriente de alimentación: 40mA tipificada a dicha tensión.

Tensión de salida:24V de tensión continua.

Corriente máxima de salida: 125mA.

Para el cálculo de la corriente que debe suministrar se consideran activas todas las señales, tanto de entrada como de salida, para asegurar un funcionamiento con amplio margen de seguridad. Los valores empleados han sido obtenidos de las etapas de diseño de cada circuito respectivamente.

Cada entrada digital del servopack, para activarse, consume por la rama del colector del transistor una corriente de: I_C=7,06mA.

Mientras que las salidas digitales del servopack, al activarse y cerrar el circuito, consumen una corriente de: I=1,52mA.

Por lo tanto, la corriente que debe suministrar la fuente es de:

$$7{,}06 * 3 + 1{,}52 * 4 = 27{,}26mA \approx 30mA$$

Este valor es ampliamente inferior al máximo capaz de suministrar la fuente.

Se observa como esta fuente soporta una menor carga que su homóloga de 5V, este es el motivo por el que se ha decidido implementar los divisores de tensión a 24V en vez de a 5V.

8.6 Otros elementos implementados y distribución de pines empelados

8.6.1 Zócalo del microcontrolador

Elemento que permite la sujeción del microcontrolador a la placa base, así como su conexión eléctrica con los demás circuitos. Se emplea para poder desconectar el microcontrolador cuando se desee, y también para poder realizar las soldaduras en el montaje, pues de otra forma se deberían haber soldado los propios pines del ESP32 provocándole probablemente daños debidos al sobrecalentamiento.

Para su construcción se han empleado dos hileras de zócalos lineales hembra, de 15 conexiones cada hilera, formando finalmente las 30 conexiones de que dispone el microcontrolador.

8.6.2 Bus de datos paralelo

Elemento que permite la conexión de la interfaz electrónica con el servopack mediante señales eléctricas. Todos los sistemas de entrada y salida interactúan con él. El servopack dispone de un puerto paralelo estandarizado de 50 pines, y paso de 1,27mm en sus dos hileras, por lo que el bus debe tener estas mismas características.

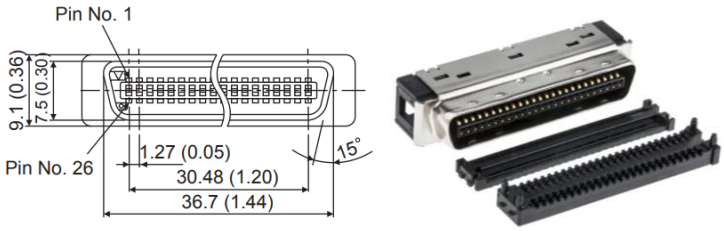

Ilustración 27 . Conector paralelo. Obtenidas del manual del servopack y del sitio web RS Pro. Esquema y fotografía del conector paralelo empleado.

Se ha llevado a cabo empleando el conector modelo 10150-6000EL, fabricado por 3M. [26]

El cual soporta corriente de hasta 1A y tensiones de 200V, siendo valores muy superiores a los empleados en este proyecto, donde los valores de corrientes de las entradas y salidas difícilmente superan los 7mA.

Sin embargo, para la realización del proyecto no se emplean todos los pines, por lo que en la placa base solamente habrá conexión de 21 de los siguientes cables. Esto último se ha conseguido separando y soldando individualmente cada cable al puerto de conexión con la placa base, desde el cual cada señal es dirigida a su circuito correspondiente.

El bus paralelo total está formado por dos buses paralelos de 25 cables cada uno, pues el conector empleado solo permite la inserción de los 50 cables de este modo. Por otra parte, esto ha facilitado la separación de los cables y su soldadura en el lado de conexión con la interfaz electrónica.

8.6.3 Recopilatorio de GPIOs, pines y señales eléctricas

Función	GPIO del ESP32	Pin del servopack	Tipo de señal
Control de la velocidad	22	5	Entrada analógica
Control del par	23	9	Entrada analógica
Activación del servo	2	40	Entrada digital
Límite de par horario	4	45	Entrada digital
Límite de par antihorario	15	46	Entrada digital
Pulsos A encoder	26	33 o 34	Salida encoder
Pulsos B encoder	25	35 o 36	Salida encoder
Velocidad de consigna alcanzada	35	25 y 26	Salida digital
Velocidad indicada superada	34	27 y 28	Salida digital
Servo preparado	39	29 y 30	Salida digital
Señal de alarma	36	31 y 32	Salida digital
Señal de referencia de masa	GND	1, 6 y 10	Referencia eléctrica
Tensión de 24V	-------	47	Alimentación de entradas

Tabla 2. Recopilación de pines y GPIOs empleados

Se hace notar que la definición de entrada o salida se ha referido desde el punto de vista del usuario y del servopack.

En el caso de los pulsos del encoder se da a elegir entre dos posibles opciones, pues los pulsos de cada señal (A o B) se envían por dos pines diferentes, uno con la señal y otro con la misma señal desplazada medio período, con el objetivo de emplear un “line receiver” y de este modo eliminar el posible ruido. No obstante, en este proyecto no se emplea dicho dispositivo, y los pulsos son enviados individualmente directamente a las entradas de interrupción del microcontrolador, por lo que finalmente solo se emplean los pines 34 y 36 para este propósito.

Las salidas digitales por su disposición emplean dos cables del bus paralelo, ya que su funcionamiento se basa en la apertura o cierre del circuito, por lo que este debe partir y regresar de la interfaz electrónica para tener continuidad.

En cuanto a las señales de masa, la ubicada en el pin 1 es la general para todo el servopack, mientras que las ubicadas en los pines 6 y 10 son las referenciales para las entradas analógicas, como se ve en el esquema de entrada. Puesto que la placa emplea el mismo potencial de referencia, estos puntos son el mismo, eléctricamente hablando, y estarán conectados a la patilla de masa del microcontrolador.

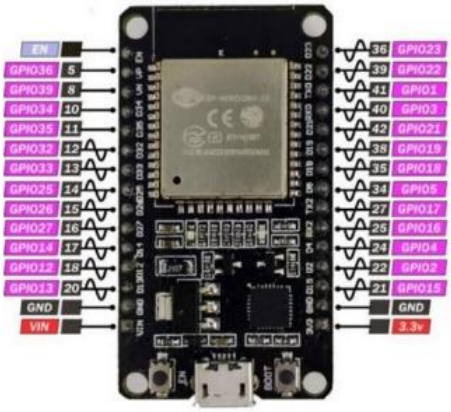

Ilustración 28. Distribución de los GPIO del ESP32. Imagen obtenida del sitio web DescubreArduino.com

8.7 Esquema de conjunto

Se expone definitivamente el esquema eléctrico de la interfaz electrónica con sus circuitos y conexiones. Se debe destacar que, para todos los circuitos, la referencia de 0V es siempre la misma, y está marcada por el pin de masa del microcontrolador. La líneas de colores representan conexiones cableadas, por lo que el cruce con otras de diferente color no implica conexión, solamente cuando se trata del final de línea en algún componente o a continuación del final de otra línea. Las líneas de igual color que se cruzan pero que en su cruce se dibuja un salto no están conectadas.

Aparece además una tabla con el valor de los componentes pasivos empleados en el montaje definitivo, obtenidos de los cálculos teóricos.

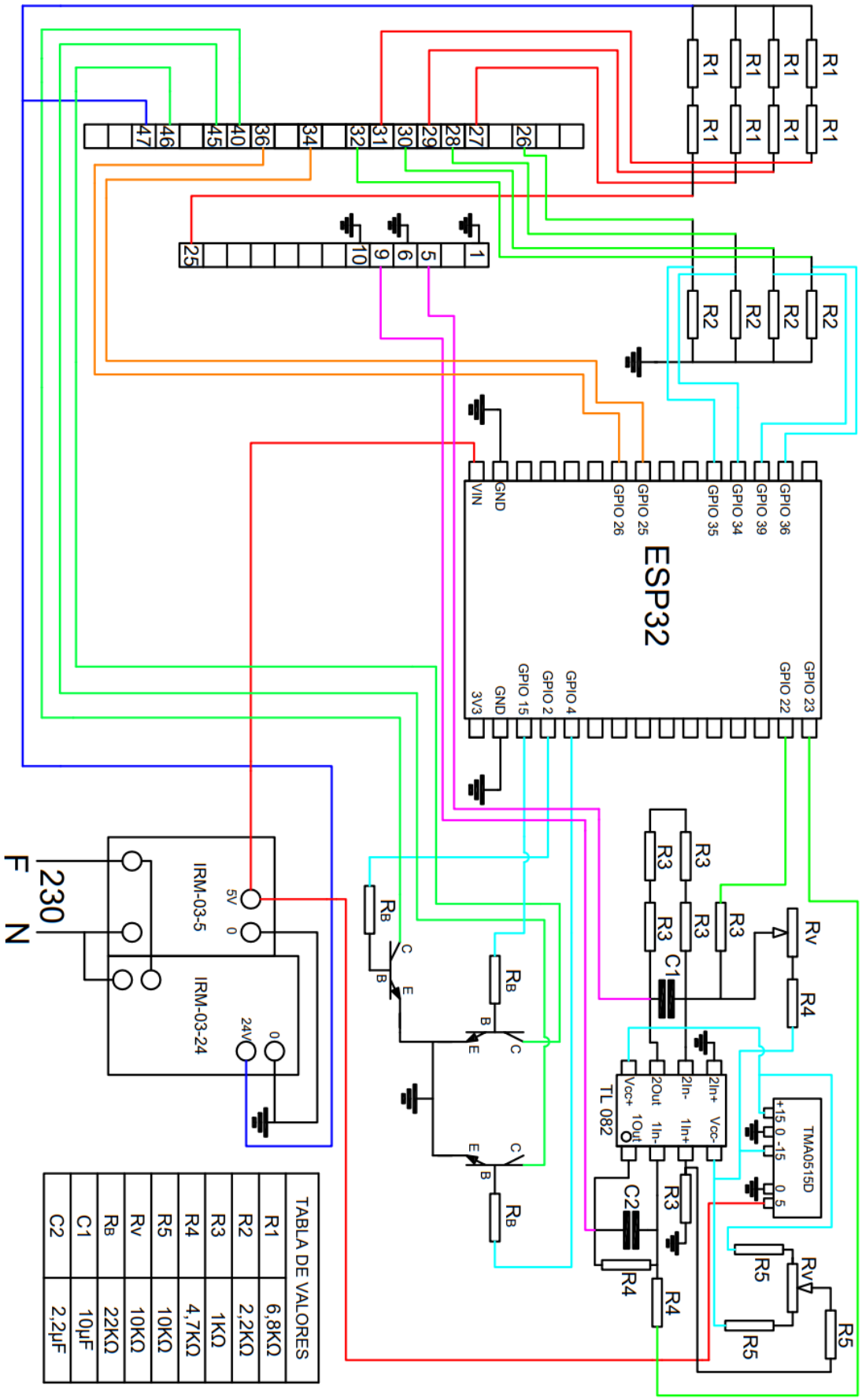

Ilustración 29. Esquema de la interfaz electrónica. Fuente propia

8.8 Construcción del prototipo

En este último apartado del capítulo se pretende mostrar cual ha sido el proceso de montaje de la placa, la recopilación de los elementos que la conforman y las herramientas empleadas en su construcción.

8.8.1 Herramientas y materiales empleados en el proceso de construcción

8.8.1.1 Placa de prototipos o placa de matriz

Sobre este elemento se construye toda la interfaz electrónica, se trata de una placa de circuito impreso de baquelita de forma rectangular, con una matriz de taladros por los que se insertan las patillas de conexión de los componentes eléctricos a soldar. En su cara superior se depositan los componentes, y en la inferior es donde se realiza la soldadura. En la parte inferior, rodeando a los agujeros pasantes, hay anillos serigrafiados de cobre de forma circular e inconexos entre ellos que permiten una buena ejecución de la soldadura con Estaño y su posterior interconexión.

8.8.1.2 Estaño

Las soldaduras se han realizado empleando este material, típicamente usado para aplicaciones electrónicas. Se trata de un hilo de aleación conductora de Estaño, con un alma de flux desoxidante, de bajo punto de fusión (aproximadamente 220ºC) que actúa como elemento de conexión eléctrica y de fijación para los componentes.

8.8.1.3 Soldador de 25W

Herramienta capaz de elevar su temperatura y de transmitir calor al estaño para lograr su fusión y poder realizar la soldadura con garantías.

8.8.1.4 Desoldador de 25W

Se trata de un elemento similar al soldador, pero cuya punta es hueca en su interior y dispone de un elemento de succión para el estaño líquido. Se ha empleado para deshacer errores durante el proceso de soldadura.

8.8.1.5 Elemento de sujeción

Se trata de una bancada dotada de pinzas de sujeción y una lupa que ayuda en la tarea de posicionamiento de los componentes y de la placa de prototipos. Ha facilitado sobremanera la construcción del dispositivo, pues permite el volteo constante de la placa para introducir el componente por una cara, y soldar por la contraria.

8.8.1.6 Pistola de cola termofusible

Empleada para la colocación de adhesivo de refuerzo en el puerto del bus paralelo en la parte de la interfaz electrónica.

8.8.1.7 Cables de cobre aislados

Empleados para la materialización de las conexiones eléctricas a distancia mayor de varios agujeros de la placa de prototipos. Pues los componentes contiguos se han conectado empleando puentes de estaño fundido entre los agujeros.

Ilustración 30. Herramientas de soldadura. Fuente propia. Se muestran de izquierda a derecha el soldador, el desoldador y la pistola de cola termofusible empleados en la construcción del prototipo.

8.8.2 Proceso constructivo

Se muestra a continuación la explicación resumida de las actividades realizadas para construir el prototipo de la interfaz electrónica, así como la indicación de los componentes finalmente utilizados. Se han ordenado siguiendo el proceso real.

8.8.2.1 Construcción del zócalo

Como se ha comentado anteriormente, está formado por dos hileras de 15 zócalos en línea, separadas el tamaño del microcontrolador. Pese a que no todas las conexiones son empleadas, sí que se han soldado los 30 conectores, con el objetivo de dotar de mayor resistencia mecánica a la conexión. Para evitar el sobrecalentamiento y deformación del plástico se han soldado alternativamente de un lado a otro, comenzando por los 4 puntos extremos para asegurar la nivelación y la correcta entrada del microcontrolador.

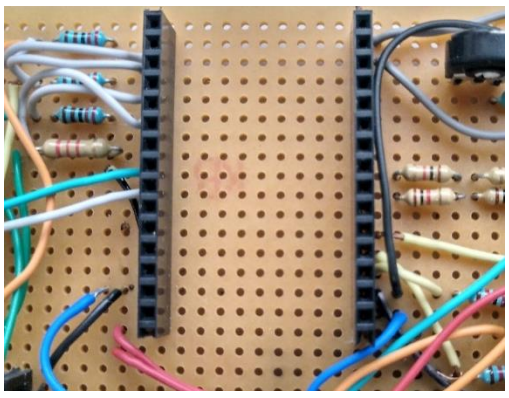

Ilustración 31. Zócalo del microcontrolador. Fuente propia.

8.8.2.2 Montaje de los divisores de tensión para las salidas digitales

Se han empleado las siguientes resistencias: 4 de 2200Ω y 8 de 6800Ω (dispuestas en serie de dos en dos para obtener 4 de 13600Ω)

Por un lado, se han conectado varios puntos de la placa empelando puentes de estaño, pues son comunes las tensiones de 24V y 0V para todos los divisores de tensión. Conectado al punto de los 24V se sueldan en serie las dos resistencias que conforman la de 13,6KΩ, y conectado al punto de masa se suelda la correspondiente a 2,2KΩ. Finalmente, las conexiones libres de ambas resistencias se conectan mediante dos cables al puerto de conexión con el servopack. Esto se ha realizado 4 veces, una por salida. Por otro lado, se une por cable el punto de conexión de 0V al zócalo correspondiente a la masa del microcontrolador.

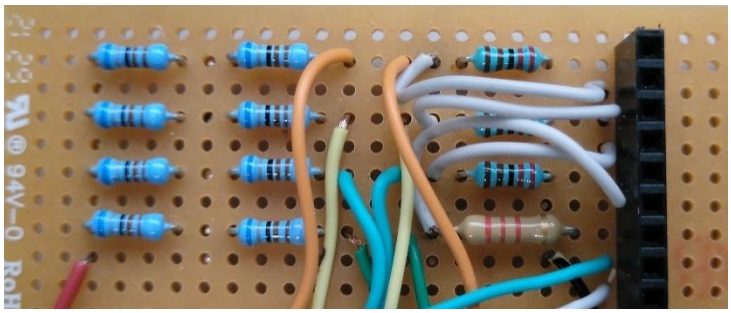

Ilustración 32. Implementación de los divisores de tensión. Fuente propia.

8.8.2.3 Construcción de los filtros para las entradas analógicas

Se han empleado los siguientes componentes: Integrado TL082X, convertidor TMA0515D, 2 resistencias variables de 10KΩ, 6 resistencias de 1KΩ, 3 resistencias de 4,7 KΩ, 3 resistencias de 10 KΩ, 1 condensador de 10 µF, 1 condensador de 2,2 µF y un zócalo para circuito integrado.

Soldadura del zócalo del circuito integrado, soldando a su alrededor los diferentes componentes que forman el filtro para cada uno de los amplificadores operacionales. Se debe destacar que la resistencia necesaria de 4 KΩ se ha construido empleando 4 de 1 KΩ en serie. Se debe de tener especial cuidado de no realizar cortocircuitos con los puentes de estaño realizados, pues se dispone de poco espacio entre puntos que deben estar aislados mutuamente. Posteriormente se ha soldado el convertidor de continua, dejando tiempo entre soldaduras para no sobrecalentar el dispositivo. A continuación, se han instalado los potenciómetros junto con sus respectivas resistencias para el ajuste de “offset”. Por último, se ha realizado el cableado de los puntos inconexos más alejados, es decir: las alimentaciones de los amplificadores y del ajuste, las señales de entrada desde el zócalo del ESP32, y las salidas de los filtros enviadas al puerto de conexión con el servopack.

