UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Diseño y desarrollo de un sistema de realimentación de posición y velocidad para banco de ensayos de servos síncronos de imanes permanentes de 2,2kW.

Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

AUTOR/A: Vicent Crespo, Arnau

Tutor/a: Martínez Román, Javier Andrés

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UN SISTEMA DE REALIMENTACIÓN DE POSICIÓN Y VELOCIDAD PARA BANCO DE ENSAYOS DE MOTORES SÍNCRONOS CON IMANES PERMANENTES

AUTOR: ARNAU VICENT CRESPO

TUTOR: JAVIER ANDRÉS MARTÍNEZ ROMÁN

Curso académico 2022-2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

ÍNDICE DE DOCUMENTOS

MEMORIA

PRESUPUESTO

ANEXOS

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

DESARROLLO DE UN SISTEMA DE REALIMENTACIÓN DE POSICIÓN Y VELOCIDAD PARA BANCO DE ENSAYOS DE MOTORES SÍNCRONOS CON IMANES PERMANENTES

MEMORIA

Trabajo Final de Grado

ARNAU VICENT CRESPO

Junio, 2023

RESUMEN

Los motores eléctricos asíncronos son una parte fundamental e indispensable de la industria moderna. Debido a ello, es muy interesante la propuesta de la implementación de un sistema de retroalimentación sobre la velocidad y el posicionamiento preciso del motor. Esto proporcionará un aumento del uso de motores eléctricos en una amplia variedad de procesos industriales.

Por lo consiguiente, el presente Trabajo Final de Grado se enfoca en la creación de un sistema de control de lazo cerrado de velocidad y posición. Además, los datos muestreados se visualizarán en tiempo real en una página web.

Para lograr este objetivo se empleará un encoder óptico incremental, que mediante tres sensores ópticos se obtendrá la velocidad y la posición del motor asíncrono. Este encoder será diseñado, implementado, fabricado y llevado a la práctica en el transcurso de este proyecto.

Con la ayuda del microprocesador ESP32, se recibirán las señales aportadas por el encoder y se redirigirán a una página web para mostrar la velocidad de forma asíncrona. El programa diseñado dentro del microprocesador se encargará de la conversión de las señales binarias captadas por los sensores, en velocidad. Asimismo, para su posterior representación, se creará una página web que muestre los resultados en una gráfica de manera asíncrona.

Palabras clave: Codificador óptico, interfaz de usuario WEB, microcontrolador

ABSTRACT

Asynchronous electric motors are a fundamental and indispensable part of modern industry. Therefore, it is an interesting option the implementation of a feedback system for precise motor speed and position. This will provide increased in the usage of electric motors in a wide variety of industrial processes.

Therefore, this Final Degree Project focuses on creating a closed-loop control system for motor speed and position. In addition, the sampled data will be displayed in real-time on a web page.

To achieve this goal, an incremental optical encoder will be employed, which will obtain the motor speed and position using three optical sensors. This encoder will be designed, implemented, manufactured, and put into test during this project.

Emptying the ESP32 microprocessor, signals provided by the encoder will be received and redirected to a web page for asynchronous display. The program implemented within the microprocessor will convert the binary signals captivated by the sensors into speed values. Furthermore, a web page will be created to display the results in a graph asynchronously.

Keywords: Optical encoder, WEB user interface, microcontroller.

RESUM

Els motors elèctrics asíncrons són una part fonamental i indispensable per a l'indústria moderna. En conseqüència, es molt interessant la proposta de la implementació d'un sistema de retroalimentació sobre la velocitat i posició precisa del motor. Aquest sistema aportarà un augment de la utilitazció de motors elèctrics en una àmplia varietat de processos industrials.

Per tant, el present Treball de Final de Grau se centra en la creació d'un sistema de control de llaç tancat de velocitat i posició. A més, les dades analitzades apareixeran en temps real en una pàgina web.

Per a aconseguir açò, s'emptarà un encoder óptic incremental, que mitjançant tres sensors òptics obtindrà la velocitat i el posicionament del motor asíncron. Aquest encoder serà dissenyat, implementat, fabricat i portat a la pràctica en el transcurs d'aquest projecte.

Amb l'ajuda del microprocessador ESP32, es rebran els senyals aportades per l'encoder i es redirigirà a una pàgina web on és mostrarà aquests senyals de forma asíncrona. El programa implementat dins del microprocessador s'encarrega de la conversió dels senyals binàries aportades pels sensors, en una velocitat. A més a més, es crearà una pàgina web que mostre en una gràfica els resultats de manera asíncrona.

Paraules clau: Codificador òptic, interfici d'usuari WEB, microcontrolador.

Índice de contenidos

Índice de figuras

Índice de tablas

Índice de códigos

CAPÍTULO 1: INTRODUCCIÓN

La tecnología tiene un papel primordial en la sociedad. Esto se debe a la relevancia que nosotros le asignamos. Cabe destacar que, como ingenieros, la tecnología no es un fin en sí misma, sino una herramienta para mejorar la calidad de vida de las personas.

Muchas veces se observa esta tecnología con cierta incertidumbre, no conociendo el proceso sino sólo el resultado. El proyecto que se va a realizar tiene como finalidad diseñar un sistema que permita medir la velocidad del motor de un banco de ensayos y mostrarla al usuario del banco. Sin embargo, está realizado para comprender fácilmente como se realiza este proceso, ya que en la realidad es un proceso difícil de visualizar.

La medición de la velocidad cuenta con diversas aplicaciones que tienen gran importancia en diversos sectores, como la industria automotriz, la robótica y la automatización de procesos industriales. Además, también está enfocada a otro sector, el docente, en el que los estudiantes de accionamientos deben comprender las diferentes tecnologías de medida de posición y de velocidad con presencia industrial significativa y deben ser capaces de medir esa posición y/o velocidad durante la realización de ensayos de accionamientos.