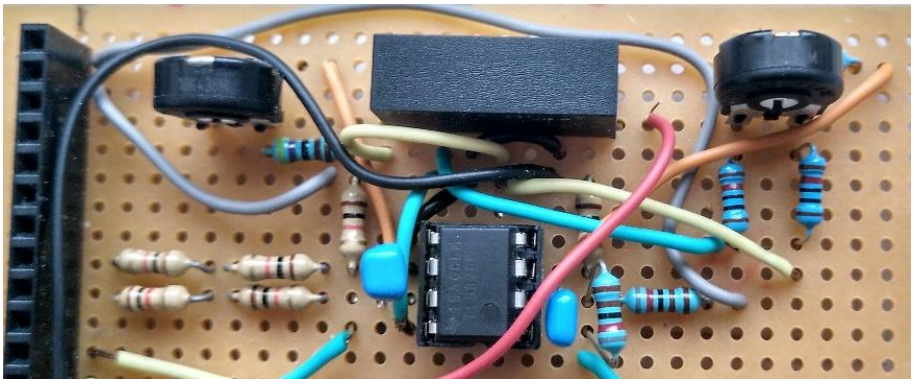

Ilustración 33. Etapa de filtrado de las señales analógicas. Fuente propia.

8.8.2.4 Montaje de los transistores de potencia para las entradas digitales

Los componentes empleados han sido: 3 resistencias de 22 KΩ y 3 transistores BC337.

La construcción se realiza con la soldadura de un punto común de masa para los emisores de los tres transistores. Una vez soldada dicha unión en la base de cada transistor se ha conectado su resistencia. Finalmente se realiza el cableado: conectando las resistencias a los pines del zócalo del ESP32, los colectores de cada transistor con su correspondiente conexión del puerto del bus paralelo y el punto común de los emisores a la masa del ESP32. La soldadura se debe realizar asegurando una buena refrigeración de los transistores, pues las uniones de semiconductores son especialmente sensibles a las altas temperaturas.

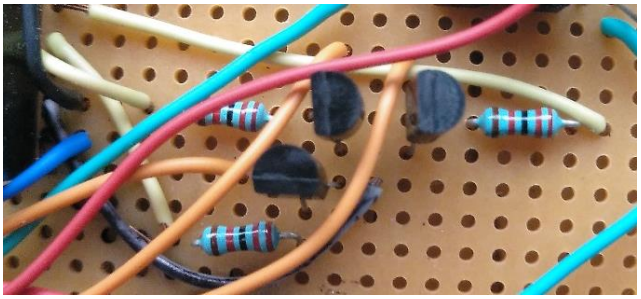

Ilustración 34. Transistores de potencia. Fuente propia.

8.8.2.5 Construcción del sistema de alimentación

En esta etapa se emplean las dos fuentes de tensión encargadas del paso de 230V de alterna a sus respectivos 5V y 24V.

En primer lugar, se suelda a sus entradas de alimentación un cable con enchufe preparado para la red doméstica, soldando la fase y el neutro en los pines de entrada de las fuentes y aislando con tubo de elastómero termo retráctil las partes expuestas con tensiones de 230V. El anclaje a la placa se ha realizado mediante el uso de varias bridas, con el objetivo de que no sufran las conexiones eléctricas, en especial la unión con el cable de alterna. Finalmente, se ha cableado la alimentación desde cada fuente a su destino, siendo para el caso de los 24V las salidas digitales y el pin de conexión correspondiente del servopack, y en el caso de los 5V la alimentación del convertidor y del microcontrolador. El borne negativo de ambas tensiones se ha conectado a la masa del microcontrolador para establecer la referencia de tensiones.

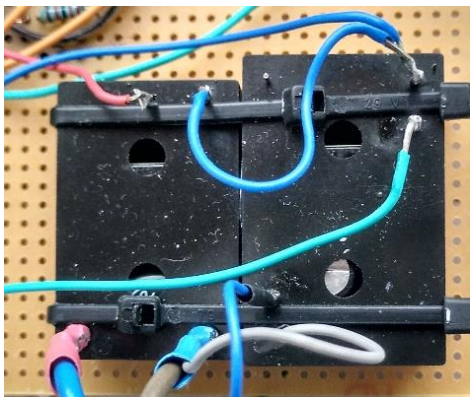

Ilustración 35. Fuentes de alimentación. Fuente propia.

8.8.2.6 Montaje del bus paralelo de unión entre servopack e interfaz electrónica

Se ha empleado el conector de puerto paralelo de 50 pines 10150-6000EL, 2 buses paralelos de 25 cables, dos hileras de zócalos macho y dos hileras de zócalos hembra.

El conector normalizado empleado requiere de la conexión de dos buses distintos de 25 hilos cada uno, su montaje se realiza a presión dentro del cabezal para realizar la conexión entre el conector y los cables. En el lado de conexión con la placa se han separado y seleccionado los cables que se requieren para este proyecto, dejando los demás inconexos. Los cables necesarios se han soldado a los zócalos macho a modo de creación de un conector artesanal, y han sido recubiertos con cola termofusible con el objetivo de evitar posibles cortocircuitos y dotar de resistencia mecánica al conjunto de pines.

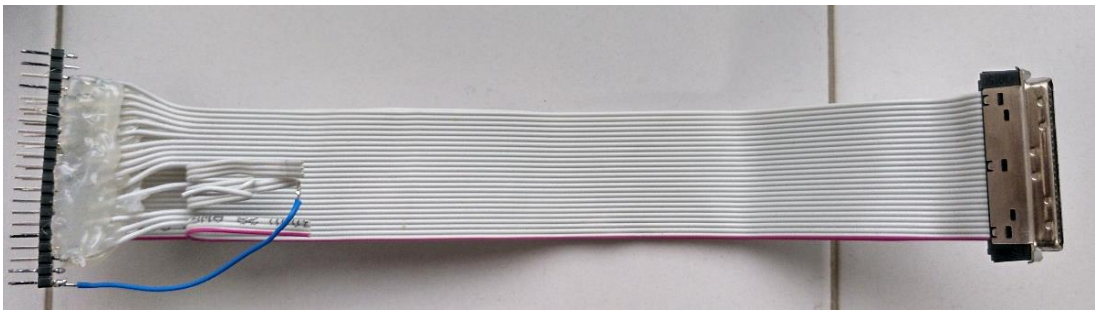

Ilustración 36. Bus paralelo. Fuente propia.

En la placa base, se han soldado los zócalos hembra que actúan como conectores para los dos buses paralelos de conexión. Desde este punto, se cablean todas las entradas y salidas a sus circuitos correspondientes o a sus referencias de tensión indicadas.

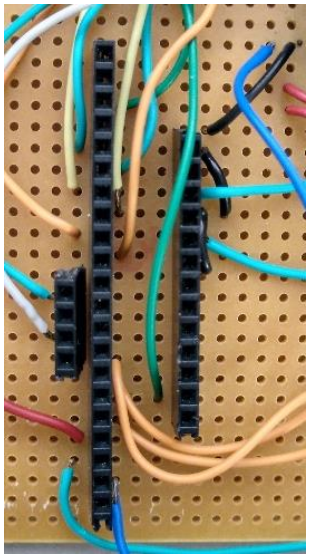

Ilustración 37. Puerto bus paralelo.
Fuente propia.

8.8.3 Prototipo finalizado

Se muestra finalmente una imagen del prototipo completo, conectado al bus paralelo y con el microcontrolador insertado.

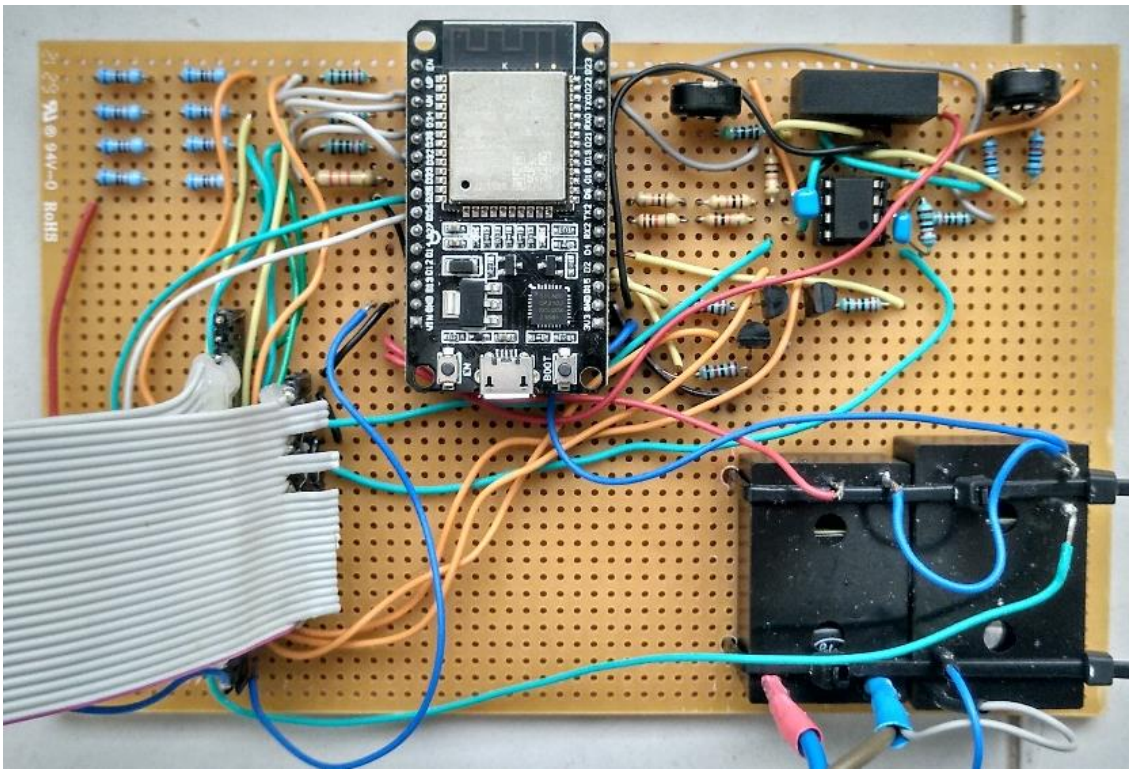

Ilustración 38.Prototipo de interfaz electrónica finalizado. Fuente propia.

9. MANUAL DE USUARIO

En este capítulo se muestran las configuraciones que debe adoptar el usuario, tanto del microcontrolador como del servopack para poder emplear con todas sus características el sistema construido.

9.1 Configuración de Arduino IDE, selección de red y dirección IP

En el caso de que sea la primera vez que vaya a usarse el conjunto, se debe configurar de tal modo que se pueda variar su código, con el objetivo de seleccionar la red Wifi a utilizar, así como la dirección IP en la que va a establecerse el servidor. Para ello se requiere del uso y configuración del entorno de desarrollo integrado de Arduino, de tal modo que se pueda modificar, compilar y cargar adecuadamente el programa en el microcontrolador.

1. Descarga e instalación del programa Arduino IDE: Se debe descargar del apartado software del sitio web oficial de Arduino. Se recomienda la descarga del ejecutable instalador para el sistema operativo de que disponga el usuario. Una vez completada la descarga se instala el programa en el ordenador.

2. Configuración de Arduino IDE para trabajar con ESP32: Una vez dentro del entorno de desarrollo se debe ir a "Archivo"-"Preferencias" y copiar el siguiente enlace en el apartado "URLs adicionales de gestor de placas": https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json Seguidamente se debe acceder a "Herramientas"-"Placa"-"Gestor de placas", donde debe buscarse e instalarse el paquete de herramientas llamado "esp32". Una vez instalado se accede a "Herramientas"-"Placa"-"esp32" en el que aparece un desplegable con todos los modelos disponibles, de entre los cuales se debe seleccionar el modelo "DOIT ESP32 DEVKIT V1". A modo de comprobación puede conectarse la placa al ordenador empleando el cable micro USB-USB, en caso de no detección se recomienda proceder con el punto 3.

3. Error en la detección de la placa: En el caso de haber seguido correctamente las instrucciones del punto 2 y no se detecte la placa se debe descargar un controlador convertidor USB a UART. Se recomienda el uso del siguiente sitio web https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads descargando el archivo correspondiente al sistema operativo del usuario. Una vez instalados los drivers se debe acceder en "Este equipo"-"Administrador de equipos"-"Administrador de dispositivos", donde si está conectado el ESP32 por cable aparecerá un apartado llamado "Otros dispositivos" y dentro "CP2102 USB to UART Bridge Controller", seleccionando este último con clic derecho se debe actualizar el software del driver, se recomienda realizarlo mediante búsqueda en el dispositivo para localizar manualmente la carpeta instalada con los drivers y proceder a la actualización.

4. Adición de librerías externas: Para una correcta compilación del programa se deben instalar las librerías empleadas en el código, en primer lugar, deben ser descargadas en formato .zip de la página de GitHub. Para ello en cada una de sus páginas se debe presionar el botón verde llamado "code" y seleccionar la opción "Download ZIP". Estos son los enlaces directos a las direcciones de cada librería:

https://github.com/me-no-dev/ESPAsyncWebServer
https://github.com/Links2004/arduinoWebSockets
https://github.com/bblanchon/ArduinoJson
Posteriormente a la descarga se deben instalar en Arduino IDE. Para ello dentro del programa se accede a "Sketch"-"Incluir biblioteca"-"Añadir biblioteca ZIP", en el cual se debe seleccionar los archivos descargados. Para más información acerca de las librerías se remite al apartado 7.4 de la memoria.

5. Modificación del código de programa: Se abre el archivo que contiene el código del programa en Arduino IDE, donde en las líneas 18 a 29 aparecen las configuraciones de red y de dirección IP estática a seleccionar. En la configuración de nombre de red y contraseña se debe escribir dentro de las comillas, mientras que en la dirección IP se escribe sustituyendo la dirección predefinida si se desea. En router domésticos por lo general solo será necesario el cambio del último término de la dirección, sin embargo, si se desea cambiar alguno de los demás términos se deberán cambiar tanto en la línea 24 como en la 25. Finalmente se carga el código en el ESP32 con el botón superior izquierdo que aparece en Arduino IDE.

6. Comprobación de comunicación Wifi: Se recomienda la realización de pruebas de conexión inalámbrica con el microcontrolador antes de su inserción en la interfaz electrónica y por supuesto antes de iniciar el sistema. Para ello una vez cargado el sistema, el microcontrolador se alimentará a través del cable de comunicación e iniciará la ejecución. Indicando su conexión y la dirección IP por el monitor serie de Arduino IDE, para acceder a esta herramienta se emplea el botón superior derecho de la pantalla con forma de lupa. En este momento cualquier dispositivo que busque en su navegador de internet esta dirección podrá comunicarse con el servidor, y por el monitor serie aparecerán los flujos de datos para una correcta comprobación.

9.2 Configuración del servopack

Ciertos parámetros de funcionamiento del servopack deben estar sincronizados con los valores empleados en el código del microcontrolador para una correcta comunicación entre ambos dispositivos, pese a haberse empleado mayormente los valores de fábrica, es conveniente la revisión de todos aquellos parámetros principales relacionados con el proyecto y con su operación directa. Ante cambios más profundos en el funcionamiento del servopack se remite a su manual de usuario. [2]

1. Alimentación del servopack: Se conecta a la red eléctrica de 230V empleando conexiones industriales y con todas las seguridades que especifica el reglamento. La entrada de tensión con fase conectada en los terminales L1 y L1C, y neutro conectado a los terminales L2 y L2C. Por otro lado, las entradas +1 y +2 se conectan mediante un cable. Finalmente, la masa del servopack se debe conectar a la tierra de la instalación. Es recomendable el uso de interruptores adecuados para controlar la alimentación.

2. Modificación de parámetros: Se emplea el panel de operaciones con la pantalla de cristal líquido. La cual dispone de los siguientes botones: MODE/SET ; UP key ; DOWN key ; DATA/<<. El primer botón se emplea para variar la selección de parámetros o funcionamiento, y también como botón de aceptación. Los botones UP y DOWN se emplean para modificar el valor de las cifras. Y el botón DATA se emplea para indicar el cambio de un parámetro, y para seleccionar la cifra a cambiar.

Ilustración 39. Selector de parámetros. Fuente propia. Imagen de la pantalla de cristal líquido y la botonera que permiten variar el valor de los parámetros del servopack.

3. Parámetros de las entradas analógicas de velocidad y par: se indica el nombre de parámetro y a su derecha el valor a introducir.
    a. Pn002: XXX3. Las X indican cualquier cifra. Selecciona el modo de funcionamiento de control de velocidad con límite de par.
    b. Pn300: 600. Valor de fábrica. Selecciona la ganancia en velocidad.
    c. Pn400: 30. Valor de fábrica. Selecciona la ganancia en límite de par.
    d. Pn402: 800. Valor de fábrica. Establece límite de par permanente en sentido horario.
    e. Pn403: 800. Valor de fábrica. Establece límite de par permanente en sentido antihorario.
    f. Pn100: 40. Valor de fábrica. Determina la velocidad de respuesta ante cambios de velocidad.

4. Parámetros de las entradas digitales de control: se indica el nombre de parámetro y a su derecha el valor a introducir.
    a. Pn404: Valor deseado por el usuario entre [0 ; 800] que indica el valor porcentual sobre el par nominal del límite de par en sentido horario al activarse la entrada de “Límite de par en sentido horario”
    b. Pn405: Valor deseado por el usuario entre [0 ; 800] que indica el valor porcentual sobre el par nominal del límite de par en sentido antihorario al activarse la entrada de “Límite de par en sentido antihorario”

5. Parámetros de las entradas digitales de control: se indica el nombre de parámetro y a su derecha el valor a introducir:
    a. Pn503: 10. Valor de fábrica. Indica la anchura del rango en rpm que activa la salida de “Velocidad de consigna”.
    b. Pn502: Valor deseado por el usuario entre [1 ; 10000] que indica el valor en rpm de la velocidad a partir de la cual se activará la salida “Velocidad superada”.

6. Parámetro que regula el envío de pulsos del encoder: se indica el nombre de parámetro y a su derecha el valor a introducir.
    a. Pn201: 1000. Indica la cantidad de pulsos por revolución que envía el encoder.

9.3 Montaje del conjunto

Una vez realizadas todas las configuraciones previas se indica el montaje del servomotor y la conexión de la interfaz electrónica con el servopack.

1. Conexión del servomotor: El cableado del circuito de potencia entre el servomotor y el servopack se realiza mediante una conexión trifásica, por lo que se emplean tres cables conectados a la salida del inversor del servopack (de nombres U, V y W) y por el otro lado a los bornes del servomotor, además se conectan las tierras de ambos dispositivos. Finalmente, se debe conectar el cable de comunicación del encoder con el servopack en la conexión CN2 empleando el recomendado por el fabricante. Por seguridad es imprescindible que esta conexión se realice con el servopack desconectado de la línea eléctrica.

2. Conexión con la interfaz electrónica: en primer lugar, se inserta el microcontrolador al zócalo de la placa base de tal modo que se empleen todos los pines del ESP32 y su puerto micro-USB quede en la parte interior de la placa. Posteriormente se procede a la conexión del bus paralelo de la interfaz electrónica al conector CN1 del servopack. En este punto no se aconseja mantener la conexión del microcontrolador con el ordenador del usuario mediante el puerto serie, por razones de seguridad del dispositivo. Además, el objetivo de este es dar información sobre la comunicación con el servidor y esta comprobación debe haberse realizado con anterioridad. Finalmente se conecta la alimentación de la interfaz a la red eléctrica.

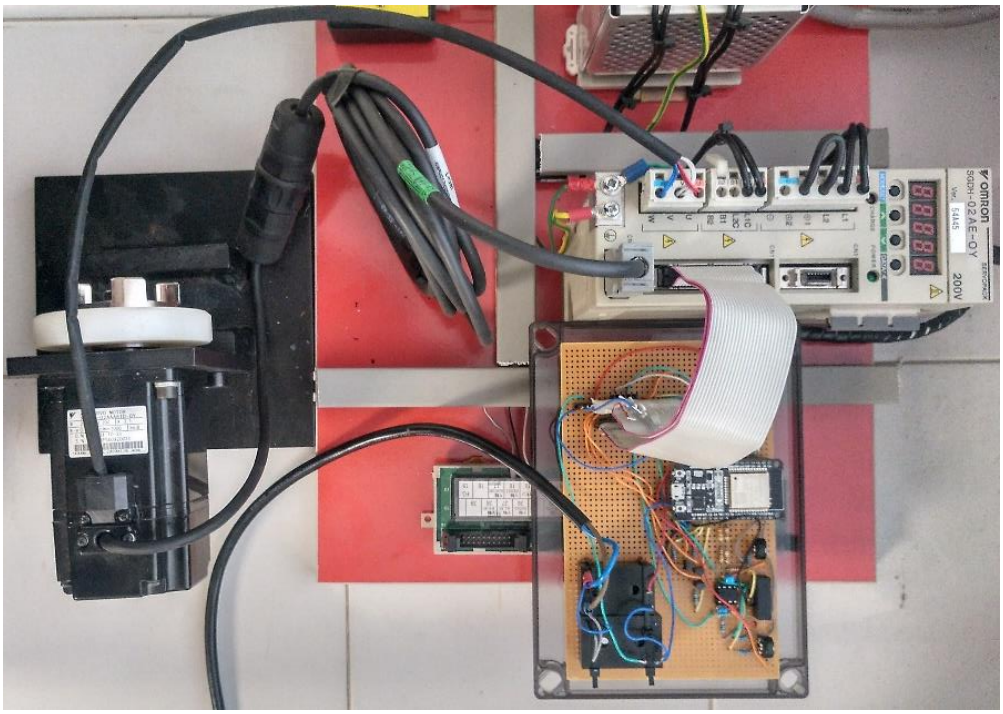

Ilustración 40.Conjunto montado. Fuente propia. Imagen que muestra las conexiones realizadas entre los diferentes elementos del sistema.

9.4 Funcionamiento normal del sistema

1. Inicio del sistema: Una vez comprobadas y aseguradas todas las conexiones se alimentan de forma simultánea el servopack y la placa base, dando comienzo al funcionamiento del sistema.

2. Conexión al servidor: El usuario debe asegurarse que tanto el microcontrolador como el dispositivo que va a emplearse para el control estén conectados a la misma red Wifi. En el navegador web de este dispositivo se debe escribir la dirección IP seleccionada para el servidor, para que le aparezca por pantalla el sitio web con el tablero de mandos del servomotor. Para una explicación más detallada del tablero de mandos se remite al punto 6.2 de la memoria.

3. Ajuste manual de tensión: Si durante la operación normal se detecta que el motor no alcanza la velocidad indicada, o que arranca repentinamente cuando debería estar detenido, se debe ajustar la tensión de desbalance de los amplificadores operacionales. Para ello se debe solicitar una velocidad y un límite de par de valor 0 en ambos casos, con la ayuda de un voltímetro o preferiblemente de un osciloscopio se debe medir la tensión entre la salida de cada etapa de filtrado y la masa del sistema y con ayuda de un destornillador plano se debe ajustar el valor de los potenciómetros de tal forma que la diferencia de tensión entre la salida de los filtros y masa sea de valor 0V.

10. CONCLUSIÓN Y PROPUESTAS DE MEJORA

Como cierre a esta memoria y una vez habiéndose explicado con detalle todos los elementos implicados en el trabajo, sus motivos de diseño y su construcción, se presenta una conclusión al proyecto y una serie de propuestas de mejora para el mismo.

El objetivo principal del proyecto de control de un servomotor mediante conexión inalámbrica e interfaz HTML se ha resuelto de forma satisfactoria, permitiendo así el empleo del banco de ensayos del servomotor de una forma más sencilla y remota. Facilitando su uso a toda aquella persona que disponga de un dispositivo con capacidad de conexión Wifi.

Se ha logrado una comunicación casi inmediata entre usuario y servomotor empleando las herramientas del diseño de servidores y de conexiones inalámbricas que mejor se adaptan a los objetivos del proyecto. Adicionalmente se ha explorado en el uso de servidores, sus estándares de comunicación y las tecnologías de software disponibles, haciéndolo eficiente y versátil. Por otro lado, se ha conseguido combinar diferentes lenguajes de programación, cada uno con un objetivo y una funcionalidad diferentes, en un conjunto capaz de funcionar sincronizado y cumpliendo con las metas deseadas.

Con la construcción del prototipo se ha podido materializar el proyecto, por lo que ha salido de la teoría y ha debido de enfrentarse al mundo real, más impredecible y complicado que las simulaciones o los cálculos. Debido a ello se han realizado variaciones y mejoras, especialmente en el ámbito de la electrónica analógica, que han permitido al sistema ser capaz de enfrentarse a condiciones inesperadas y desfavorables. Paralelamente, el hecho de construir un dispositivo y comprobar efectivamente que funciona según los requerimientos demuestra que el trabajo teórico y de diseño se ha realizado de la forma acertada.

Finalmente se ha demostrado la potencia de que dispone la electrónica integrada, pues con un microcontrolador barato y de pequeño tamaño se soporta y gestiona un servidor web al mismo tiempo que se adquieren datos y se envían señales eléctricas de accionamientos industriales.

A lo largo del diseño del sistema, de la confección de la memoria y durante el proceso de construcción del prototipo se han detectado ciertos aspectos donde se podría haber actuado de otro modo o donde hay aún margen de mejora, principalmente se destacan los siguientes puntos.

- La implementación de elementos que permitieran un control autónomo del servomotor durante cierto tiempo como curvas de arranque o frenado a una velocidad o par progresivos o capacidad de temporización de la operación.

- Un sistema que permitiera imprimir por pantalla la curva de velocidad del servomotor en función del tiempo y que se pudiera ver su operación a lo largo de un intervalo temporal seleccionable.

- En el caso de la interfaz electrónica, se podría haber construido una carcasa o envolvente en el que poder alojar la placa base, que la protegiera frente a golpes, contactos eléctricos indeseados y que pudiera acoplarse al servopack. Por otro lado, se debería haber procedido a la soldadura de los cables del bus paralelo directamente sobre la placa base, en vez de construir un puerto artesanal con zócalos que, aunque permita intercambiabilidad, fragiliza enormemente esa unión.

- Se podría implementar un servidor con mayor seguridad empleando claves de acceso o una mayor personalización de lo que se les permite hacer a los diferentes clientes que se conectan, evitando la actuación indeseada de terceros y dotando de mayor libertad a los usuarios.

En definitiva, al tratarse de un proyecto con tantas disciplinas relacionadas y gran diferenciación de sus partes, es de esperar que un estudio focalizado en alguna de ellas muestre carencias y amplios márgenes de mejora para llegar a ser un producto comercializable. Sin embargo, el objetivo es la creación de un prototipo que cumple todos y cada una de las metas planteadas al comienzo de esta memoria, capaz de integrar las diversas ramas empleadas de la técnica en un solo objetivo que de forma individual no hubieran podido alcanzar.

11. BIBLIOGRAFÍA Y WEBGRAFÍA

Se emplea el estilo de referencias IEEE.

[1] "Objetivos y metas de desarrollo sostenible - Desarrollo Sostenible." https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/ (consultado Mayo 14, 2023).

[2] S. Servopack, "H / Sgdh User ' S Manual."

[3] "An Introduction to Encoder | Encoder Product Company." https://www.encoder.com/article-que-es-un-encoder (consultado Abril 28, 2023).

[4] "ESP32 Technical Reference Manual About This Manual," 2023, Visitado: Abril 27, 2023. [Online]. Disponible en: https://www.espressif.com/en/support/download/documents.

[5] "About This Guide," Visitado: Mayo 05, 2023. [Online]. Disponible en: https://www.espressif.com/en/subscribe.

[6] "Arduino MKR WiFi 1010 — Arduino Official Store." https://store.arduino.cc/products/arduino-mkr-wifi-1010?pr_prod_strat=use_description&pr_rec_id=9dd52d82f&pr_rec_pid=5517833109655&pr_ref_pid=5517831307415&pr_seq=uniform (consultado Mayo 05, 2023).

[7] "La revolución de la Web Asíncrona." https://dosideas.com/noticias/java/547-la-revolucion-de-la-web-asincronica (consultado Mayo 16, 2023).

[8] "Generalidades del protocolo HTTP - HTTP | MDN." https://developer.mozilla.org/es/docs/Web/HTTP/Overview (consultado Mayo 03, 2023).

[9] "¿Qué es un puerto de ordenador? | Puertos en la red | Cloudflare." https://www.cloudflare.com/es-es/learning/network-layer/what-is-a-computer-port/ (consultado Mayo 03, 2023).

[10] "Métodos de petición HTTP - HTTP | MDN." https://developer.mozilla.org/es/docs/Web/HTTP/Methods (consultado Mayo 16, 2023).

[11] "Web sockets vs Ajax | Learn Top 4 Beneficial Differences With Infographics." https://www.educba.com/web-sockets-vs-ajax/ (consultado Mayo 16, 2023).

[12] IBM, "WebSocket - Documentación de IBM," 2023. https://www.ibm.com/docs/es/was/9.0.5?topic=applications-websocket (consultado Mayo 16, 2023).

[13] "WebSocket | Un canal de comunicación para la web en tiempo real - IONOS." https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-websocket/ (consultado Mayo 16, 2023).

[14] "WebSocket." https://es.javascript.info/websocket (consultado Mayo 16, 2023).

[15] "Archivo Json: descubre qué es y para qué sirve." https://rockcontent.com/es/blog/archivo-json/ (consultado Mayo 16, 2023).

[16] "HTML - Concepto, historia, cómo funciona y etiquetas." https://concepto.de/html/ (consultado Mayo 16, 2023).

[17] “¿Qué es Arduino? | Arduino.cl - Compra tu Arduino en Línea.” https://arduino.cl/que-es-arduino/ (consultado Mayo 16, 2023).

[18] “Cómo usar las interrupciones del ESP32 GPIO con Arduino.” https://descubrearduino.com/interrupciones-esp32-gpio/ (consultado Mayo 16, 2023).

[19] “GitHub - me-no-dev/ESPAsyncWebServer: Async Web Server for ESP8266 and ESP32.” https://github.com/me-no-dev/ESPAsyncWebServer (consultado Mayo 10, 2023).

[20] “GitHub - Links2004/arduinoWebSockets: arduinoWebSockets.” https://github.com/Links2004/arduinoWebSockets (consultado Mayo 10, 2023).

[21] “GitHub - bblanchon/ArduinoJson: JSON library for Arduino and embedded C++. Simple and efficient.” https://github.com/bblanchon/ArduinoJson (consultado Mayo 10, 2023).

[22] D. Information, “TL08xx FET-Input Operational Amplifiers,” vol. 082, no. February 1977, 2021.

[23] “BC337 NPN datasheet.” .

[24] W. R. A. Md-w, “广州忆凝云电子科技有限公司 DC / DC Converters 产品特点 :,” *2019 IEEE Electr. Sh. Technol. Symp.*, pp. 1–3, 2019.

[25] S. Output and E. Type, “IRM-03 IRM-03,” pp. 1–4, 2017.

[26] T. R. Angle and G. Lock, “3M ™ Mini D Ribbon (MDR) Connectors repetitive plugging 3M ™ Mini D Ribbon (MDR) Connectors,” 2007.

[27] W. INSTRUMENTS, “MANUAL DE APLICACIÓN DE ENCODERS Encoders Ópticos,” p. 21, 2010, [Online]. Disponible en: https://www.acomee.com.mx/ENCODERS.pdf.

Paralelamente se han consultado los apuntes de las asignaturas: “Sistemas electrónicos” y “Tecnología electrónica” para la realización de los circuitos de la interfaz electrónica.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

DISEÑO DE UN BANCO DE ENSAYOS DE MÁQUINAS DE IMANES PERMANENTES DE TÉCNICA SENOIDAL (SERVOACCIONAMIENTO PMSM BRUSHLESS AC) DE 200W CON INTERFAZ HTML BASADA EN ESP32

PRESUPUESTO

Autor: Ismael Ferrer Vayá

Tutor: Javier Andrés Martínez Román

Universidad politécnica de Valencia

Curso académico: 2022 – 2023

Índice del presupuesto

1. Introducción 1
2. Precios de la mano de obra, materiales y equipo 1
3. Precios descompuestos 3
4. Precios unitarios 6
5. Presupuesto de ejecución material 7
6. Presupuesto de ejecución por contrata y licitación 7

1. Introducción

El siguiente presupuesto indica el coste económico de realización del proyecto "Diseño de un banco de ensayos de máquinas de imanes permanentes de técnica senoidal (servoaccionamiento PMSM Brushless AC) de 200W con interfaz HTML basada en ESP32". Con el objetivo de aumentar la exactitud, la construcción y diseño del proyecto se ha dividido en sus partes constituyentes, llamadas unidades de obra, que cuantifican de forma más precisa los materiales, tiempo y costo de cada uno de los trabajos diferenciados que son realizados.

En el presupuesto se presentan:

- Precios de la mano de obra, materiales y equipo
- Precios descompuestos de cada unidad de obra
- Precios unitarios
- Presupuesto de ejecución material, por contrata y licitación

En base a ello se pretende definir la suma económica requerida para la realización del proyecto.

2. Precios de la mano de obra, materiales y equipo

Se muestra en el siguiente cuadro de precios el costo de la mano de obra, los materiales y el equipo empleados.