Con el propósito de lograr tal objetivo, se pretende construir un medidor de velocidades para un motor síncrono de imanes permanentes. Las herramientas que se emplearán son el chip ESP32 y un encoder óptico incremental que se diseñará desde cero. Estas herramientas se encargarán de medir la velocidad y la posición del movimiento de los ejes del motor y transmitir la información a una web para poder visualizarla en forma de gráfica.

El ESP32, será de suma utilidad debido a su capacidad de comunicación, en este caso se hará un uso preciso de su capacidad de transmisión de información vía internet. Para emplear tal herramienta se utilizará la plataforma Visual Studio Code, que, de forma similar a la plataforma de Arduino IDE, permitirá programar el chip para obtener los datos necesarios y enviarlos a una página web.

El proyecto además de incluir el cómo emplear las herramientas mencionadas, también incluye el cómo crear alguna de ellas, más específicamente el encoder óptico incremental. Este proceso implica una gran búsqueda y contraste de información para así poder elegir el encoder óptimo.

Para su creación, se emplearán diversos programas para realizar su diseño y construcción. La precisión y eficacia serán unos de los parámetros con mayor relevancia dentro de este proceso.

En la última parte del proceso se combinarán las dos herramientas mencionadas anteriormente: el chip programado y el encoder óptico incremental construido.

En resumen, el presente proyecto de medidor de velocidad para motores síncronos de imanes permanentes es una muestra del avance tecnológico en el campo de la electrónica, la programación y el diseño. Se emplearán múltiples herramientas para obtener un resultado factible. Como ingenieros, debemos ser responsables en el desarrollo y uso de la tecnología, asegurándonos de que su impacto sea positivo para la sociedad.

1.1 OBJETIVOS

Este trabajo tendrá como objetivo principal la obtención de la velocidad del eje de un motor síncrono con imanes permanentes. Para ello se deberán de lograr dos objetivos.

Primeramente, se deberá de construir una herramienta que sea capaz de medir la velocidad. Esta herramienta como bien se ha comentado en la introducción será un encoder óptico incremental. Este instrumento permitirá medir la distancia que hay entre dos posiciones y el tiempo que transcurren de una a otra, es decir, permitirá medir la velocidad. Para construir el encoder, se dividirá el proceso en tres partes: El diseño, la construcción y el montaje.

La segunda acción por realizar será la traducción de los datos recibidos del encoder a unos datos que puedan ser leídos por el ser humano, es decir, habrá que transformar las señales binarias aportadas por los sensores ópticos en la velocidad correspondiente del motor. Para ello, se empleará el chip ESP32 y la plataforma de programación Visual Studio Code. Este chip recibirá los datos del encoder y mediante un programa implementado en Visual Studio Code, los transformará en la velocidad a la que gira el motor y que posteriormente se visualizará en una web a través de una gráfica.

El trabajo presente de estudio tiene un amplio foco de aplicación, ya que las acciones a realizar tras obtener la velocidad son diversas. Además, este proyecto tiene otra función más trascendente y es el mostrar como realmente funciona un medidor de velocidades de un motor. El encoder óptico que se realizará, será una pieza a bastidor abierto, es decir, se podrá ver en todo momento su funcionamiento, permitiendo una fácil comprensión del proceso. Por consiguiente, el objeto de este proyecto tiene también un carácter y una finalidad docente.

En resumen, los objetivos que se deberán de cumplir son:

-Selección de la tipología del encoder.

-Diseño del encoder.

-Construcción del encoder.

-Montaje del encoder.

-Creación del código para transformar las señales aportadas por el encoder en velocidad.

-Implementación del programa para crear una página web para la realimentación al usuario.

1.2 ANTECEDENTES

Un servomotor es un motor eléctrico que se utiliza para controlar la posición, velocidad y aceleración de un mecanismo o sistema. Se compone de un motor eléctrico, un sistema de control y un sistema de retroalimentación. Los cuales permiten ajustar con exactitud la posición y la velocidad del servomotor; este proceso se repite continuamente.

El sistema de retroalimentación consta de un sensor que proporciona información sobre la posición actual del eje del motor. Esta información se utiliza para ajustar la posición del motor y lograr una precisión y estabilidad en el movimiento.

El presente estudio se realizará en un banco de ensayos con dos servomotores síncronos opuestos uno al otro. Estos servomotores presentan un inconveniente: su sistema de realimentación es un sistema hermético, al cual no se puede acceder ni visualizar. Por ende, se desconoce el funcionamiento del sensor empleado para la obtención de señales y con ello presenta un desafío a la hora de diagnosticar y solucionar cualquier problema relacionado con el sistema de control.

1.3 MOTIVACIÓN

El proyecto que se plantea realizar es crear un nuevo sistema de retroalimentación para estos dos motores síncronos. Este sistema medirá la posición angular y la velocidad del eje de forma asíncrona.

Esto implicará la creación de un encoder que permita medir estos valores, y posteriormente transmitirlos a una web en tiempo real.

Además, el proyecto tendrá otro objetivo, el entendimiento del proceso por terceros, es decir, las personas podrán conocer los fundamentos del sistema de retroalimentación una vez montado. Para ello, el sistema que se deberá de montar tendrá que ser a bastidor abierto, y así poder visualizar el proceso.

El crear un nuevo sistema de retroalimentación implicará un alto grado de conocimientos y aplicación de conceptos de diversos campos, como pueden ser la física dinámica, la programación, diseño y máquinas eléctrica. Esto deriva en un proyecto multidisciplinar.

1.5 OBJETIVOS DE DESARROLLO SOSTENIBLE

El mundo está en una situación que necesita inmediatamente un cambio. Por ello, todos deben de ser conscientes, y ayudar en todo lo posible para conseguir mejorar esta situación.

En la siguiente tabla se mostrarán los diferentes Objetivos de Desarrollo Sostenible para lograr ese fin.