Concepto	Descripción	Unidad	Precio/ud
Mano de obra			
Ingeniero junior	Hora de trabajo de ingeniero junior. Principal encargado del desarrollo del proyecto, su diseño y su construcción	h	30€
Supervisor del proyecto	Hora de trabajo de ingeniero, de especialidad en la disciplina eléctrica. Encargado de la supervisión general del proyecto.	h	60€
Materiales			
ESP32	Microcontrolador ESP32 modelo DO IT DEVKIT 1.	ud	10€
Placa de prototipos	Placa de matriz de 61 x 38 orificios sobre la que se realiza el montaje de la interfaz electrónica.	ud	3,81€
Integrado AO	Circuito integrado modelo TL082 que contiene 2 amplificadores operacionales.	ud	0,818€
Fuente 5V	Fuente de alimentación modelo IRM-03-5 que adapta la tensión de red a 5V de corriente continua.	ud	8,28€
Fuente 24V	Fuente de alimentación modelo IRM-03-24 que adapta la tensión de red a 24V de corriente continua.	ud	8,11€
Convertidor DC/DC	Convertidor de continua modelo TMA0515D que convierte 5V en tensiones de ±15V.	ud	5,44€
Zócalo AO	Zócalo para la conexión del circuito integrado del amplificador operacional	ud	0,169€
Conector macho	Peine lineal para realizar la conexión del bus paralelo.	ud	0,22€
Zócalo hembra	Zócalos lineales para realizar la conexión del ESP32 con la placa y la conexión del bus paralelo.	ud	0,23€

Conector paralelo	Conector normalizado de puerto paralelo de 50 pines, paso de 1,27mm modelo 10150-6000EL.	ud	6,58€
Bus paralelo	Bus paralelo de 25 pines de 1,27mm de paso, con una longitud de 25cm.	ud	0,682€
Resistencia 2K2Ω	Resistencia de 2200 Ω	ud	0,2€
Resistencia 6K8Ω	Resistencia de 6800 Ω	ud	0,21€
Resistencia 10KΩ	Resistencia de 10000 Ω	ud	0,204€
Resistencia 1KΩ	Resistencia de 1000 Ω	ud	0,16€
Resistencia 4K7Ω	Resistencia de 4700 Ω	ud	0,179€
Resistencia 22KΩ	Resistencia de 22000 Ω	ud	0,134€
Resistencia variable 10KΩ	Resistencia variable de 10000 Ω	ud	0,63€
Condensador 10µF	Condensador cerámico de capacidad 10µF	ud	0,744€
Condensador 2,2µF	Condensador cerámico de capacidad 2,2µF	ud	0,332€
Transistor	Transistor de potencia modelo BC337	ud	0,254€
Servopack	Servopack modelo SGDH-02AE-OY de alimentación monofásica a 230V.	ud	295€
Servomotor	Servomotor modelo SGMAH-02AAA61D-OY de 200 W, con alimentación trifásica desde el servopack y con encoder incorporado.	ud	570€
Router Wifi	Enrutador Wifi que permite la conexión inalámbrica de control del sistema.	ud	20€
Elementos de soldadura	Cableado de conexión e hilo de estaño empleado en el proceso de soldadura considerando mermas. Se establece un coste arbitrario al conjunto al tener difícil cuantificación.	ud	20€
Cableado de potencia.	Elementos como los cables de conexiones entre servomotor y servopack, regletas o conexiones industriales. Al ser de difícil cuantificación se incluirán como costes porcentuales del total de la unidad.	%	-
Equipo			
PC	Uso de ordenador personal para diferentes aspectos tanto de programación como de ofimática.	mes	10€
Arduino IDE	Empleo del entorno de desarrollo integrado de Arduino para la codificación del servidor.	mes	0€
Visual Studio	Empleo del editor de textos Visual Studio Code para la programación de la interfaz HTML.	mes	0€
Equipo de soldadura	Empleo de soldador, desoldador, soporte para soldadura de circuitos impresos, y demás herramientas necesarias para la construcción de la interfaz electrónica.	h	2,4€

Los precios de los componentes electrónicos son los indicados por el distribuidor “RS PRO”, el cual ha sido el suministrador principal durante todo el proyecto. El precio del uso del PC se ha supuesto un coste de adquisición de 600€ con un período de amortización de 5 años, quedando su

coste mensual en 10€. El coste del equipo de soldadura se ha obtenido teniendo en cuenta su precio de aproximado adquisición y el desgaste provocado durante su uso, mientras que el coste del material de soldadura se ha estimado en aproximadamente 20€.

3. Precios descompuestos

Se muestra a continuación la descripción de cada unidad de obra, así como todos los elementos implicados en cada una de ellas y su coste resultante.

Unidad de obra N.º 1: Programación interfaz HTML

Unidad de obra	**Descripción**			**Unidad**
Programación interfaz HTML	Programación de la interfaz HTML que controla el sitio web. Incluye la codificación del código, la búsqueda de información acerca de los lenguajes empleados, el diseño gráfico de la página y su comprobación.			Ud
Costes directos				
Concepto	**Cantidad**	**Unidad**	**Precio/ud (€)**	**Importe (€)**
Ingeniero junior	70	h	30	2100
Supervisor del proyecto	3	h	60	180
PC	1,5	mes	10	15
Visual Studio	1,5	mes	0	0
			Coste total unidad de obra:	**2295**

Unidad de obra N.º 2: Programación servidor

Unidad de obra	**Descripción**			**Unidad**
Programación servidor	Programación del servidor codificado en el lenguaje de programación Arduino. Incluye la búsqueda de información acerca de redes, servicios inalámbricos y el diseño de sus características además de la escritura del código.			Ud
Costes directos				
Concepto	**Cantidad**	**Unidad**	**Precio/ud (€)**	**Importe (€)**
Ingeniero junior	90	h	30	2700
Supervisor del proyecto	6	h	60	360
PC	2	mes	10	20
Arduino IDE	2	mes	0	0
			Coste total unidad de obra:	**3080**

Unidad de obra N.º 3: Construcción interfaz electrónica

Unidad de obra	**Descripción**			**Unidad**
Construcción interfaz electrónica	Montaje de la placa base que forma la interfaz electrónica. Incluye soldadura de todos los componentes electrónicos incluyendo al microcontrolador, así como su cableado, su alimentación, su diseño y los cálculos realizados para asegurar su funcionamiento.			Ud
Costes directos				
Concepto	**Cantidad**	**Unidad**	**Precio/ud (€)**	**Importe (€)**
Ingeniero junior	60	h	30	1800
Supervisor del proyecto	6	h	60	360
PC	1	mes	10	10
ESP32	1	ud	10	10
Placa de prototipos	1	ud	3,81	3,81
Integrado AO	1	ud	0,818	0,818
Fuente 5V	1	ud	8,28	8,28
Fuente 24V	1	ud	8,11	8,11
Convertidor DC/DC	1	ud	5,44	5,44
Zócalo AO	1	ud	0,169	0,169
Conector macho	25	ud	0,22	5,5
Zócalo hembra	55	ud	0,23	12,65
Conector paralelo	1	ud	6,58	6,58
Bus paralelo	2	ud	0,682	1,364
Resistencia 2K2Ω	4	ud	0,2	0,8
Resistencia 6K8Ω	8	ud	0,21	1,68
Resistencia 10KΩ	3	ud	0,204	0,612
Resistencia 1KΩ	6	ud	0,16	0,96
Resistencia 4K7Ω	3	ud	0,179	0,537
Resistencia 22KΩ	3	ud	0,134	0,402
Resistencia variable 10KΩ	1	ud	0,63	0,63
Condensador 10µF	1	ud	0,744	0,744
Condensador 2,2µF	1	ud	0,332	0,332
Transistor	3	ud	0,254	0,762
Elementos de soldadura	1	ud	20	20
Equipo de soldadura	25	h	2,4	60
			Coste total unidad de obra:	**2320,18**

Unidad de obra N.º 4: Montaje del servomotor

Unidad de obra	Descripción			Unidad
Montaje del servomotor	Montaje y puesta en marcha del conjunto formado por interfaz electrónica, servopack y servomotor. Incluye los costos de adquisición de los elementos de potencia y del enrutador Wifi encargado de generar la red de conexión inalámbrica.			Ud
Costes directos				
Concepto	**Cantidad**	**Unidad**	**Precio/ud (€)**	**Importe (€)**
Ingeniero junior	5	h	30	150
Supervisor del proyecto	5	h	60	300
Router Wifi	1	ud	20	20
Servopack	1	ud	295	295
Servomotor	1	ud	570	570
			Total coste directo:	1335
Cableado de potencia.	2	%	1335	26,7
			Coste total unidad de obra:	**1361,7**

Unidad de obra N.º 5: Redacción de la memoria

Unidad de obra	Descripción			Unidad
Redacción de la memoria	Escritura de la memoria y cálculo del presupuesto, incluyendo la búsqueda de información, la edición de imágenes y la creación de esquemas.			Ud
Costes directos				
Concepto	**Cantidad**	**Unidad**	**Precio/ud (€)**	**Importe (€)**
Ingeniero junior	70	h	30	2100
Supervisor del proyecto	10	h	60	600
PC	1	mes	10	10
			Coste total unidad de obra:	**2710**

4. Precios unitarios

Se muestra a continuación la recopilación de las unidades de obra empleadas en la ejecución de este proyecto, con su descripción y su precio unitario.

Unidad de obra	Descripción	Precio unitario (€)
Programación interfaz HTML	Programación de la interfaz HTML que controla el sitio web. Incluye la codificación del código, la búsqueda de información acerca de los lenguajes empleados, el diseño gráfico de la página y su comprobación.	**2295**
Programación servidor	Programación del servidor codificado en el lenguaje de programación Arduino. Incluye la búsqueda de información acerca de redes, servicios inalámbricos y el diseño de sus características además de la escritura del código.	**3080**
Construcción interfaz electrónica	Montaje de la placa base que forma la interfaz electrónica. Incluye soldadura de todos los componentes electrónicos incluyendo al microcontrolador, así como su cableado, su alimentación, su diseño y los cálculos realizados para asegurar su funcionamiento.	**2320,18**
Montaje del servomotor	Montaje y puesta en marcha del conjunto formado por interfaz electrónica, servopack y servomotor. Incluye los costos de adquisición de los elementos de potencia y del enrutador Wifi encargado de generar la red de conexión inalámbrica.	**1361,7**
Redacción de documentos	Escritura de la memoria y presupuesto, incluyendo la búsqueda de información, la edición de imágenes y la creación de esquemas.	**2710**

5. Presupuesto de ejecución material

Se muestra a continuación el costo material de realizar el proyecto, como suma de las diferentes unidades de obra realizadas.

Unidad de obra	Cantidad	Unidad	Precio/ud (€)	Importe (€)
Programación interfaz HTML	1	ud	2295	2295
Programación servidor	1	ud	3080	3080
Construcción interfaz electrónica	1	ud	2320,18	2320,18
Montaje del servomotor	1	ud	1361,7	1361,7
Redacción de documentos	1	ud	2710	2710
Presupuesto de ejecución material (PEM):				**11766,88**

El presupuesto de ejecución material asciende a la cantidad de: ONCE MIL SETECIENTOS SESENTA Y SEIS EUROS CON OCHENTA Y OCHO CENTIMOS

6. Presupuesto de ejecución por contrata y licitación

Se muestran a continuación el presupuesto de ejecución por contrata y el presupuesto base de licitación. En caso de quererse realizar el proyecto a terceros.

Concepto	Importe (€)
Presupuesto de ejecución material (PEM)	11766,88
Gastos generales (6% del PEM)	706,02
Beneficio industrial (6% del PEM)	706,02
Presupuesto de ejecución por contrata (PEC)	**13178,92**
IVA (21% del PEC)	2767,57
Presupuesto base de licitación	**15946,49**

El presente presupuesto asciende a la cantidad de: QUINCE MIL NOVECIENTOS CUARENTA Y SEIS EUROS CON CUARENTA Y NUEVE CENTIMOS

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

DISEÑO DE UN BANCO DE ENSAYOS DE MÁQUINAS DE IMANES PERMANENTES DE TÉCNICA SENOIDAL (SERVOACCIONAMIENTO PMSM BRUSHLESS AC) DE 200W CON INTERFAZ HTML BASADA EN ESP32

ANEXOS DE LA MEMORIA

Autor: Ismael Ferrer Vayá

Tutor: Javier Andrés Martínez Román

Universidad politécnica de Valencia

Curso académico: 2022 – 2023

Índice de los anexos

1. ANÁLISIS DEL CÓDIGO EN HTML 1
1.1 Estructura global en HTML 1
1.2 Apartados individuales en HTML 3
1.2.1 Título de la página 3
1.2.2 Entradas digitales 3
1.2.3 Entradas analógicas 4
1.2.4 Salidas del encoder 5
1.2.5 Salidas digitales 6
1.2.6 Comunicación con el servidor 7
1.2.7 Autor y tutor 7
1.2.8 Escuela y departamento 7
1.3 Estilos del sitio web empleando CSS 7
1.3.1 Estilos del encabezado <header> 7
1.3.2 Estilos del cuerpo <body> 8
1.3.3 Estilos del pie de página <footer> 9
1.4 Control y comunicación del sitio web con JavaScript 10
1.4.1 Declaración de variables 10
1.4.2 Envío de datos al servidor 10
1.4.3 Recepción de datos del servidor 12
1.4.4 Función cambioEscrito() 13
2. CÓDIGO HTML 15
3. ANÁLISIS DEL CÓDIGO EN ARDUINO 22
3.1 Inclusión de las librerías y definición de variables globales 22
3.2 Selección de red y dirección IP estática 23
3.3 Variable del código HTML 24
3.4 Definición de los puertos y de la función timer 24
3.5 Función setup() 24
3.5.1 Definición de entradas y salidas 25
3.5.2 Conexión a la red Wifi 25
3.5.3 Inicialización de servidor y Websocket 26
3.5.4 Temporización de la función de envío de datos 27
3.6 Recepción de datos del usuario 27
3.6.1 Bucle de comunicación 27
3.6.2 Codificación de la respuesta a eventos 28
3.7 Determinación velocidad y sentido con el encoder 30

3.7.1 Funcionamiento del encoder 30
3.7.2 Rutinas de interrupción 31
3.7.3 Función “control_interrupciones()” 32
3.8 Envío de datos al usuario *32*
3.8.1 Creación del archivo JSON 32
3.8.2 Recopilación de información 32
3.8.3 Protocolo de alarma 33
3.8.4 Envío del archivo JSON 33
4. CÓDIGO ARDUINO 34

1. ANÁLISIS DEL CÓDIGO EN HTML

A continuación, se explica con detalle las líneas de código más destacables para la explicación del sitio web. Se debe recordar que la mayoría de las etiquetas requieren a la respectiva de cierre, sin embargo, se hará referencia a la primera para referirse a ambas con el objetivo de evitar repetir esto a lo largo del análisis. El código aparece sobre un fondo negro para diferenciarlo del texto explicativo, con la fuente característica de Visual Studio.

El código completo se encuentra en el capítulo siguiente de los anexos.

1.1 Estructura global en HTML

Todo código HTML debe iniciar con la etiqueta <!DOCTYPE html>, pues le indica al navegador que se emplea dicho lenguaje facilitando la carga rápida del sitio web. Además, todo debe estar incluido entre ésta y la correspondiente de cierre. Después del inicio del código se incluye la etiqueta <head> que se emplea para dar cierta información acerca del código, así como para definir el estilo gráfico de la página con la etiqueta <style>, donde se escribirá con lenguaje CSS.

Fuera de <head> se codifican el resto de las etiquetas y la estructura del sitio web que aparece por pantalla. La organización se ha llevado a cabo con el uso de 3 etiquetas muy empleadas en diseño web que sirven para ordenarla verticalmente, estas son: <header> (Encabezado), <body> (Cuerpo) y <footer> (Pie de página). Se ubican después del cierre de </head>. A cada una de ellas se le ha asignado una clase, la cual será su identificador en CSS para modificar su diseño.

Dentro de estas etiquetas se definen tablas, para la organización horizontal. Para ello se emplea la etiqueta <table> y dentro de ella las que definen nuevas filas <tr> y nuevas columnas <th>. Dentro de ellas se definirán entradas, salidas y el texto correspondiente. A las etiquetas de columna también se les ha asignado una clase para identificarlas en CSS.

Normalmente al final se incluye la etiqueta <script> que indica el inicio de la codificación con JavaScript. A modo de esquema:

```html
<!DOCTYPE html>
<head>
    <style>
        /*Código en CSS*/
    </style>
</head>
<header class="encabezado">
    <!--Título de la página-->
</header>
<body class="cuerpo">
    <table>
        <tr>
            <th class="entradaDigital">
                <!--Entradas digitales-->
            </th>
            <th class="entradaAnalogica">
                <!--Entradas analógicas-->
            </th>
            <th class="separacion">
                <!--Separación para incluir una línea entre entradas y salidas-->
            </th>
            <th class="salidaEncoder">
                <!--Salida encoder-->
            </th>
            <th class="salidaDigital">
                <!--Salidas digitales-->
            </th>
        </tr>
    </table>
</body>
<footer class="piepagina">
    <table>
        <tr>
            <th class="comunicacion">
                <!--Comunicación servidor-->
            </th>
            <th class="TFG">
                <!--Texto autor y tutor-->
            </th>
            <th class="escuela">
                <!--Texto escuela y departamento-->
            </th>
        </tr>
    </table>
</footer>
<script>
    /*Código en JavaScript*/
</script>
</html>
```

Se muestra el uso de las tablas y de las etiquetas de encabezado, cuerpo y pie de página en rojo sobre la página definitiva, no obstante, solo con este código no es suficiente para obtener este resultado, pues se requiere también CSS y el uso de la etiqueta , la cual se ha empleado incluyendo todas las etiquetas de <header>, <body> y <footer> para que aparezcan centradas en la página web.

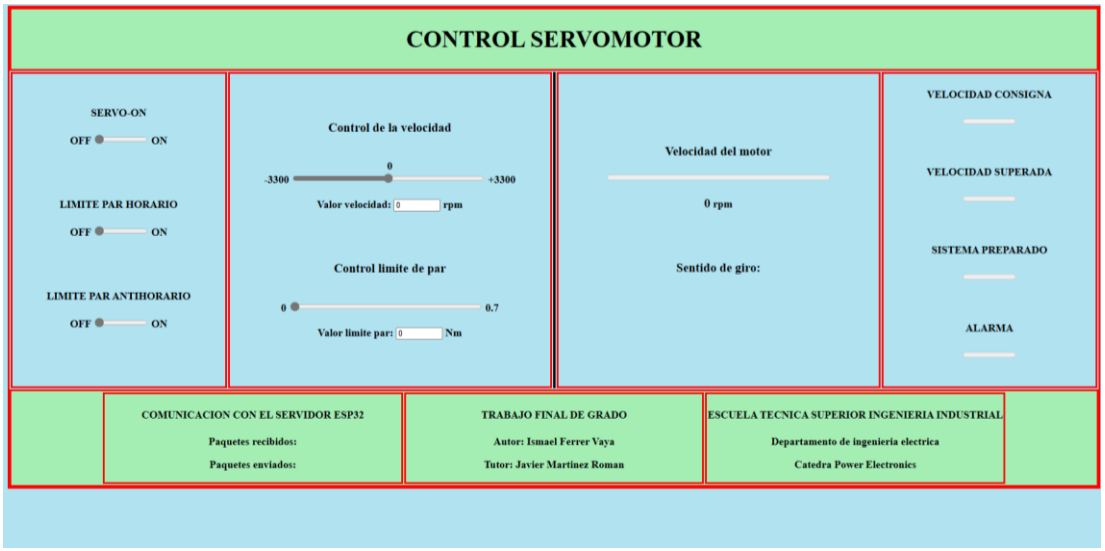

Ilustración 1 de los anexos. Estructura HTML. Fuente propia. Indicación de los diferentes elementos que componen el sitio web encuadrados en bordes rojos.

1.2 Apartados individuales en HTML

Se consideran apartados individuales a aquellos que corresponden a entradas, salidas, títulos, etc. Cuya aparición se programa mediante HTML. Su código aparecerá en el lugar de los comentarios con el mismo nombre del esquema en el código del apartado anterior.

1.2.1 Título de la página

Se imprime en la parte superior de la página web e indica su finalidad. Al ser el título principal se ha empleado la etiqueta de <h1>, que es la que mayor tamaño y vistosidad da de forma base al texto. Se le ha aplicado un identificador por id para aplicarle estilos.

```
<h1 id="titulo">CONTROL SERVOMOTOR</h1>
```

1.2.2 Entradas digitales

El funcionamiento de las tres entradas es similar, pues las tres se realizan con la etiqueta <input> que permite interacción por parte del usuario. Alrededor de la misma se han escrito las palabras OFF y ON para indicar su valor. La etiqueta se ha modificado con los siguientes atributos:

- onchange= "enviarDato(this)": Al variar el valor de la entrada se ejecutará la función de JavaScript “enviarDato()” y se le enviará el valor seleccionado.
- id: Nombra a la etiqueta con un identificador por id.

- type="range": Hace que la etiqueta input aparezca por pantalla y se interactúe con ella como una barra deslizante, pretendiendo simular un interruptor.
- min="0" max="1": Establece los valores máximos y mínimos posibles para esta entrada. En este caso al ser digitales es 0 o 1.
- value="0": Indica el valor por defecto que tendrá la entrada al cargar la página.
- step ="1": Indica el intervalo de cambio entre posiciones. En este caso solo de 1.

A parte de estas etiquetas también aparecen las de texto para indicar cada entrada diferente, empleando las etiquetas <h4>. Para organizar verticalmente y dar espacio entre las entradas se ha empleado la etiqueta de salto de línea
.

```
<h4>SERVO-ON</h4>
OFF<input onchange= "enviarDato(this)" id="control_S_ON" type="range" min="0" max="1" value="0" step ="1">ON
<br>
<br>
<br>
<br>
<h4>LIMITE PAR HORARIO</h4>
OFF<input onchange= "enviarDato(this)" id="control_P_CL" type="range" min="0" max="1" value="0" step ="1">ON
<br>
<br>
<br>
<br>
<h4>LIMITE PAR ANTIHORARIO</h4>
OFF<input onchange= "enviarDato(this)" id="control_N_CL" type="range" min="0" max="1" value="0" step ="1">ON
<br>
<br>
<br>
<br>
```

1.2.3 Entradas analógicas

La estructura es muy similar a la de las entradas digitales, pues se vuelven a emplear las etiquetas de títulos en este caso <h3>, las de <input> y las de salto de línea
. Sin embargo, como el valor de cada entrada analógica se puede variar desde la barra deslizante o desde el cuadro de texto se requieren dos <input> por entrada con atributos diferentes.

Con respecto a la barra deslizante los atributos empleados son los mismo que en el apartado de entradas digitales, sin embargo, cambian ciertos valores para ajustarse a la operación. Al haber empleado 12 bits en la codificación en el servidor, las señales eléctricas se pueden codificar como 4096 valores diferentes, del 0 al 4095, siendo estos valores los que el servidor reconoce para generar la señal. Por ello el usuario al deslizar la barra está seleccionando alguno de estos valores y siempre con variaciones unitarias entre ellos, enviando alguno de estos valores permitidos al servidor. Por ello el valor máximo de la entrada es 4095 y su variación es unitaria (pues es el valor por defecto si no se especifica lo contrario). En el caso de la velocidad, como se codifica la negativa y la positiva, el cero está en el centro del intervalo de valores y por ello se inicia al cargar la página en 2048. Alrededor de la respectiva etiqueta se expresa el valor de la velocidad o par que representa la barra entre las posiciones límite para indicar al usuario su equivalencia.

Para el cuadro de texto se deben emplear algunos atributos diferentes sobre la etiqueta de <input> teniendo los ya comentados el mismo uso. Se debe destacar que en este caso el usuario introduce un valor de la velocidad o del par que desea, por lo que el propio sitio web debe traducir este dato al valor aceptable más próximo, entre 0 y 4095, para poder enviarlo al servidor. Además, debe mostrar en este mismo cuadro el valor de velocidad o par solicitado si el cambio se ha realizado mediante la barra deslizante. Los atributos distintos son:

- onchange=”cambioEscrito(this)”: Al efectuarse un cambio se ejecuta la función de JavaScript cambioEscrito(), enviándole el valor escrito por el usuario.
- type="text": Para indicar que el tipo de entrada es un cuadro de texto.
- size="6": Indica la cantidad de caracteres permitidos en la entrada.

Hay que destacar que en los rangos de máximo o mínimo se impondrán los valores de velocidad o de par pues son las magnitudes a las que se referirá el usuario. Y como también muestran el valor solicitado a la izquierda del cuadro se escribirá la variable representada y a su derecha la unidad de medida.

```
<h3>Control de la velocidad</h3>
<br>
0
<br>
-3300 <input onchange= "enviarDato(this)" id="control_V_REF" type="range" min="0" max="4095"
value="2048"> +3300
<br>
<br>
Valor velocidad: <input onchange="cambioEscrito(this)" id="control_V_REF_numero" type="text"
min="-3300" max="3300" value="0" size="6" > rpm
<br>
<br>
<br>
<br>
<h3>Control limite de par</h3>
<br>
0 <input onchange= "enviarDato(this)" id="control_T_REF" type="range" min="0" max="4095"
value="0"> 0.7
<br>
<br>
Valor limite par: <input onchange="cambioEscrito(this)" id="control_T_REF_numero" type="text"
min="0" max="0.700527" value="0" size="6"> Nm
<br>
<br>
```

1.2.4 Salidas del encoder

Pretenden mostrar por pantalla la velocidad y el sentido de giro del servomotor. Se emplean igualmente etiquetas de título <h3> y de salto de línea
 para organizar la información y en algunos casos para mostrarla por pantalla.

Para la indicación de la velocidad se emplea una nueva etiqueta llamada <meter> que imprime una barra de progreso por pantalla más o menos llena en función del valor recibido a modo de

velocímetro. Los atributos aplicables a dicha etiqueta son los mismos que en apartados anteriores por lo que no volverán a ser explicados. Debajo de la barra de progreso se imprime el valor de la velocidad en revoluciones por minuto. Para ello se emplea el atributo id en una etiqueta de título como es <h3>, pues ello permitirá al código en JavaScript enviar a dicha etiqueta el valor que aparecerá por pantalla. Esto mismo se utiliza para indicar el sentido de giro, pero en este caso JavaScript enviará una cadena de caracteres en vez de un valor numérico. En las dos etiqueta <h3> con identificador para JavaScript se aplica el siguiente atributo: style="display: inline-block;", que se emplea para considerar la etiqueta como un elemento de línea y no alterar la disposición de los elementos cercanos al variar su valor representado.

```
<h3>Velocidad del motor</h3>
<meter value="0" min="0" max="3300" id="Vel_encoder_meter"></meter>
<br>
<h3 id="Vel_encoder_value" style="display: inline-block;">0</h3> rpm
<br>
<br>
<br>
<h3>Sentido de giro: </h3>
<br>
<h3 id="sentido" style="display: inline-block;"></h3>
<br>
```

1.2.5 Salidas digitales

Representan a una salida de control digital del servopack y sólo representan su estado activo o inactivo. Para simular un indicador luminoso se ha empleado la etiqueta meter con solo dos estados, 0 o 1, que en las disposiciones de barra vacía o llena simulan una luz que se apaga o se enciende en función del valor recibido. Por ello su valor mínimo es 0 y su máximo 1.

```
<h4>VELOCIDAD CONSIGNA</h4>
<meter value="0" min="0" max="1" id="V_CMP_meter"> </meter>
<br>
<br>
<br>
<h4>VELOCIDAD SUPERADA</h4>
<meter value="0" min="0" max="1" id="TGON_meter"> </meter>
<br>
<br>
<br>
<h4>SISTEMA PREPARADO</h4>
<meter value="0" min="0" max="1" id="S_RDY_meter"> </meter>
<br>
<br>
<br>
<h4>ALARMA</h4>
<meter value="0" min="0" max="1" id="ALM_meter"> </meter>
<br>
<br>
<br>
```

1.2.6 Comunicación con el servidor

En estas líneas de código se pretende indicar la cantidad de paquetes de información intercambiados con el servidor, cuyo conteo se realiza en el código de JavaScript. Para presentar y organizar la información se emplean las etiquetas de título <h4> y las de párrafos <p>, con el menor tamaño de fuente por defecto. Dentro de estas últimas se escribe la indicación del valor medido, y para que aparezca este valor se emplea la etiqueta dentro de la etiqueta <p>, que permite imprimir el valor deseado desde JavaScript con el atributo id sin alterar el texto fuera de ella.

```
<h4>COMUNICACION CON EL SERVIDOR ESP32</h4>
<p>Paquetes recibidos: <span id="recibidos"> </span></p>
<p>Paquetes enviados: <span id="enviados"> </span></p>
```

1.2.7 Autor y tutor

Imprime el texto correspondiente empleando etiquetas de títulos y párrafos.

```
<h4>TRABAJO FINAL DE GRADO</h4>
<p>Autor: Ismael Ferrer Vaya</p>
<p>Tutor: Javier Martinez Roman</p>
```

1.2.8 Escuela y departamento

Imprime el texto correspondiente empleando etiquetas de títulos y párrafos.

```
<h4>ESCUELA TECNICA SUPERIOR INGENIERIA INDUSTRIAL</h4>
<p>Departamento de ingenieria electrica</p>
<p>Catedra Power Electronics</p>
```

1.3 Estilos del sitio web empleando CSS

El principal objetivo de CSS en este trabajo es ordenar los elementos que aparecen por pantalla, ajustar ciertos tamaños y aplicar colores de fondo y bordes. La explicación del código se dividirá en las tres partes en las que se ha estructurado la visualización por pantalla del sitio web: encabezado, cuerpo y pie de página. Para referirse a elementos identificados con “class” se escribe un punto “.” antes del nombre del identificador y los identificados con “id” una almohadilla “#”.

1.3.1 Estilos del encabezado <header>

El encabezado está formado por dos elementos principales, la etiqueta de <header> en sí misma, la cual se identifica con el atributo class="encabezado" y el título <h1>, identificado con el atributo id="titulo". Al encabezado se le ha agregado un color de fondo verdoso y un borde negro

de 3 píxeles (px) de grosor para destacarlo del cuerpo de la página. Al título se le ha aplicado un tamaño de fuente de 45 px para dotarlo de un tamaño muy superior al resto.

```
.encabezado{
  background:#a4eeb4;
  border:3px solid #000000;
}
#titulo{
  font-size: 45px;
}
```

1.3.2 Estilos del cuerpo <body>

En este lugar aparecen todas las diferentes entradas y salidas de control, así como la línea separadora entre ambos tipos de señales. La etiqueta del conjunto es <body> identificada con el atributo class= "cuerpo", dentro de ella hay una tabla y en cada una de sus celdas aparecen las respectivas a cada tipo de entrada o salida, identificada cada una de ellas con sus atributos class correspondientes, y finalmente algunas etiquetas de <input> o texto han sido referenciados con el atributo id.

Al cuerpo se le ha asignado un color de fondo azulado y además se ha seleccionado un tamaño de fuente grande “large” que se aplica escalado a cada tipo de texto dentro del cuerpo.

Posteriormente a cada tipo de entrada o salida que se corresponde con una determinada celda de la tabla se le han aplicado 2 propiedades, la primera es (position:relative;) que asegura que este elemento siempre estará en la misma posición relativa que el elemento que lo contiene y la segunda es (width: XX%;) que regula la anchura del elemento porcentualmente a la anchura de su elemento contenedor. En ambos casos el elemento que lo contiene es la propia tabla formada dentro de <body>. Lo que se pretende es ajustar el tamaño correspondiente a cada entrada o salida para permitir una operación adecuada y una buena visibilidad. Por ejemplo las entradas analógicas o los datos del encoder requieren un mayor tamaño, por ello las celdas que los contienen ocupan cada una un 30% del total de la página, mientras que las correspondientes de las entradas y salidas digitales ocupan un 20% cada una.

En las etiquetas <input> modificadas se ha hecho uso otra vez de la propiedad (width: XX%;), en este caso para que el elemento que representa esta etiqueta tenga una anchura de tamaño relativo a la celda en la que se encuentra. En el caso de entradas analógicas o del velocímetro del encoder se ha aumentado su tamaño haciendo que ocupe el 60 o 70 % de la celda, sin embargo, para entradas y salidas digitales estas solo ocupan el 20 o 25 %.

Se expone el código correspondiente al cuerpo y a las entradas digitales, las demás entradas y salidas están codificadas de forma similar:

```
.cuerpo{
  background: #b0e2f0;
  font-size: large;
}
.entradaDigital{
  position:relative;
  width: 20%;
}
```

```
#control_S_ON, #control_P_CL, #control_N_CL{
  width: 25%;
}
```

Para incluir una línea que separe las entradas de las salidas se ha empleado otra celda en medio, identificada mediante el atributo class=”separación”, a la cual también se le ha aplicado la propiedad de ubicación relativa y se le ha dotado de un fondo de color negro entre unos bordes negros de tamaño 3 pixeles en total.

```
.separacion{
  position:relative;
  background: #000000;
  border:1.5px solid #000000;
}
```

1.3.3 Estilos del pie de página <footer>

Formada por la propia etiqueta <footer> , la cual contiene una tabla y en cada una de sus celdas aparecen el apartado de comunicación con el servidor class="comunicacion", el texto con el autor y tutor del trabajo class="TFG" y el texto con la escuela y departamento class="escuela".

Al pie de página se le ha aplicado un fondo y borde de iguales características que el encabezado para diferenciarlo del cuerpo y darle cohesión a la página, por otro lado empleando las propiedades (position:relative;) y (width: 33.333%;), para que se reparta igual entre tres, se ha conseguido que las celdas de la tabla, con lo que contienen, ocupen de forma simétrica, ordenada y equitativa el espacio disponible.

Se expone el estilo correspondiente al pie de página y a una sola celda, pues las demás son exactamente iguales cambiando su identificador.

```
.piepagina{
  background:#a4eeb4;
  border:3px solid #000000;
}
.comunicacion{
  position: relative;
  width: 33.333%;
}
```

1.4 Control y comunicación del sitio web con JavaScript

Todos los procesos que lleva a cabo la página web de envío y recepción de datos, así como ciertos aspectos de control interno están codificados en este lenguaje. La explicación del código se estructura alrededor de la definición de variables y el uso de las tres funciones que regulan el funcionamiento y se realizará a medida que se expone el código.

1.4.1 Declaración de variables

Se han colocado al inicio del código, además se definen todas como variables globales para que todas las funciones implicadas puedan hacer uso de estas y no tengan que ser declaradas cada vez con el objetivo de ahorrar tiempo.

La más importante es la que inicializa al Websocket, pues JavaScript lo considera como una variable más a la cual se le ha dado el nombre websocket. Para crearlo se aplica la siguiente línea de código, que considera la dirección IP del hosting del servidor, ubicado en el puerto 81 como se comenta en el capítulo del servidor.

```
var websocket = new WebSocket('ws://'+location.hostname+':81/');
```

Las demás variables se han declarado de una forma más sencilla, y solo se han inicializado al valor 0 las que cuentan la cantidad de paquetes recibidos y enviados y la variable de control cambio.

```
    var enviados=0;
    var recibidos=0;
    var valor_S_ON
    var valor_P_CL;
    var valor_N_CL;
    var valor_V_REF;
    var valor_T_REF;
    var valor_V_REF_numero;
    var valor_T_REF_numero;
    var cambio=0;
```

Las variables que llevan valor en su nombre se refieren a entradas que serán leídas y enviadas, las que tienen al final la palabra numero son las que almacenan la información de las entradas de cuadro de texto y la variable cambio es una variable de control interna entre las funciones.

1.4.2 Envío de datos al servidor

Se trata de una función que se ejecuta cada vez que se produce un cambio en las etiquetas <input> que tienen el atributo onchange= "enviarDato(this)", pues este es el nombre que tiene la función, por lo que solo se trabaja con entradas. Se hace notar que este atributo solo lo poseen las barras deslizantes, pero no los cuadros de texto. El propósito de esta función es realizar una lectura a todas las entradas, independientemente de cuál haya sufrido un cambio, en algunos casos realizar un cambio sobre las etiquetas <input> de cuadros de texto, almacenar estas entradas en un archivo JSON y finalmente enviarlo a través de Websocket al servidor. Se muestra a continuación, por partes y explicado, el código.

```
function enviarDato(element) {
```

Declaración de la función enviarDato, está preparada para recibir el valor recibido “element”, aunque ella internamente volverá a leerlo.

```
valor_V_REF = document.getElementById("control_V_REF").value;
if(cambio!=1){
  valor_V_REF_numero=(valor_V_REF-4096/2)*1.611328125;
  valor_V_REF_numero=valor_V_REF_numero.toFixed(2);
  document.getElementById("control_V_REF_numero").value = valor_V_REF_numero;
}

valor_T_REF = document.getElementById("control_T_REF").value;
if(cambio!=1){
 valor_T_REF_numero=valor_T_REF*0.000171069;
 valor_T_REF_numero=valor_T_REF_numero.toFixed(5);
 document.getElementById("control_T_REF_numero").value = valor_T_REF_numero;
}
cambio=0;
```

Estas líneas de código realizan la lectura de la barra deslizante que indica el control de velocidad “valor_V_REF” o par “valor_T_REF”, mediante la función “getElementById()” a la cual se le especifica que se encuentra en el documento HTML (document.) y que se hace referencia a su atributo de valor numérico (.value). Esta función permite el cambio de algún atributo o incluso el contenido de las etiquetas de HTML empleando su identificador id correspondiente.