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No procede
ODS 1. Fin de 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 la desigualdad				x
ODS 11. Ciudades y comunidades sostenibles				x
ODS 12. Producción y consumo responsable				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. Alianza para lograr objetivos			x	

Tabla 1 ODS

El proyecto a realizar se concentrará en una de las ramas más importantes, promover y fomentar la educación, intentado conseguir así el objetivo de aumentar considerablemente el número de jóvenes y adultos que tienen las competencias necesarias, en particular técnicas y profesionales, para acceder al empleo, el trabajo decente y el emprendimiento [1]. La educación es el arma más versátil para combatir la ignorancia y la pobreza.

Como bien se ha comentado anteriormente, este proyecto tiene diversos objetivos. Uno de ellos es directamente el enseñar a futuras generaciones cuál es la función de un sistema de retroalimentación que permita medir las velocidades y posición angular del eje de un motor. Este conocimiento, aunque a priori parezca insignificante, está presente en la mayor parte de actividades industriales actualmente, sobre todo con las aplicaciones de motores eléctricos. Esto enlazaría con los ODS 7, 8 y 13; aunque no sean los principales objetivos, también tienen cierta relevancia en el proyecto.

1.4 ESTRUCTURA DE LA MEMORIA

El documento técnico conocido como memoria, presentará la metodología empleada para el desarrollo del proyecto y los resultados obtenidos a través de él. Esta memoria se dividirá en capítulos, los cuales explicarán cada parte del proceso. El orden será el siguiente:

El primer capítulo contendrá las nociones básicas de este proyecto. En él se introducirá el tema, se verá el problema a tratar, se explicarán los motivos de su elección, la repercusión social que tendrá el proyecto y por último la estructura que seguirá este documento técnico.

En el segundo capítulo se explicarán las bases del proyecto realizado. Al necesitar crear un sistema de retroalimentación que pueda medir la velocidad y la posición angular de un eje, será de vital importancia emplear un encoder. Por ello, se estudiarán las características más notorias de cada tipo de encoder, con el fin de tener suficiente información para poder decir el modelo óptimo.

El tercer capítulo, consistirá en el diseño y fabricación del tipo de encoder elegido. Los pasos que se seguirán son los siguientes: Se crearán los planos del encoder para posteriormente imprimir las piezas en una impresora en 3D. Se calculará la resistencia a tracción en los lugares más vulnerables del disco que formará el encoder, para elegir el material que cumpla dichas características específicas. Por último, se mostrará el montaje eléctrico realizado para el enlace entre el ESP32 y el encoder.

En el cuarto capítulo, se explicarán las características del microprocesador ESP32, el motivo por el cual se ha elegido y las herramientas necesarias para trabajar con él.

El quinto capítulo, incluirá el programa empleado para operar el ESP32. Este programa contará de dos partes. La primer será el desarrollo de la página web, que mostrará la velocidad del motor en una gráfica. La segunda parte, se creará el programa que transforme las señales cuadradas obtenidas por los sensores del encoder, en la velocidad a transmitir a la web de forma asíncrona.

En el sexto capítulo, se mostrarán los pasos a seguir para hacer funcionar el banco de ensayos, además de una breve explicación del programa que se emplea.

Por último, en el capítulo siete, se expondrán los resultados obtenidos al desarrollar este proyecto.

CAPÍTULO 2: ENCODERS, CARACTERÍSTICAS Y TIPOS

En este primer capítulo, se mostrarán los tipos más comunes de encoders, sus características tecnológicas y finalmente la ponderación acerca de cuál se tomará para realizar su diseño.

2.1 ENCODER MAGNÉTICO

La función de los encoders magnéticos es la detección de alteraciones de campo magnético a partir de sensores. Estas alteraciones son producidas cuando se mueven los polos magnéticos respecto al sensor. Finalmente, esas alteraciones son transformadas en señales digitales que muestran información relevante al movimiento del encoder, como puede ser velocidad angular, posición y dirección.

Su forma de medir estos parámetros es bastante intuitiva. Al igual que el encoder óptico emplean luz para saber la posición específica del encoder, el encoder magnético emplea el mismo principio para determinar su posición. Pero en vez de emplear luz, utiliza un campo magnético. Hay 3 tipos de encoders magnéticos dependiendo de su forma de determinar la posición del mismo: [2]

-Encoder de engranaje ferromagnético:

Está formado por un engranaje ferromagnético y sensores magnéticos. En el momento que el diente pase por delante del sensor, el cambio magnético que producirá en el campo magnético generará un pulso de tensión que puede ser transformado en velocidad.

Sin embargo, este sensor, tiene una muy baja resolución debido a su limitación por la cantidad de dientes que tiene la rueda. La resolución está normalmente limitada a unas 120 o 240 pulsaciones por revolución (bastante pocas).

-Encoder magnético-resistivo:

Consiste en incorporar una serie de filamentos de resistencias distribuidas a lo largo de una rueda con campos magnéticos alternativos. Conforme la rueda gira, los campos magnéticos pasarán por el conjunto de sensores y la resistencia de salida irá cambiando, lo que producirá una señal sinusoidal.

Ideal para funciones incrementales ya que, con 32 polos, genera 32 pulsos, al contrario que los de efecto Hall, que con 32 polos generan 16 pulsos.

Obtienen una mayor resolución que el encoder mencionado anteriormente. Sin embargo, son más difíciles de acoplar e integrar, encareciendo el precio y aumentando su complejidad.

-Encoder de Efecto Hall:

El efecto Hall, es la tensión que aparece en un material semiconductor cuando se hace circular por él una corriente perpendicular a un campo magnético y al campo eléctrico de la batería [Fig. 1. Encoder de efecto Hall]. Este campo Hall, se forma cuando las cargas eléctricas, en movimiento (corriente) son empujadas por la fuerza de Lorentz en dirección perpendicular a su velocidad y al campo magnético.