En caso de que la variable cambio tenga un valor diferente de 1, lo cual significa que la variable ha sido modificada desde la barra deslizante y no desde el cuadro de texto, se debe modificar el valor que aparece en el cuadro de texto por el que ahora representa la barra deslizante.

En otra variable se traduce los valores de la barra deslizante entre 0 y 4095 a sus respectivos de velocidad o par empleando la proporcionalidad con cada una de estas magnitudes (se ve con más detalle en el apartado de precisión de las señales analógicas de la interfaz electrónica), ajustando el cero en el caso de la velocidad, y recortando su número de decimales con la función “.toFixed()”. Este valor se envía al cuadro de texto volviendo a hacer uso de la función “getElementById()”, pero esta vez expresada a la izquierda del igual por lo que no será una lectura si no una escritura sobre el cuadro de texto indicado por el atributo id correspondiente.

Finalmente, la variable cambio se pone a 0 para establecer un valor seguro que no sea 1. Si el cambio se produjera en los <input> de cuadro de texto se ejecuta otra función que se verá más adelante que pone a 1 la variable “cambio” para que no se realicen estos condicionales aquí explicados, pues ya no son necesarios.

```
valor_S_ON = document.getElementById("control_S_ON").value;
valor_P_CL = document.getElementById("control_P_CL").value;
valor_N_CL = document.getElementById("control_N_CL").value;
```

En estas líneas se procede a la lectura por id de las diferentes entradas digitales y a su almacenamiento en las variables correspondientes.

```
var input = {
        DC_V_REF: valor_V_REF,
        DC_T_REF: valor_T_REF,
        S_ON: valor_S_ON,
        P_CL: valor_P_CL,
        N_CL: valor_N_CL
      }
```

Se declara internamente esta variable llamada "input" a la cual se le aplica el formato JSON, JavaScript lo reconoce solo con la sintaxis empleada. Las palabras a la izquierda de los dos puntos indican el nombre que tienen ahora esos datos en el JSON, lo que se conoce como clave y los que se deberán emplear en el servidor para ser leídas, y a la izquierda se escribe su valor almacenado en las variables empleadas.

```
websocket.send(JSON.stringify(input));
```

Esta línea de código realiza el envío mediante Websocket con la función .send, y además la variable "input" que contiene la información es convertida en un vector de formato JSON gracias a la función "JSON.stringfy()", y es este vector el que finalmente se envía al servidor.

```
enviados=enviados+1;
document.getElementById("enviados").innerHTML=enviados;
};
```

Se aumenta en uno el contador de paquetes enviados y se imprime en la página web el nuevo valor. Al hacer referencia a texto en vez de a valores numéricos se debe emplear la referencia a texto de la función "getElementById()" con (.innerHTML) en vez de con (.value).

1.4.3 Recepción de datos del servidor

Esta función recibe y decodifica el vector JSON recibido e imprime en la página web los nuevos datos con las salidas de control. Su código es el siguiente.

```
(websocket.onmessage)  = function(event) {
```

Esta función se ejecuta cada vez que aparece el evento "onmessage" de Websocket, que indica la recepción de un paquete de información del servidor.

```
var output = JSON.parse(event.data);
```

Se define la variable output para almacenar la información que hay en el JSON recibido, la cual se puede decodificar gracias a la función "JSON.parse()" que decodifica el dato recibido en el evento del Websocket.

```
document.getElementById("Vel_encoder_meter").value = output.Vel_encoder;
document.getElementById("Vel_encoder_value").innerHTML = output.Vel_encoder;
if(output.Vel_encoder>3000){
 document.getElementById('Vel_encoder_value').style.color = '#ff0000';
}
else{
 document.getElementById('Vel_encoder_value').style.color = '#000000';
}
```

Estas líneas imprimen en el sitio web los valores de velocidad del encoder. Una vez decodificada la información se escribe en los elementos de la página web empleando la función "getElementById()" comentada anteriormente, el único cambio es que va a escribirse sobre una

salida con texto en vez de con números, en lugar de (.value) se emplea (.innerHTML). Por otro lado, para hacer referencia a la información que había en el JSON se debe escribir la variable donde se ha decodificado el archivo, seguida de un punto y el nombre que a dicho dato se le ha asignado en el JSON cuando se creó en el servidor, es decir "output.Vel_encoder" es la variable que contiene el valor que se ha almacenado en el JSON con el nombre "Vel_encoder".

Los condicionales posteriores se emplean para que en el caso de que se supere la velocidad nominal, siendo esta de 3000 rpm, el texto que imprime su valor numérico en el sitio web aparezca de color rojo. Esto se consigue con la función "getElementById()" haciendo referencia a su atributo de estilo y color (.style.color), el cual será el indicado a la derecha del igual con la codificación hexadecimal de colores.

```
document.getElementById("sentido").innerHTML = output.sentido
document.getElementById("V_CMP_meter").value = output.V_CMP;
document.getElementById("TGON_meter").value = output.TGON;
document.getElementById("S_RDY_meter").value = output.S_RDY;
document.getElementById("ALM_meter").value = output.ALM;
```

Estas líneas de código imprimen por pantalla en la página web las salidas correspondientes. El único comentario es que, al ser la variable "output.sentido" una cadena de caracteres y no un valor, la función "getElementById()" hace referencia al atributo de texto (.innerHTML).

```
recibidos=recibidos+1;
document.getElementById("recibidos").innerHTML=recibidos;
};
```

Finalmente, se aumenta en uno el contador de elementos recibidos y se imprime por pantalla en su lugar correspondiente.

1.4.4 Función cambioEscrito()

Esta función gestiona los cambios en las entradas analógicas cuando se realizan empleando los cuadros de texto pues son estas etiquetas <input> las que tiene el atributo (onchange="cambioEscrito(this)"). Su objetivo es que cuando se escriba un valor dentro del rango permitido, este sea leído y traducido a cierta posición equivalente de velocidad o par en las barras deslizantes entre 0 y 4095, poner a 1 la variable "cambio" y ejecutar la función "enviarDato()", pues ésta última envía el valor leído de las barras deslizantes, no de los cuadros de texto.

```
function cambioEscrito(element){
valor_V_REF_numero = document.getElementById("control_V_REF_numero").value;
valor_T_REF_numero = document.getElementById("control_T_REF_numero").value;
```

Declaración de la función y lectura en las variables globales indicadas anteriormente los valores escritos en los cuadros de texto correspondientes.

```
if(((valor_V_REF_numero<=3300)&&(valor_V_REF_numero>=-3300))&&((valor_T_REF_numero<=0.70053)&&(valor_T_REF_numero>=0))){
        cambio=1;
      }
      else{
        cambio=0;
      }
```

Este condicional es el que activa a 1 la variable cambio solo en caso de que el valor pedido esté dentro del rango permitido para la velocidad y el par. En caso contrario su valor será 0. Esto se ha

codificado de esta forma para que, en caso de recibirse un valor fuera de rangos, al traducirlo al valor de las barras deslizantes se imprima en ellas el valor máximo o mínimo permitido más cercano, pero además, en la ejecución de la función “enviarDato()” al valer 0 la variable cambio en el cuadro de texto se imprimirá el valor de las barras deslizantes el cual será el máximo o mínimo permitidos.

```
valor_V_REF_numero=(valor_V_REF_numero/1.611328125)+4096/2;
valor_T_REF_numero=valor_T_REF_numero/0.000171069;
document.getElementById("control_V_REF").value = valor_V_REF_numero;
document.getElementById("control_T_REF").value = valor_T_REF_numero;
enviarDato();
    };
```

Por último, se traduce el valor de velocidad o par al rango de 0 a 4095 para poder ser enviado a las barras deslizantes. Esta traducción se realiza empleando la precisión en velocidad o par que permite la codificación en 12 bits de las señales eléctricas a las que hacen referencia y el desplazamiento del 0 en el caso de la velocidad. Se explica con más detalle en el capítulo de la memoria de la interfaz electrónica. Se escribe este valor en las barras deslizantes y se ejecuta la función “enviarDato()” para proceder al envío de estos cambios en las entradas.

2. CÓDIGO HTML

En este capítulo se presenta el código que gobierna la interfaz con los controles del servomotor en la pantalla del navegador del dispositivo del usuario. Se destaca que debe ser copiado en el interior de la variable dedicada a ello en el código de Arduino, sin embargo se presenta por separado debido a la gran diferencia entre ambos códigos. Se ha dispuesto de tal modo que permite el copiado de texto en caso de que quiera ser comprobado o ejecutado en Visual Studio o en navegadores.

```
<!DOCTYPE html>

<head>

  <style>

    .encabezado{
      background:#a4eeb4;
      border:3px solid #000000;
    }

    #titulo{
      font-size: 45px;
    }

    .cuerpo{
      background: #b0e2f0;
      font-size: large;
    }

    .entradaDigital{
      position:relative;
      width: 20%;
    }

    #control_S_ON, #control_P_CL, #control_N_CL{
      width: 25%;
    }

    .entradaAnalogica{
      position:relative;
      width: 30%;
    }

    #control_V_REF, #control_T_REF {
      width: 60%;
    }

    .salidaEncoder{
```

```
    position:relative;
    width: 30%;
  }

  #Vel_encoder_meter{
    width: 70%;
  }

  #Vel_encoder_value{
    color:#000000;
  }

  .separacion{
    position:relative;
    background: #000000;
    border:1.5px solid #000000;
  }

  .salidaDigital{
    position: relative;
    width: 20%;
  }

  .piepagina{
    background:#a4eeb4;
    border:3px solid #000000;
  }

  .comunicacion{
    position: relative;
    width: 33.333%;
  }

  .TFG{
    position: relative;
    width: 33.333%;
  }

  .escuela{
    position: relative;
    width: 33.333%;
  }

 </style>



</head>
```

```
    <center>
      <header class="encabezado">

        <h1 id="titulo">CONTROL SERVOMOTOR</h1>

      </header>

      <body class="cuerpo">

          <table>

          <tr>

          <th  class="entradaDigital">
              <h4>SERVO-ON</h4>
              OFF  <input onchange= "enviarDato(this)" id="control_S_ON" type="range" min="0"
max="1" value="0" step ="1">  ON
              <br>
              <br>
              <br>
              <br>
              <h4>LIMITE PAR HORARIO</h4>
              OFF  <input onchange= "enviarDato(this)" id="control_P_CL" type="range" min="0"
max="1" value="0" step ="1">  ON
              <br>
              <br>
              <br>
              <br>
              <h4>LIMITE PAR ANTIHORARIO</h4>
              OFF  <input onchange= "enviarDato(this)" id="control_N_CL" type="range" min="0"
max="1" value="0" step ="1">  ON
              <br>
              <br>
              <br>
              <br>
          </th>

          <th class="entradaAnalogica">
              <h3>Control de la velocidad</h3>
              <br>
              0
              <br>
              -3300 <input onchange= "enviarDato(this)" id="control_V_REF" type="range" min="0"
max="4095" value="2048"> +3300
              <br>
              <br>
              Valor velocidad: <input onchange="cambioEscrito(this)" id="control_V_REF_numero"
type="text" min="-3300" max="3300" value="0" size="6" > rpm
```

```
            <br>
            <br>
            <br>
            <br>
            <h3>Control limite de par</h3>
            <br>
            0 <input onchange= "enviarDato(this)" id="control_T_REF" type="range" min="0"
max="4095" value="0"> 0.7
            <br>
            <br>
            Valor limite par: <input onchange="cambioEscrito(this)" id="control_T_REF_numero"
type="text" min="0" max="0.700527" value="0" size="6"> Nm
            <br>
            <br>
        </th>

        <th class="separacion"></th>

        <th class="salidaEncoder">
            <h3>Velocidad del motor</h3>
            <meter value="0" min="0" max="3300" id="Vel_encoder_meter"></meter>
            <br>
            <h3 id="Vel_encoder_value" style="display: inline-block;">0</h3> rpm
            <br>
            <br>
            <br>
            <h3>Sentido de giro: </h3>
            <br>
            <h3 id="sentido" style="display: inline-block;"></h3>
            <br>
        </th>

        <th class="salidaDigital">
            <h4>VELOCIDAD CONSIGNA</h4>
            <meter value="0" min="0" max="1" id="V_CMP_meter"> </meter>
            <br>
            <br>
            <br>
            <h4>VELOCIDAD SUPERADA</h4>
            <meter value="0" min="0" max="1" id="TGON_meter"> </meter>
            <br>
            <br>
            <br>
            <h4>SISTEMA PREPARADO</h4>
            <meter value="0" min="0" max="1" id="S_RDY_meter"> </meter>
            <br>
            <br>
            <br>
            <h4>ALARMA</h4>
```

```
            <meter value="0" min="0" max="1" id="ALM_meter"> </meter>
            <br>
            <br>
            <br>
        </th>

        </tr>

        </table>

    </body>

    <footer class="piepagina">
     <table>
       <tr>
         <th class="comunicacion">
           <h4>COMUNICACION CON EL SERVIDOR ESP32</h4>
           <p>Paquetes recibidos: <span id="recibidos"> </span></p>
           <p>Paquetes enviados: <span id="enviados"> </span></p>

         </th>

         <th class="TFG">
           <h4>TRABAJO FINAL DE GRADO</h4>
           <p>Autor: Ismael Ferrer Vaya</p>
           <p>Tutor: Javier Martinez Roman</p>

         </th>

         <th class="escuela">
           <h4>ESCUELA TECNICA SUPERIOR INGENIERIA INDUSTRIAL</h4>
           <p>Departamento de ingenieria electrica</p>
           <p>Catedra Power Electronics</p>

         </th>

       </tr>

      </table>

    </footer>
  </center>
<script>

  var websocket = new WebSocket('ws://'+location.hostname+':81/');

  var enviados =0;
  var recibidos=0;
  var valor_S_ON
```

```
var valor_P_CL;
var valor_N_CL;
var valor_V_REF;
var valor_T_REF;
var valor_V_REF_numero;
var valor_T_REF_numero;
var cambio=0;

function enviarDato(element) {

  valor_V_REF = document.getElementById("control_V_REF").value;
  if(cambio!=1){
    valor_V_REF_numero=(valor_V_REF-4096/2)*1.611328125;
    valor_V_REF_numero=valor_V_REF_numero.toFixed(2);
    document.getElementById("control_V_REF_numero").value = valor_V_REF_numero;
  }

  valor_T_REF = document.getElementById("control_T_REF").value;
  if(cambio!=1){
    valor_T_REF_numero=valor_T_REF*0.000171069;
    valor_T_REF_numero=valor_T_REF_numero.toFixed(5);
    document.getElementById("control_T_REF_numero").value = valor_T_REF_numero;
  }
  cambio=0;

  valor_S_ON = document.getElementById("control_S_ON").value;
  valor_P_CL = document.getElementById("control_P_CL").value;
  valor_N_CL = document.getElementById("control_N_CL").value;

  var input = {
    DC_V_REF: valor_V_REF,
    DC_T_REF: valor_T_REF,
    S_ON: valor_S_ON,
    P_CL: valor_P_CL,
    N_CL: valor_N_CL
  }

  websocket.send(JSON.stringify(input));
  enviados=enviados+1;
  document.getElementById("enviados").innerHTML=enviados;

}

(websocket.onmessage)  = function(event) {

  var output = JSON.parse(event.data);

  document.getElementById("Vel_encoder_meter").value = output.Vel_encoder;
  document.getElementById("Vel_encoder_value").innerHTML = output.Vel_encoder;
```

```
      if(output.Vel_encoder>3000){
        document.getElementById('Vel_encoder_value').style.color = '#ff0000';
      }
      else{
        document.getElementById('Vel_encoder_value').style.color = '#000000';
      }

      document.getElementById("sentido").innerHTML = output.sentido
      document.getElementById("V_CMP_meter").value = output.V_CMP;
      document.getElementById("TGON_meter").value = output.TGON;
      document.getElementById("S_RDY_meter").value = output.S_RDY;
      document.getElementById("ALM_meter").value = output.ALM;

      recibidos=recibidos+1;
      document.getElementById("recibidos").innerHTML=recibidos;

    };

    function cambioEscrito(element){

      valor_V_REF_numero = document.getElementById("control_V_REF_numero").value;
      valor_T_REF_numero = document.getElementById("control_T_REF_numero").value;

      if(((valor_V_REF_numero<=3300)&&(valor_V_REF_numero>=-3300))
&&((valor_T_REF_numero<=0.70053)&&(valor_T_REF_numero>=0))){
        cambio=1;
      }
      else{
        cambio=0;
      }

      valor_V_REF_numero=(valor_V_REF_numero/1.611328125)+4096/2;
      valor_T_REF_numero=valor_T_REF_numero/0.000171069;
      document.getElementById("control_V_REF").value = valor_V_REF_numero;
      document.getElementById("control_T_REF").value = valor_T_REF_numero;
      enviarDato();
    };

  </script>


</html>
```

3. ANÁLISIS DEL CÓDIGO EN ARDUINO

En este apartado se pretende analizar con detalle el código que conforma el servidor y la comunicación con el servopack. Al actuar como vía de comunicación entre usuario y servomotor las funciones y código que cumplen con cada uno de los dos objetivos principales están estrechamente relacionadas entre sí resultando imposible su separación y análisis individual, pues se mantiene en todo momento la interacción con las dos partes.