Este empuje, es producido por la fuerza creada por el campo magnético sobre la carga en movimiento. En caso de que el campo magnético cese, entonces la tensión será nula. [3]

Para determinar la posición con el efecto Hall, se pone el imán permanente en el rotor y varios sensores Hall en el estator. Estos sensores determinan la polaridad del campo creado por el imán. Dependiendo de la polaridad que detecta el conjunto se puede determinar en qué dirección está orientado el imán (cuando el rotor pasa por el elemento vivo del sensor, la corriente cambia el campo magnético perpendicular al efecto Hall).

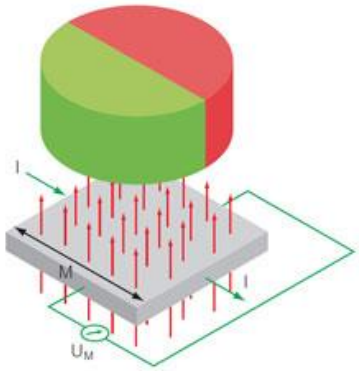


Fig. 1. Encoder de efecto Hall [3]

2.1.1 Encoder incremental

El encoder magnético incremental genera una señal de salida cada vez que el eje gira un determinado ángulo. Estas señales, también llamadas PPR (pulsos por revolución), son producidas por una serie de imanes que se encuentran en la periferia del disco y un sensor de efecto Hall (se producen tantos pulsos como pares de polos hay). Por otra parte, este tipo de encoder, tiene algunas limitaciones, como, por ejemplo: que necesitamos tener un punto de referencia para poder conocer la posición del encoder y que se requiere otro sensor de efecto Hall para poder obtener el sentido en el que gira el disco. Este último sensor, deberá de estar desfasado 90 º en el espacio respecto al otro, ya que así se producirán dos señales desfasadas 90º en el tiempo [Fig. 2. Señales obtenidas por un encoder magnético incremental].

Estas señales dependerán de la cantidad de pares de polos que haya. En el caso de que se emplee un solo imán, la señal tendrá una resolución de dos bits, por lo que se mostrará en pantalla como una señal cuadrada. Sin embargo, cuantos más imanes estén puestos en la periferia del disco o más sensores de efecto Hall haya, mayor será la resolución de la señal. Por ejemplo: en caso de tener 64 pares de polos, y 2 sensores, obtendremos 256 posiciones que equivale a una resolución de 8 bits.

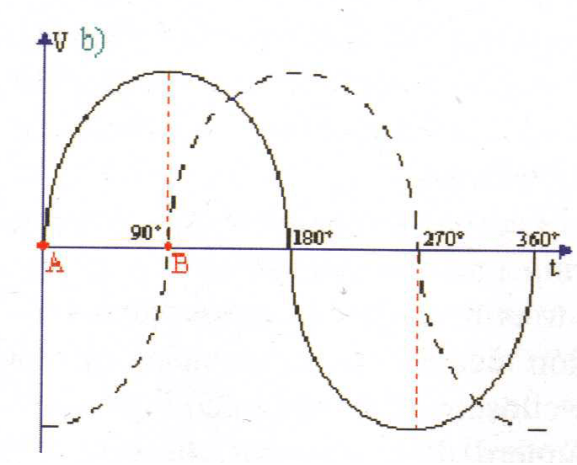


Fig. 2. Señales obtenidas por un encoder magnético incremental [4]

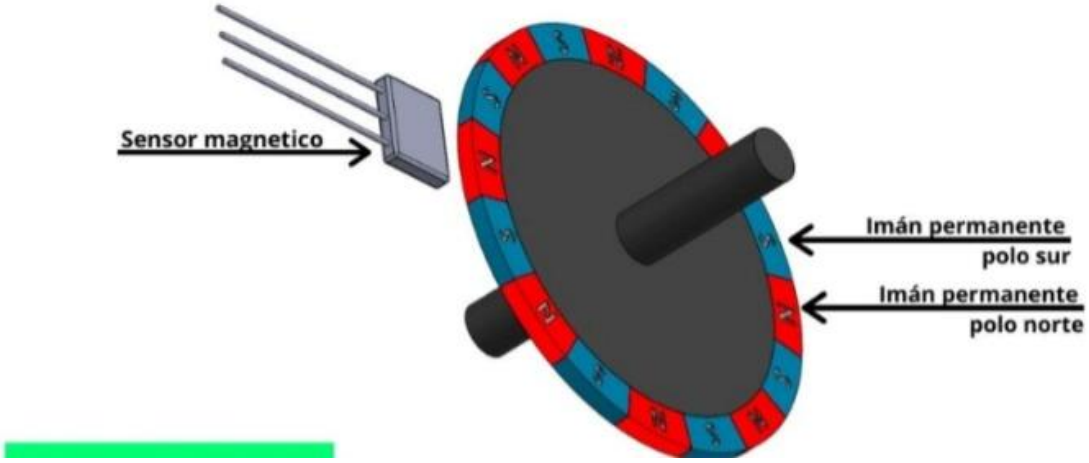


Fig. 3. Encoder magnético incremental [5]

2.1.2 Encoder absoluto

El encoder magnético absoluto constructivamente es mucho más complejo que el encoder incremental, pero también ofrece mayores prestaciones. Una de esas prestaciones, es que no requiere configurar ninguna referencia, ya que no pierden la posición, aunque se interrumpa la corriente.

Los encoders magnéticos absolutos tienen el mismo principio de funcionamiento que los encoders ópticos absolutos, que se explicarán posteriormente. Están formados por un disco que gira solidario al motor y tiene cavidades. El resto del disco es opaco, bloqueando las señales que se pudieran emitir. El patrón que tiene el disco permite obtener una señal digital que nos permitirá conocer tanto la posición como la velocidad y la dirección.

En contra posición a los encoders incrementales, los absolutos proporcionan mayor resolución y orientación general, mejor rendimiento de arranque y mejor recuperación de fallas del sistema o de energía. También proporcionan una mayor información, aunque a veces sea más difícil de interpretar que en los incrementales (ya que sólo tiene una señal A o B en onda cuadrada).

2.1.3 Características de los encoders magnéticos

Ahora que se conocen el funcionamiento de los diferentes encoders magnéticos que podemos encontrar, observaremos una serie de características que muestran para que situaciones serían los más aptos y alguna diferencia frente a otros encoders.

Ventajas:

-Todos los elementos de los que se componen los encoders magnéticos se encuentran protegidos dentro de carcasas sólidas haciéndolos capaces de resistir condiciones agresivas. Esto permite tener una gran resistencia a polvos, temperatura, humedad y vibración.

-Son más ligeros y pequeños que otros encoders, como es el caso de los encoders ópticos.

-No contienen ni producen material inflamable, por lo que son de uso frecuente en aplicaciones donde el riesgo de explosión es muy elevado.

-Tienen un sistema de dirección bidireccional.

-Pueden funcionar con tecnología inalámbrica hasta una distancia de 300 metros.

-Mantienen una alta capacidad de funcionamiento en todo momento.

-Necesitan de poco mantenimiento debido al poco desgaste que hay en las piezas. Todo esto se debe a que los encoders magnéticos no generan fricción entre sus elementos móviles.

-Tiene un precio menos elevado que otros encoders como es el caso de los ópticos. [6]

Inconvenientes:

- Tienen una resolución limitada, que se traduce en baja precisión a la hora de medir.

- Tienen graves problemas en entornos donde puede haber campos magnéticos cercanos ya que pueden interferir en correcto funcionamiento del encoder. [6]

2.2. ENCODER ÓPTICO

Los encoders ópticos tienen un principio de funcionamiento más sencillos que los encoders magnéticos. Estos mismos, pueden diferenciarse entre dos grupos, los lineales y los rotativos. En el proyecto que nos concierne: medir la posición angular y determinar con ella la velocidad del eje del motor eléctrico, se descartarán los lineales.

Los encoders ópticos están formado por unos fotoemisores, un disco perforado, y fotoreceptores. El disco perforado estará compuesto por sectores opacos y sectores transparentes. Por los sectores transparentes será por donde pase la luz emitida por los fotoemisores hasta impactar en los fotorreceptores. Esta información que vendrá de forma intermitente, debido a la alternatividad de los

sectores, nos dará la señal de salida que buscamos, es decir, una señal con la que podamos saber la posición angular del eje del motor.

Dependiendo de cómo se obtiene esa señal de salida podemos encontrar dos tipos de encoders, los incrementales y los absolutos. Aunque tienen las mismas bases que los encoders incrementales y absolutos de los encoders magnéticos, tienen ciertas diferencias.

2.2.1 Encoder Incremental

En el encoder incremental, como bien se ha mencionado anteriormente, las señales vienen en forma de pulsos.

El disco que permitirá este proceso está formado por una corona de sectores transparentes y opacos como se ha mencionado. Sin embargo, los sectores se distribuirán alternativamente, de forma uniforme y equidistante unos de otros. [7]

El eje que se quiere medir se incorporará en el centro del disco, haciendo solidario el movimiento del disco con el del eje. Una vez empiece a girar el disco, la luz que emitirán los foto-emisores pasarán por las cavidades transparentes incidiendo en los fotorreceptores de forma intermitente (debido a los sectores opacos). La cuenta de todos estos pulsos provenientes de la señal emitida por el fotorreceptor finalizará en la obtención de la posición del eje. [Fig. 4. Representación de un encoder óptico incremental.]

Sin embargo, aunque sepamos la posición del eje, también necesitaremos obtener el sentido en el que gira. Para resolver este problema, emplearemos dos foto-emisores, que estarán dispuestos de forma que haya un desfase entre los impulsos de uno y del otro de 90º (como podemos comprobar en la figura 1.3). Para conseguir este desfase, se pondrá un foto-emisor delante de una ranura transparente y al otro en una ranura opaca.

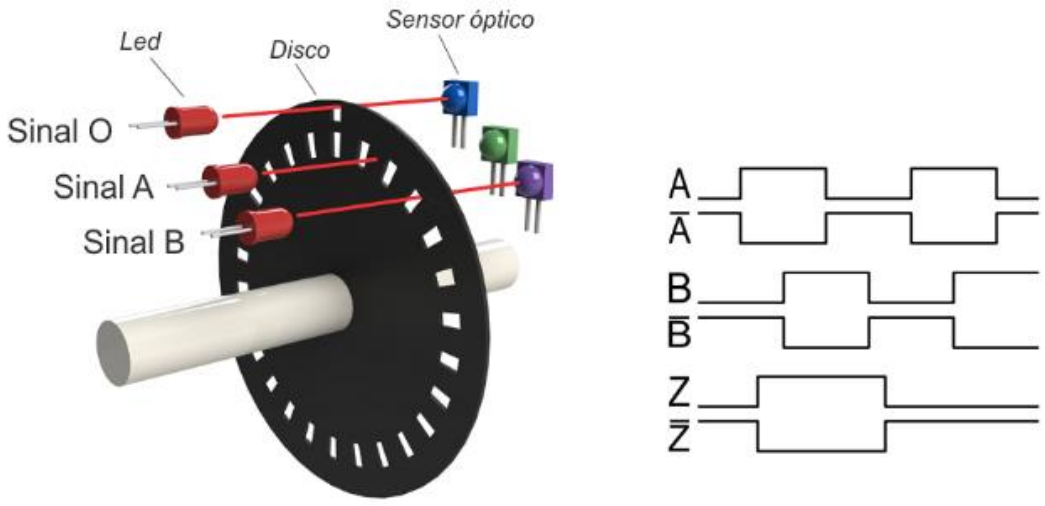


Fig. 4. Representación de un encoder óptico incremental. [19]

Se deberá de disponer de una ranura adicional (marca de referencia) que nos indique cuando se ha dado una vuelta completa y que por tanto comienza la cuenta de nuevo (nos proporcionará la señal Z que se

muestra en la figura 1.3). Esta marca también nos permitirá volver a la posición en que estábamos tras una caída de tensión.