Arduino cuenta con dos funciones propias tradicionalmente empleadas para el control de los proyectos. La función "setup()", la cual se ejecuta una única vez en todo el programa, empleada para inicializar entradas y salidas, conexiones inalámbricas, canales o todo tipo de funcionalidades que deban ser activadas previamente a la ejecución continua del programa. Y la función "loop()", la cual se trata de un bucle infinito, muy inclinado a procesos de control continuados, en el que las líneas de código escritas se repetirán a cada iteración y es donde se programa la actuación del microcontrolador.

Sin embargo, en este proyecto a esta estructura habitual se le añade una nueva función temporizada que se ejecuta en paralelo a la función "loop()" y que se encarga de enviar los datos al usuario, adicionalmente la función "loop()" pierde importancia pues solo se encarga de mantener la función de atención de eventos de Websocket, pero la codificación de la respuesta se lleva a cabo en una función auxiliar a esta.

Se expone a continuación la explicación del código de Arduino dividido en sus apartados más significativos. El código se diferencia del resto del texto empleando una fuente y unos colores típicos del entorno de desarrollo de Arduino.

El código completo se encuentra en el capítulo siguiente de estos anexos.

3.1 Inclusión de las librerías y definición de variables globales

```
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <WebSocketsServer.h>
#include <Ticker.h>
#include <ArduinoJson.h>
```

Estas líneas incluyen en el código las diferentes librerías empleadas comentadas anteriormente.

```
int DC_V_REF;
int DC_T_REF;
int S_ON;
int P_CL;
int N_CL;
float Vel_encoder=0;
int horario=0;
int pulsosA;
int alarma;
```

Inmediatamente después de la inclusión de librerías se deben declarar las variables globales, es decir, aquellas que pueden ser empleadas por todas las funciones del programa manteniendo su valor, son todas de tipo entero a excepción de Vel_encoder que es empleada para el cálculo de la

velocidad con el encoder. Las dos primeras almacenan la información de las señales analógicas de velocidad y par, las tres siguientes las señales digitales de SERVO-ON, límite de par horario y antihorario respectivamente. La variable horario almacena el sentido de giro, pulsosA se emplea para el recuento de pulsos del encoder y alarma es una señal de control que indica de la activación de la alarma. Pese a que muchas se podrían haber expresado como booleanas, se han declarado como enteras debido a problemas con las librerías externas.

3.2 Selección de red y dirección IP estática

Este apartado tiene gran importancia para el uso por parte del usuario del sistema, pues en esta parte del código deberá escribir la dirección de la red Wifi a emplear, su contraseña y la dirección IP estática que más le convenga que tenga el servidor.

Las direcciones IP estáticas, a diferencia de las dinámicas, aparte de tener ciertas ventajas como mayor velocidad de conexión, tienen la capacidad de hacer que el router siempre les asigne la misma dirección. Esto es una gran ventaja, pues de lo contrario cada vez que se reiniciara el microcontrolador o el servidor se le podría asignar una dirección IP diferente y cada vez habría que conectarse por el puerto serie del microcontrolador para conocer la IP que le ha sido asignada y poder acceder a ella desde otro dispositivo. Por ello se opta por las direcciones estáticas, sin embargo, no todas las direcciones IP están disponibles, pues puede estar siendo usada o no es soportada por la red, por ello se permite al usuario modificarla. La seleccionada por defecto es la dirección 192.168.1.116, probada con éxito en diversos router.

```
//Seleccion de nombre de red y clave de acceso
const char* ssid = "Nombre_de_red_wifi";
const char* contrasena = "La_clave_de_red_wifi";
```

Se declaran dos constantes de tipo caracter para almacenar el nombre de la red y su contraseña. Más adelante se emplearán en la función que inicializa la conexión Wifi.

```
//Definicion de la direccion IP
IPAddress local_IP(192, 168, 1, 116);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(8, 8, 8, 8);
IPAddress secondaryDNS(8, 8, 4, 4);
```

Estas líneas permiten el uso de la dirección IP estática, siendo esta los números indicados en la primera línea. Para modificarla se recomienda el cambio del último número, pero si se requiere el cambio de alguno de los tres primeros, se deben cambiar tanto en la primera como la segunda línea. “gateway” es el punto de acceso que conecta el dispositivo a la red local, pues esta es la dirección IP del router. “subnet” se emplea para la creación de subredes en una dirección IP, en este proyecto no se emplea, pero la función que establece la IP estática la requiere, por lo que se emplea su valor por defecto. Lo mismo ocurre con los “DNS” que sirven para la búsqueda en internet traduciendo el texto de búsqueda en la dirección IP correspondiente.

3.3 Variable del código HTML

Los servidores requieren de tener almacenado el código HTML para enviarlo en el momento de la conexión. De esta tarea se encarga la siguiente variable.

```
char paginaweb[] PROGMEM = R"=====(
<!DOCTYPE html>
<head>
  <style>
… Resto del código en HTML …
</script>
</html>
)=====";
```

Se trata de un vector de caracteres de gran tamaño llamado "paginaweb", además se ha almacenado en la memoria FLASH del microcontrolador empleando la palabra clave al lado de su nombre "PROGMEM". Es común almacenar en esta memoria variables de gran tamaño cuyo contenido no vaya a variar durante la ejecución del programa, pues se trata de una memoria de elevada capacidad pero de baja velocidad de transferencia. Normalmente en la memoria FLASH se almacenan las instrucciones de programa y en RAM las variables. Los primeros caracteres del vector, antes de ser el código HTML, son indicaciones para la función que envía dicho código al usuario y conocer su principio y fin.

3.4 Definición de los puertos y de la función timer

```
AsyncWebServer server(80);
WebSocketsServer websockets(81);
Ticker timer;
```

Se asigna al servidor asíncrono su puerto, el 80, y al servidor de Websocket el 81, siendo estos sus puertos por defecto. También se definen las palabras "server" y "websockets" como indicadores de ambos servidores. La última línea de código se refiere a una técnica que utiliza la interrupción de hardware del temporizador "timer" para generar eventos periódicos con precisión temporal y así conseguir realizar acciones a intervalos regulares.

3.5 Función setup()

Primera de las funciones principales del código, en ella se definen entradas y salidas, se establece la conexión Wifi, se inicializan los servidores y los contadores y también se programan ciertas actuaciones del servidor. Se ejecuta una única vez durante la ejecución normal del programa.

```
void setup(void){
```

Es de tipo "void" (vacío) pues no devuelve ningún valor, así como tampoco recibe ninguno. Solo se programa su definición, pues la llamada a la función la realizará autónomamente el compilador.

Dentro de ella se codifican los siguientes apartados.

3.5.1 Definición de entradas y salidas

Como se comentó con anterioridad, las señales que son entrada al microcontrolador, el usuario las considera salidas, pues las lee del servopack, pero las recibe el ESP32 y viceversa. En este apartado se considera entrada todo aquello que lee el microcontrolador, y salida toda aquella señal que este genera.

```
Serial.begin(115200);
```
Inicia el uso de la comunicación serial mediante el puerto micro-USB. El número entre paréntesis son los baudios o bits por segundo empleados en la comunicación, cuyo valor deberá coincidir con el del receptor para una buena comunicación serie.

```
pinMode(23, OUTPUT);
ledcSetup(0,19500, 12);
ledcAttachPin(23, 0);
```
Define el GPIO 23 como salida mediante la función "pinMode", su primer argumento indica el GPIO y el segundo si se trata de una entrada o una salida. En la siguiente línea se inicializa el canal 0 de la funcionalidad "ledc" capaz de generar PWM, indicándole la frecuencia de 19500 Hz y su codificación del ciclo de trabajo en 12 bits. En la última línea se establece que el canal 0 se corresponde con el GPIO 23 definido como salida, pues en ella se va a generar la señal analógica de control de par. La codificación para la señal analógica de control de la velocidad es análoga, pero empleando el GPIO 22 y el canal de PWM 1.

```
//Salidas digitales ESP32
pinMode(2,OUTPUT);//Salida Servo-on
//Entradas digitales ESP32
pinMode(26,INPUT);//Entrada pulsos encoder
pinMode(35,INPUT);//Entrada velocidad de consigna
```
Se muestran algunos ejemplos de la definición de entradas y salidas digitales, para ello se requiere exclusivamente el uso de la función "pinMode()". La codificación para las demás señales es igual variando el GPIO empleado.
A continuación, se ha añadido la siguiente línea de código:

```
ledcWrite(1, 4096/2);
```
Esta función genera y modula la señal PWM saliente por el canal indicado en su primer argumento y el ciclo de trabajo indicado en el segundo. Se debe recordar que al emplear 12 bits en su codificación su ciclo de trabajo varía de 0 a 4095. Debido a que la señal de velocidad es simétrica el cero se encuentra a la mitad de este intervalo, por lo que esta línea se encarga de enviar la tensión que se corresponderá con una velocidad nula, para evitar arranques bruscos del servomotor al iniciar el sistema y no enviar tensión antes de estar preparado el servopack, pues esta función inicia el ciclo de trabajo en 0 por defecto.

3.5.2 Conexión a la red Wifi

Se emplean un conjunto de funciones pertenecientes a la librería "Wifi.h" dedicadas a iniciar la conexión Wifi en la red deseada y con la dirección IP indicada.

```
if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)){
Serial.println("Fallo en la configuración de IP estática");
}
```

La función incluida dentro del condicional es capaz de configurar la conexión de modo que se le asigne al servidor la dirección IP estática solicitada. Para ello requiere como argumentos las diferentes variables de conexión wifi que han sido definidas con anterioridad en el código. Esta función devolverá un 0 lógico en caso de que se haya producido algún error, por ello la negación del resultado de la función con el operador "!" provoca que ante un error el condicional reciba un 1, haciendo que se ejecute su código entre corchetes.

La función "Serial.println()" envía mediante la comunicación del puerto serie el texto indicado entre comillas seguido de un salto de línea. Se diferencia de "Serial.print()" en que esta última no realiza salto de línea al final. El usuario conectado a dicho puerto del microcontrolador podrá leer dicho texto en el monitor serial de que dispone el entorno de desarrollo de Arduino.

```
WiFi.begin(ssid, contrasena);
Serial.print("Conectando");
while (WiFi.status()!= WL_CONNECTED){
}
Serial.print("Conectado, direccion IP: ");
Serial.println(WiFi.localIP());
```

La función de la primera línea solicita la conexión a la red de nombre almacenado en la variable "ssid", con contraseña almacenada en la variable "contrasena". La conexión se completa una vez la función "WiFi.status()" devuelve la palabra reservada WL_CONNECTED, hasta que esto ocurra el bucle seguirá consultando su estado. Una vez realizada la conexión se imprimirá por el monitor serie el mensaje de conexión y la dirección IP asignada.

3.5.3 Inicialización de servidor y Websocket

Se emplean funciones incluidas en las librerías "ESPAsyncWebServer.h" y "WebSocketsServer.h". El código siguiente se encarga de establecer los servidores web y Websocket, definir el envío de la página ante una petición de conexión y su reacción frente a un error.

```
server.on("/", HTTP_GET, [](AsyncWebServerRequest * request){
request->send(200, "text/html", paginaweb);
});
```

La primera línea establece un manejador para la ruta "/", es decir la dirección principal del servidor pues "/" equivale a la dirección IP con una barra final, y el método HTTP GET. Cuando se realiza una solicitud GET por parte del usuario a esta ruta, se ejecutará la función indicada entre los corchetes, "request->send". Esta se encarga de enviarle al cliente (el usuario) que realizó la petición una respuesta. Se utiliza el código de estado HTTP 200, que indica una respuesta exitosa. El segundo parámetro es el tipo de contenido de la respuesta, que en este caso es "text/html", indicando que se trata de una página web en HTML. El tercer parámetro es el contenido de la página web, que se almacena en la variable "paginaweb". Enviando así ante una conexión el código del sitio web al usuario.

```
server.onNotFound(notFound);
```

Esta función se utiliza para configurar un manejador de ruta para todas las solicitudes que no coinciden con ninguna ruta previamente definida en el servidor web. Se le pasa como argumento la función "notFound" cuyo código se muestra más adelante.

```
void notFound(AsyncWebServerRequest *request){
  request->send(404, "text/plain", "Página no encontrada");}
```
Se trata de la declaración de una función definida fuera de la función “setup()”, pero es dentro de ella donde se le da uso. La función “notFound()” toma un puntero a un objeto AsyncWebServerRequest como parámetro, que representa la solicitud HTTP que no coincide con ninguna ruta definida para el servidor. Dentro de la función, se utiliza “request->send()” para enviar una respuesta al cliente con el código de estado 404 que indica no encontrado, un tipo de contenido de “text/plain” (texto plano) y el mensaje “Página no encontrada”.

```
server.begin();
websockets.begin();
```
Una vez definidas todas las rutas del servidor se llama a esta función, encargada de inicializar el servidor y permitir que comience a manejar solicitudes. La siguiente línea inicia la comunicación mediante Websocket.

```
websockets.onEvent(webSocketEvent);
```
Esta función establece un manejador ante la recepción de los diferentes eventos que se dan mediante la comunicación con Websocket. Se le indica como argumento el nombre de otra función llamada “webSocketEvent” que se ha codificado a parte y que programa el comportamiento frente a los distintos eventos, será analizada con más detalle en apartados siguientes.

3.5.4 Temporización de la función de envío de datos

Se pretende que los datos lleguen al dispositivo del usuario a intervalos de 0,1 segundos. Esto se consigue gracias a la temporización interna y a la posibilidad de ejecución en paralelo.

```
timer.attach(0.1,enviarDatos);
```

Esta función de la librería “Ticker.h” configura el temporizador para que cada 0,1 segundos como se le indica en el primer parámetro, se ejecute la función “enviarDatos()”, cuya codificación se especifica más adelante y en paralelo a la función “loop()”.

3.6 Recepción de datos del usuario

Este conjunto de código responde a la aparición de eventos de la comunicación Websocket, por lo que debe estar siempre a la espera de estos. Una vez el cliente realice algún tipo de interacción con el servidor, como puede ser la conexión, la desconexión o el envío de información, aparecerá un evento que mediante las funciones empleadas de la librería “WebSocketsServer.h” podrán ser detectados y se realizará la acción programada correspondiente como respuesta.

3.6.1 Bucle de comunicación

Para ello se hace uso de otra función tradicional de Arduino, la función “loop()” que se ejecuta repetidamente como un bucle infinito durante toda la ejecución del programa después de finalizar la función “setup()”.

```
void loop(void) {
  websockets.loop();
}
```
Dentro de “loop()” se ejecuta esta función perteneciente a la librería externa, la cual se encarga de establecer el canal de comunicación entre cliente y servidor y predispone a ambos a la recepción de los eventos, deberá estar ejecutándose continuamente durante toda la ejecución del programa.

3.6.2 Codificación de la respuesta a eventos

Como se ha definido en la función “setup()” ante un evento de Websocket se ejecutará la función “webSocketEvent”, en la que se ha programado toda respuesta posible frente a los eventos que puedan ser recibidos. Se presenta la definición de dicha función.

```
void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length) {
```
Esta función se ejecuta cada vez que se reciba un evento. Los parámetros que se le pasan se guardan en variables internas de esta función y su significado es el siguiente:

“uint8_t” Representa el número del cliente WebSocket al que se refiere el evento, y esta información se almacena en la variable “num”.

“WStype_t” es un valor que almacena palabras reservadas que hacen referencia al tipo de evento que se ha producido, algunas de estas pueden ser “WStype_DISCONNECTED” para indicar la desconexión del cliente o “WStype_TEXT” para indicar el envío de datos del cliente, se almacena en la variable “type”.

“uint8_t *” se trata de un puntero que apunta a los datos recibidos en el evento, especialmente en un envío, se almacena en “payload”.

“size_t” representa el tamaño de los datos recibidos, se guarda en la variable “length”.

```
switch (type){
    case WStype_DISCONNECTED:
      Serial.printf("Usuario desconectado\n");
      break;
    case WStype_CONNECTED: {
        IPAddress ip = websockets.remoteIP(num);
        Serial.printf("Usuario conectado en %d.%d.%d.%d url: %s\n", ip[0], ip[1], ip[2], ip[3], payload);
      }
      break;
    case WStype_TEXT: {
```

Dentro de la función aparece la estructura de control de tipo switch que evalúa el valor de la variable “type” y si este coincide con alguno de los escritos después de la palabra “case” ejecutará el código siguiente hasta la llegada a la palabra “break”. Ante la desconexión imprimirá en el monitor serie al mensaje de desconexión. Ante la conexión almacenará la dirección IP del cliente conectado mediante la función “websockets.remoteIP(num)” siendo “num” el identificador del cliente, y posteriormente imprimirá por el monitor serie el mensaje de conexión, con la dirección IP y con los datos recibidos de la conexión. La función “Serial.printf()” tiene un funcionamiento análogo a la correspondiente en lenguaje C, en la que se puede imprimir el valor de las variables indicadas entre comas al final de los parámetros empleando el “%” seguido de una letra

identificadora del tipo de variable dentro del texto. En caso de recibir el evento de recepción de mensaje se ejecuta el siguiente código.

```
if ((num==0)&&(alarma==0)){
      Serial.printf("Vector recibido: %s\n", payload);
      String mensaje = String((char*)( payload));
      Serial.println(mensaje);
      DynamicJsonDocument doc(400);
      DeserializationError error = deserializeJson(doc, mensaje);
      if (error) {
        Serial.print("No se ha podido decodificar el mensaje: ");
        Serial.println(error.c_str());
        return;
        }
```

El condicional exige que solo se acepte la información en caso de que la haya enviado el primer cliente conectado “num=0” y que no se haya activado el protocolo de alarma “alarma=0”. Si se cumplen estas condiciones aparece en el monitor serie el mensaje de recepción de información, posteriormente se almacena en la variable “mensaje” el vector de caracteres de aquello a lo que apunta “payload”, es decir, se transforma los datos recibidos a un vector de caracteres que en la línea siguiente se imprime por pantalla.