Por último, cabe mencionar que se puede ver claramente la relación que tienen los anchos de pulso con la velocidad angular. Esto se debe a que cuando el motor gira a una velocidad constante, el ancho de pulso es constante; en caso de que el motor aumente su velocidad, el ancho de pulso se hará más estrecho; y en caso de que el motor disminuya su velocidad, el ancho de pulso se hará más amplio. Finalmente, podemos sacar la velocidad angular empleando un reloj y midiendo los anchos de pulso o la cuenta de pulsos entre dos pasos de reloj.

2.2.2 Encoder Absoluto

Al igual que con el encoder magnético absoluto, el encoder óptico absoluto es más complejo de construir que el incremental. Esto se debe a que éste siempre dará la posición angular y que además no necesitaremos de una herramienta especial para poder conocer el sentido de giro.

El encoder absoluto está compuesto por diferentes pistas, obteniéndose así diferentes salidas, equivalentes a la cantidad de pistas que haya. Las señales de salida se corresponden con dos niveles lógicos (0 y 1), de tal modo que el conjunto de señales se corresponderá con un número de codificación en binario o en código gray. [8]

Los encoders absolutos requieren mayor cantidad de fotoemisores y fotorreceptores que los incrementales. Esto se debe, a que deben de tener una pareja de ambos para cada pista. Por contraparte los incrementales como ya hemos podido ver, sólo requerirán de dos parejas para la pista principal y una pareja para la obtención de la señal de referencia.

Hay dos tipos de distribución de sectores transparentes y opacos que permiten la codificación del disco: codificación mediante código binario y el empleo del código Gray.

Primeramente, se analiza la distribución compuesta por el código binario. En esta codificación encontramos que puede haber una diferencia de más de un dígito en dos señales de salidas consecutivas, provocando problemas en la lectura de las señales a altas velocidades de giro (unos bits pueden llegar a cambiar antes que otros).

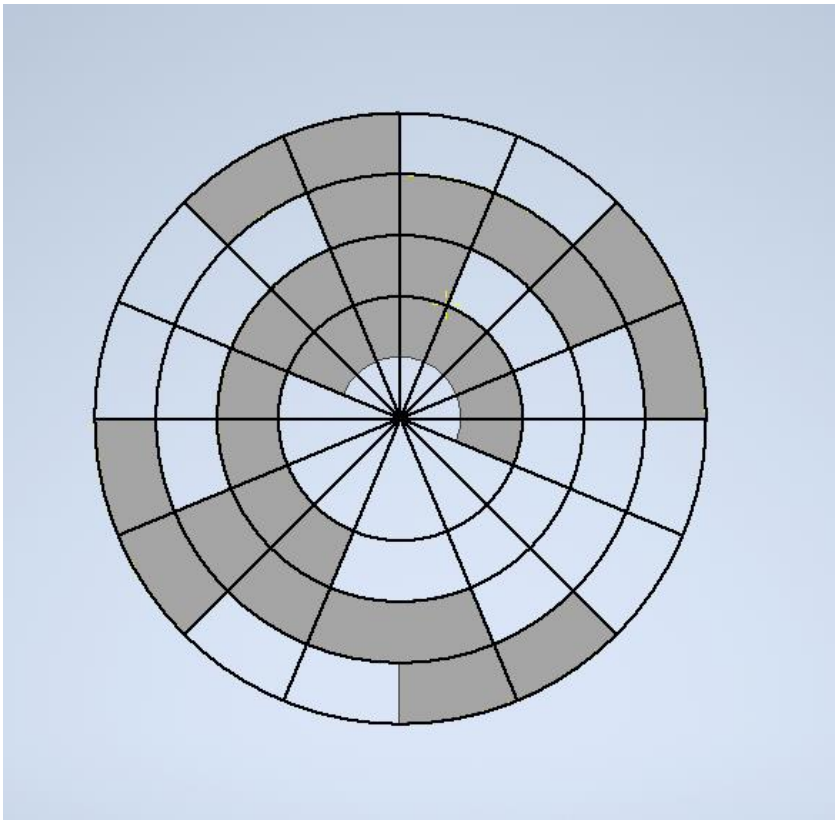


Fig. 5. Representación de un encoder óptico absoluto codificado por código binario (Imagen propia)

Para solucionar el problema mencionado en la codificación por código binario, se empleará el código Gray. Es un código binario especial, cuya principal característica es que entre una combinación de dígitos y la siguiente, sea anterior o posterior, sólo hay una diferencia de un dígito.

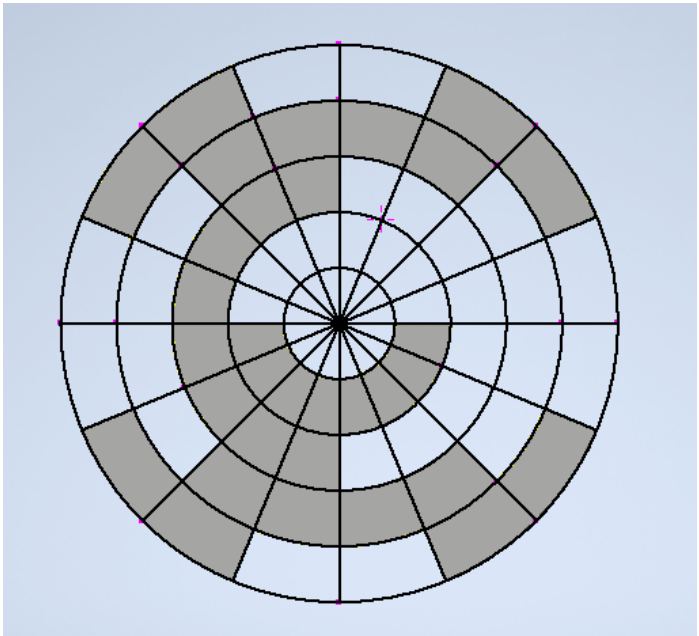


Fig. 6. Representación de un encoder óptico absoluto codificado por código Gray (Imagen propia)

Decimal	Gray code	Binary
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

Tabla 2. Tabla de Gray

2.2.3 Características de los encoders ópticos

-Ventajas:

-Presentan una gran precisión con sus medidas.

-Su funcionamiento es mucho más intuitivo que el encoder magnético.

-Se puede construir de una forma más sencilla, debido a su diseño.

-Desventajas:

-Tienen un precio mucho más elevado que el magnético.

2.3. SELECCIÓN DEL ENCODER

Para este proyecto será necesario un encoder con gran detalle de resolución y precisión. También se necesitará que tenga una estructura que haga visible todo el proceso de medición. Esto permitirá ver exactamente cómo es el funcionamiento de un encoder.

Al contrario que los encoders ópticos, los encoders magnéticos requieren de una construcción más compleja. Esto se debe a que habría que montar un disco integrado con imanes alternativos.

En cuanto a sí se debería escoger un encoder incremental o absoluto, se decantará por un incremental. Esta decisión está fundamentada en que los absolutos requerirán de una mayor cantidad de sensores ópticos como bien se ha comentado en anteriores apartados (requieren un sensor por cada pista).

Con todas estas premisas, la elección más acertada es el encoder óptico incremental.

Los sensores ópticos, elegidos para el encoder óptico incremental son el modelo Interruptor óptico ranurado Optek de un canal, ranura de 5.08mm (OPB916BZ) [9]. Su elección es debido al tiempo de respuesta tan bajo que tienen (3µs), que es menor que los 10 µs que se necesitarían para obtener un buen resultado en las señales proporcionadas por el ESP32. Además, la ranura tan amplia que tienen, permite un margen de error bastante amplio post fabricación.

CAPÍTULO 3: DISEÑO DEL ENCODER

En este capítulo se procederá a explicar los pasos que se han seguido a la hora de realizar el diseño del encoder. Se realizará el estudio de los diferentes limitantes a los que ha estado sometido durante la creación del mismo. Estos limitantes abarcan desde las fuerzas a las que está sometido el encoder hasta la cantidad de resolución que permitirá mostrar. Para ello, se mostrarán los planos de los que se ha hecho uso en su construcción y el proceso de fabricación empleado para llevarlo a cabo. También se mostrará el montaje realizado a posteriori, para la conexión entre el encoder y el microprocesador ESP32, que leerá las señales dadas por este.

3.1 DISEÑO

Primeramente, se empezará con el diseño del encoder. Para la creación de los planos, se empleará el programa de modelaje llamado Inventor. Con este programa, los planos contarán con una alta precisión, imprescindible para su fabricación en 3D y su posterior montaje.

Los planos realizados en Inventor serán cuatro: el plano de diseño del disco perforado, el plano de diseño del soporte, el plano de los sensores ópticos y finalmente el plano de ensamblaje de todo el conjunto de piezas.

3.1.1 Inventor

El programa empleado para el diseño de todos los planos es Inventor.

Inventor es un paquete de modelos paramétricos de sólidos en 3D y pertenece a la empresa de software Autodesk.

Este programa permite realizar planos, y posteriormente darles volumen, para así conseguir piezas en 3D. Una de las mayores ventajas que presenta este software, y el motivo del por qué se ha empleado, es que permite ensamblar las distintas piezas creadas en 3D, dando pie al conjunto final. El ensamblaje permite que se puedan ajustar mejor las medidas, con el propósito de evitar fallos en el montaje a posteriori.

Además, Inventor permite guardar los archivos en “. stl”, formato necesario para importar el proyecto a la aplicación de la impresora en 3D (PrusaSlicer 2.5.0) y por consiguiente imprimir las piezas en 3D.

3.1.2 Plano del disco perforado

Como bien se ha expuesto anteriormente, el encoder que se va a crear será un encoder incremental óptico. Esto deriva en un diseño específico del disco perforado.

Para la creación del disco perforado, primeramente, se deberán de tener en consideración todas las limitaciones que estaban intrínsecas al modelo, que en este caso son 4:

-El diámetro del disco debe de ser de 108 mm, permitiendo así tener un margen de error para aproximar los sensores ópticos.

-El material con el que se realiza, creando un máximo de lo que se puede aumentar el tamaño de las ranuras, al igual que crea un mínimo para el espesor de la pieza.

-La precisión de la Impresora en 3D, que limita la cantidad de ranuras que podrá haber en la pista principal.

-El espacio ranurado que hay entre el fotoemisor y el fotorreceptor del sensor óptico, que limita el espesor del disco perforado (En el presente caso de estudio, queda limitado a 4 mm).

Una vez vistos los aspectos limitantes en la construcción del disco perforado, se procede a hacer un boceto que cumpla con tales requisitos:

Como bien se ha comentado en el apartado 1.2.1, el disco perforado requiere de una pista principal. En el caso presente de estudio, se ha decido hacer una pista de ciento veintiocho ranuras. Hay exactamente ciento veintiocho ranuras ya que debe de haber un número de sectores opacos y transparentes (ambos iguales) que sean igual a dos elevado a un número entero positivo (será la resolución binaria del encoder). La principal causa de esta decisión es que, con esa cantidad de ranuras, se obtendrá una resolución lo suficientemente alta, es decir, se obtendrá una muy buena precisión.