No obstante, para que el programa pueda interpretar estos datos se debe decodificar el archivo JSON en el que se almacenan. En primer lugar, se debe crear un documento JSON en memoria con “DynamicJsonDocument doc(400)”, creando uno de tamaño variable al necesario de como máximo 400 bytes. Las líneas siguientes emplean la función “deserializeJson(doc, mensaje)” para almacenar en el JSON creado los datos que se almacenan en el vector de caracteres “mensaje” y así el programa pueda trabajar con ellos empleando el formato JSON con el que fueron enviados facilitando así su comprensión. Esta función en caso de error devuelve un 1 lógico que se almacenará en la variable “error”, en caso de producirse provocará junto con el condicional que se imprima el mensaje del fallo en la decodificación por el monitor serie.

En este formato es sencillo referenciar el dato buscado escribiendo la palabra que se empleó en JavaScript en la parte del usuario para almacenar el valor en cuestión. Por ejemplo, el valor requerido de velocidad podrá conocerse escribiendo “doc["DC_V_REF"]”.

```
DC_V_REF = doc["DC_V_REF"];
ledcWrite(1, DC_V_REF);
```

Se almacena el ciclo de trabajo (valor entre 0 y 4095) en la variable de Arduino “DC_V_REF”, y esta posteriormente se empleará como parámetro para indicar el valor de la señal analógica de la velocidad. La codificación es análoga para el caso del par.

```
N_CL=doc["N_CL"];
digitalWrite(15,N_CL);
```

Se almacena en la variable de Arduino “N_CL” el valor lógico digital correspondiente a la señal de límite de par antihorario. Para generar la señal digital se emplea la función “digitalWrite()” indicando su GPIO correspondiente y el estado lógico almacenado en la variable. El empleo es igual para las demás señales digitales.

3.7 Determinación velocidad y sentido con el encoder

Estas líneas de código las emplea la función "enviarDatos()" para conocer la información proporcionada por los pulsos del encoder, sin embargo, para explicarla detalladamente se ha considerado necesario su explicación en un apartado propio. La función encargada de la lectura y cálculo de los pulsos del encoder se llama "control_interrupciones()", pues las operaciones se realizan empleando las interrupciones del procesador por su inmediatez y la capacidad de definir rutinas de respuesta ante su activación.

3.7.1 Funcionamiento del encoder

Como se explicó en el apartado 4.1.2 de la memoria, el encoder es un elemento muy empleado en control para determinar la velocidad y posición de los servomotores. En este caso se trata de uno incremental el cual se puede emplear para el cálculo de velocidad y de sentido de giro. La tecnología puede variar en función del encoder, pero la mayoría funciona enviando trenes de pulsos eléctricos cuya frecuencia de envío puede determinar la velocidad, pues lo que indica cada pulso con respecto al anterior es que se ha completado una porción del giro total. El servopack permite modificar mediante parámetros internos la cantidad de pulsos que equivalen a una vuelta completa, de este modo se obtiene una relación directa entre ambas magnitudes. [27]

Si se le añade la variable temporal contando cuantos pulsos se han dado en un intervalo de tiempo determinado se puede obtener la velocidad de giro en revoluciones por minuto con la siguiente fórmula:

$$Vel_{encoder} = \frac{60 * 10^6 * N\text{º}PulsosContados}{\left(Parámetro\,\frac{Pulsos}{revolución}\,servopack\right) * (TiempoLectura\ en\ microsegundos)} \tag{18}$$

Después de la realización de diversas pruebas en el laboratorio para optimizar las mediciones se ha decidido emplear un parámetro de 1000 pulsos/revolución medidos durante 20000 microsegundos. Dejando como variable determinante de la velocidad el número de pulsos recibidos.

Para conocer el sentido de giro se generan dos trenes de pulsos iguales, cada uno por un canal diferente, pero desfasados entre sí una cuarta parte de período. En función del sentido será una señal la que alcance el nivel alto antes que la otra en cada pareja de pulsos. Por ello el código se encarga de medir el estado de una señal cuando la otra alcanza el estado activo.

Cada una de las dos señales de pulsos está conectada a un pin de interrupción del microcontrolador, por lo que la rutina de interrupción deberá encargarse del tratamiento de estos pulsos.

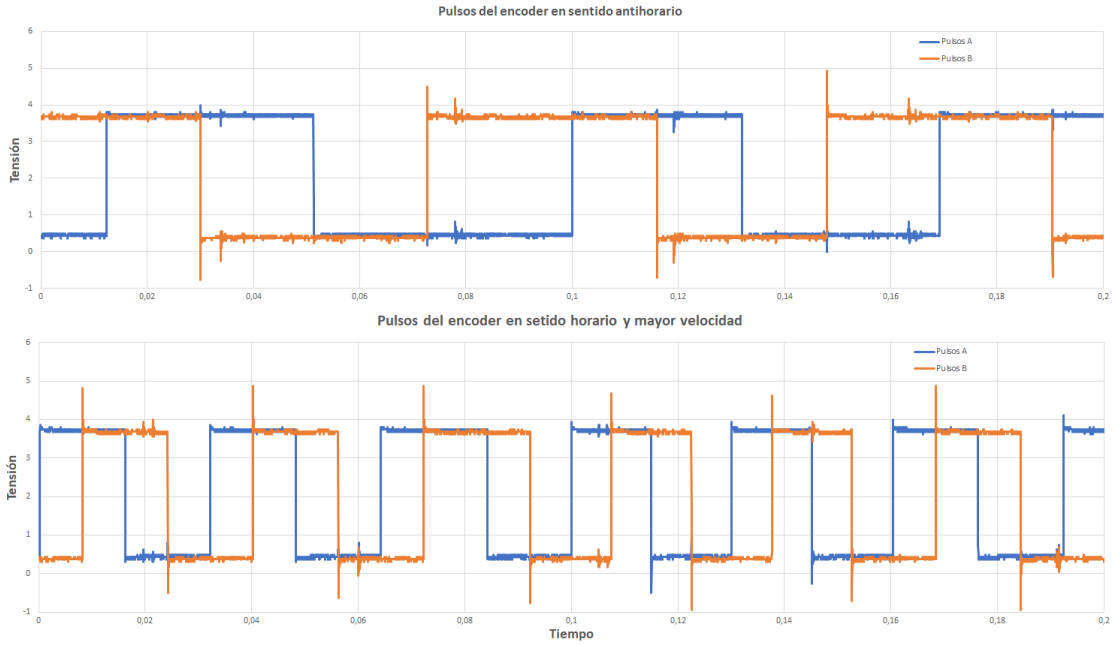

Ilustración 2 de los anexos. Pulsos del encoder. Fuente propia. Mediciones realizadas en el laboratorio de las combinaciones de pulsos enviadas por el encoder del servomotor. Con la tensión medida en voltios y el tiempo en segundos.

3.7.2 Rutinas de interrupción

Estas se declaran antes de la función de "setup()", y su programación es como si se tratara de una función habitual. En ambas rutinas se emplea la palabra dedicada "IRAM_ATTR" antes del nombre de la rutina para hacer que sus instrucciones se almacenen en la memoria RAM para aumentar al máximo su velocidad de actuación.

```
void IRAM_ATTR calculo_velocidad() {
  pulsosA++;
}
```

Dedicada al recuento de pulsos para el cálculo de la velocidad, a cada interrupción debida a un pulso se aumenta la variable "pulsosA" en una unidad.

```
void IRAM_ATTR calculo_sentido() {
  if(digitalRead(26)==HIGH){
    horario=0;}
  else{
    horario=1;}
}
```

A cada interrupción mide el estado de la señal desfasada con la función "digitalRead" indicándole el GPIO correspondiente, en caso de que el pulso esté a nivel alto el sentido será antihorario, de lo contrario será horario.

3.7.3 Función "control_interrupciones()"

En ella se activan y desactivan las interrupciones y se calcula la velocidad resultante.

```
void control_interrupciones(){
  pulsosA=0;
  attachInterrupt(26, calculo_velocidad, RISING);
  delayMicroseconds(20000);
  detachInterrupt(26);
  Vel_encoder=(60000000.0/20000.0)*(1/1000.0)*pulsosA;
…Sentido…
}
```

Para el cálculo de la velocidad primero se debe poner el contador de pulsos a 0, seguidamente se activan las interrupciones con la función "attachInterrupt()" indicándole el GPIO que recibirá la interrupción, el nombre de la rutina a ejecutar y su activación que será al flanco de subida "RISING". La función "delayMicroseconds()" provoca una pausa de los microsegundos indicados para realizar el recuento, una vez pasado este tiempo se desactiva la interrupción con "detachInterrupt()" y se procede al cálculo de la velocidad. El código de control de las interrupciones para el sentido se realiza de forma similar con solamente las tres líneas centrales, cambiando el GPIO al 25, el tiempo de espera a 400 microsegundos e indicando el nombre de rutina de "calculo_sentido".

3.8 Envío de datos al usuario

Tal como se definió en la función "setup()", la función que realiza el envío se ejecuta cada 0,1 segundos. En ella se programa la generación del archivo JSON a enviar, la recepción de datos tanto del encoder como de los pines de hardware, el protocolo en caso de alarma y finalmente el envío de los datos.

3.8.1 Creación del archivo JSON

```
void enviarDatos(){
StaticJsonDocument<200> doc;
JsonObject object = doc.to<JsonObject>();
```

Se crea el documento JSON de forma dinámica con capacidad máxima de 200 bytes y se le asigna el nombre "doc", finalmente la última línea emplea un método que permite modificar el contenido del documento JSON creado.

3.8.2 Recopilación de información

Se mide y escribe en el documento JSON los valores de las diferentes señales de control que se envían al cliente. Para ello se debe emplear la sintaxis "object["Vel_encoder"]=Vel_encoder", donde dentro del corchete se especifica el nombre que tendrá dicho elemento en JSON, y al otro lado de la igualdad su valor, en este caso almacenado en la variable "Vel_encoder".

```
control_interrupciones();
object["Vel_encoder"]=Vel_encoder;

if(horario==1){
 object["sentido"]="Horario";}
…
```

En primer lugar, se realiza la llamada a la función “control_interrupciones()” para obtener los datos de velocidad y sentido, posteriormente son almacenados en el formato JSON. Se expone un fragmento del código que indica el sentido comparándolo con la variable “horario”.

```
…
object["S_RDY"] = digitalRead(39);
object["ALM"] = !digitalRead(36);
```

Se presenta un fragmento de la lectura de señales digitales, mediante la función “digitalRead()” indicando el GPIO correspondiente, las cuales devolverán un 1 lógico en caso de nivel alto de tensión o un 0 en caso contrario. En el caso de la alarma, GPIO 36, al ser activa a valor bajo se debe invertir la lógica binaria para una buena comunicación con el usuario.

3.8.3 Protocolo de alarma

```
if (!digitalRead(36)){
    alarma=1;
    ledcWrite(1, 4096/2);
    ledcWrite(0, 0);
    digitalWrite(2,0);
    Serial.println("ALARMA");}
  else{
    alarma=0;}
```

En caso de activación de la alarma, la variable “alarma” toma el valor 1 bloqueando la recepción de datos del usuario, y se fuerza la desactivación de las señales analógicas, así como la señal SERVO-ON para detener la comunicación.

3.8.4 Envío del archivo JSON

Finalmente se realiza el envío de los datos almacenados al usuario.

```
String jsonString = "";
serializeJson(doc, jsonString);
Serial.println(jsonString);
websockets.broadcastTXT(jsonString);
```

Se define el vector de caracteres “jsonString”, en el cual se almacenará codificado en formato JSON la información almacenada en “doc” mediante la función “serializeJson()”,este vector se escribe en el monitor serie y posteriormente se procede a su envío a todos los clientes conectados como un mensaje de texto a través del canal Websocket.

4. CÓDIGO ARDUINO

Se presenta el código encargado de gestionar el servidor y las comunicaciones eléctricas con el servopack. Para realizar el envío del sitio web debe copiarse el código HTML descrito anteriormente dentro de la variable en la que se indica. Se ha dispuesto de tal modo que permite el copiado de texto para su comprobación o ejecución en la plataforma de Arduino IDE, siempre siguiendo las instrucciones de uso detalladas en la memoria.

```
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <WebSocketsServer.h>
#include <Ticker.h>
#include <ArduinoJson.h>


//Definición de variables globales
int DC_V_REF;
int DC_T_REF;
int S_ON;
int P_CL;
int N_CL;
float Vel_encoder=0;
int horario=0;
int pulsosA;
int alarma;

//Seleccion de nombre de red y clave de acceso
const char* ssid = "Nombre_de_red_Wifi";
const char* contrasena = "La_clave_de_red_wifi";

//Definicion de la direccion IP
IPAddress local_IP(192, 168, 1, 116);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(8, 8, 8, 8);
IPAddress secondaryDNS(8, 8, 4, 4);


char paginaweb[] PROGMEM = R"=====(


    //Se debe copiar dentro de estos paréntesis todo el código HTML.


)=====";
```

```
AsyncWebServer server(80);
WebSocketsServer websockets(81);

Ticker timer;

void notFound(AsyncWebServerRequest *request)
{
  request->send(404, "text/plain", "Página no encontrada");
}

void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length)
{

  switch (type){
    case WStype_DISCONNECTED:
      Serial.printf("Usuario desconectado\n");
      break;

    case WStype_CONNECTED: {
        IPAddress ip = websockets.remoteIP(num);
        Serial.printf("Usuario conectado en %d.%d.%d.%d url: %s\n", ip[0], ip[1],
ip[2], ip[3], payload);
      }
      break;

    case WStype_TEXT:

      if ((num==0)&&(alarma==0)){
      Serial.printf("Vector recibido: %s\n", payload);
      String mensaje = String((char*)( payload));
      Serial.println(mensaje);

      DynamicJsonDocument doc(400);
      DeserializationError error = deserializeJson(doc, mensaje);
      if (error) {
        Serial.print("No se ha podido decodificar el mensaje: ");
        Serial.println(error.c_str());
        return;
        }

      DC_V_REF = doc["DC_V_REF"];
      ledcWrite(1, DC_V_REF);

      DC_T_REF = doc["DC_T_REF"];
      ledcWrite(0, DC_T_REF);

      S_ON=doc["S_ON"];
      digitalWrite(2,S_ON);
```

```
      P_CL=doc["P_CL"];
      digitalWrite(4,P_CL);

      N_CL=doc["N_CL"];
      digitalWrite(15,N_CL);
    }
  }

}

void IRAM_ATTR calculo_velocidad() {
  pulsosA++;
}

void IRAM_ATTR calculo_sentido() {
  if(digitalRead(26)==HIGH){
    horario=0;
  }
  else{
    horario=1;
  }
}

void setup(void)
{
  Serial.begin(115200);

  pinMode(23, OUTPUT);
  ledcSetup(0,19500,12);
  ledcAttachPin( 23, 0);

  pinMode(22, OUTPUT);
  ledcSetup(1,19500,12);
  ledcAttachPin( 22, 1);

  //Salidas digitales ESP32
  pinMode(2,OUTPUT);//Salida S-ON
  pinMode(4,OUTPUT);//Salida P-CL
  pinMode(15,OUTPUT);//Salida N-CL

  //Entradas digitales ESP32
  pinMode(26,INPUT);//Entrada PAO (encoder)
  pinMode(25,INPUT);//Entrada PBO (encoder)

  pinMode(35,INPUT);//Entrada V-CMP
  pinMode(34,INPUT);//Entrada TGON
  pinMode(39,INPUT);//Entrada S-RDY
  pinMode(36,INPUT);//Entrada ALM
```

```
  ledcWrite(1, 4096/2);


  if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS))
{
    Serial.println("Fallo en la configuración de IP estática");
  }

  WiFi.begin(ssid, contrasena);
  Serial.print("Conectando");
  while (WiFi.status() != WL_CONNECTED)
  {
  }
  Serial.print("Conectado, direccion IP: ");
  Serial.println(WiFi.localIP());

  delay(3000);

  server.on("/", HTTP_GET, [](AsyncWebServerRequest * request){
   request->send(200, "text/html", paginaweb);
  });

  server.onNotFound(notFound);
  server.begin();
  websockets.begin();
  websockets.onEvent(webSocketEvent);

  timer.attach(0.1,enviarDatos);

}


void enviarDatos(){

  String jsonString = "";
  StaticJsonDocument<200> doc;
  JsonObject object = doc.to<JsonObject>();

  control_interrupciones();

  object["Vel_encoder"]=Vel_encoder;

  if(horario==1){
    object["sentido"]="Horario";
  }
  else{
    object["sentido"]="Antihorario";
  }
```

```
  if(Vel_encoder==0){
    object["sentido"]="Parado";
  }

  object["V_CMP"] = digitalRead(35);
  object["TGON"] = digitalRead(34);
  object["S_RDY"] = digitalRead(39);
  object["ALM"] = !digitalRead(36); //Para invertir el valor, pues la
alarma es activa a nivel bajo.
  if (!digitalRead(36)){
    alarma=1;
    ledcWrite(1, 4096/2);
    ledcWrite(0, 0);
    digitalWrite(2,0);
    Serial.println("ALARMA");
  }
  else{
    alarma=0;
  }
  serializeJson(doc, jsonString);
  Serial.println(jsonString);
  websockets.broadcastTXT(jsonString);

}

void control_interrupciones(){

  attachInterrupt(25, calculo_sentido, RISING);
  delayMicroseconds(400);
  detachInterrupt(25);

  pulsosA=0;
  attachInterrupt(26, calculo_velocidad, RISING);
  delayMicroseconds(20000);
  detachInterrupt(26);
  Vel_encoder=(60000000.0/20000.0)*(1/1000.0)*pulsosA;


}

void loop(void) {

  websockets.loop();

}
```