Otra opción hubiera sido tener una mayor cantidad de ranuras, ya que una menor cantidad hubiera sido inadecuado en términos de precisión. Esto se debe a que cuantos más agujeros haya, mayor resolución se obtendrá en la lectura. Sin embargo, la limitación que conlleva el uso de la impresora en 3D empleada impide una mayor cantidad de agujeros. Esta limitación, que incide explícitamente en la precisión de la impresión, repercute en que los agujeros presenten ciertas deformaciones que impidan una correcta lectura de los sensores (esta opción es impensable ya que el modelo de encoder óptico incremental está elegido para presentar unos resultados detallados y precisos). Por otra parte, el problema de la precisión de la impresora también se podría haber resuelto aumentando el diámetro del disco, pero eso será imposible debido a la restricción del diámetro.

Finalizando la parte del ranurado, cabe destacar que la pista estará situada lo más próximo al borde del disco, pero a una distancia prudencial para que no pueda presentar problemas mecánicos. El material del que está fabricado también limita esta distancia.

Una vez hecha la pista principal, se escoge una ranura y se alarga diez milímetros, permitiendo así obtener una marca de referencia. Esta marca sólo podrá ser leída por uno de los tres sensores.

Además del ranurado, se deberá de profundizar en las demás características que definen el disco perforado, como por ejemplo el espesor. Deberá de tener un espesor lo suficientemente grande para no tener problemas mecánicos ni estructurales, y en contraparte, lo suficientemente pequeño para permitir que quepa dentro de los huecos que dejan los sensores ópticos ranurados. El espesor acordado para cumplir con ambas restricciones es de 2,5 mm.

Por último, el disco perforado contará con un cilindro acoplado concéntrico, de un espesor considerable y con un tamaño diferente al agujero realizado para el eje. Este cilindro sirve para acoplar el disco a una parte del eje del motor de mayor tamaño (el acoplamiento elástico que permite el acoplamiento en oposición de los dos motores que forman el banco de ensayos), permitiendo así que el disco sea solidario a éste (el anclaje será con cinta aislante de doble cara. Se explicará posteriormente en el apartado 2.3).

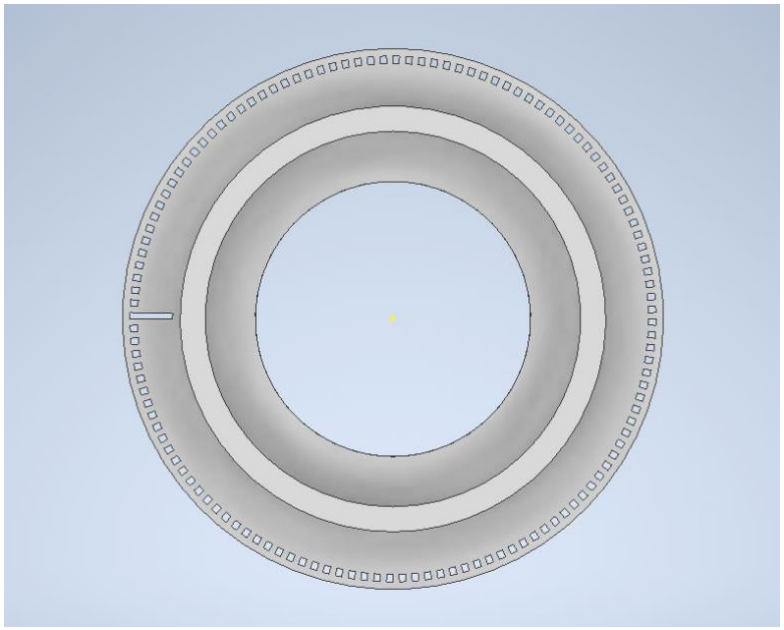


Fig. 7. Disco perforado (Foto propia)

disco de alta resolución 2_4 - 3D model by arnauvicentcrespo (@arnauvicentcrespo) [e183e2e] (sketchfab.com)

3.1.3 Plano del soporte

Para realizar el soporte se han debido tener en cuenta varios factores:

-La cantidad de material que se empleará para poder aguantar el peso de los sensores ópticos, y la cantidad necesaria para que tenga una rigidez suficiente y no se rompa fácilmente.

-El espacio que se requerirá para acoplar los sensores y que entre ellos haya un ángulo suficiente para que se dé el desfase de 90º, anteriormente mencionado. Este ángulo lo obtenemos con la suma de $\pi/4+(360/256) = 46.40625º$.

-Debe de acoplarse a unos espárragos que ya están incorporados al motor como medio de acoplamiento entre los dos estatores y es lo que servirá de base de sujeción del soporte.

La estructura interior del soporte sigue un patrón circular. Esto permitirá una perpendicularidad perfecta entre los sensores ópticos y el disco. El haz de luz debería de tener menos problemas en atravesar el ranurado (mayor precisión en las medidas).

Al tener una forma circular el interior, los sensores no podrán acoplarse bien, debido a su base plana, por lo que se han creado tres pilares que hagan de base. Estos pilares permitirán regular bien la distancia para que el haz de luz incida perfectamente por las ranuras. Los pilares contarán con 6 agujeros pasantes para acoplar los sensores ópticos. Habrá dos agujeros pasantes por pilar para cada sensor, y estarán situados a una distancia uno del otro con el fin de dejar el sensor céntrico. Además de estos agujeros, se diseñarán 3 cavidades para que los cables de los sensores atraviesen el soporte.

La distribución de pilares no será simétrica. El pilar céntrico será el encargado de albergar el sensor óptico que proporcione la señal de referencia, por ello también contará con mayor altura. Los otros dos pilares se situarán uno a cada lado. El primero estará a 22, 5º del central, y el segundo a 23,906º (tendrán ambos la misma altura).

Finalmente, cabe mencionar que los extremos del soporte contarán con dos agujeros de un diámetro mucho mayor que los anteriores. Uno de ellos estará partido a la mitad en un ángulo de 60 grados respecto al eje “y”, para así permitir acoplar y desacoplar el soporte del motor cuando se desee. El otro permitirá que el soporte esté sujeto a los espárragos de acoplamiento del estator.

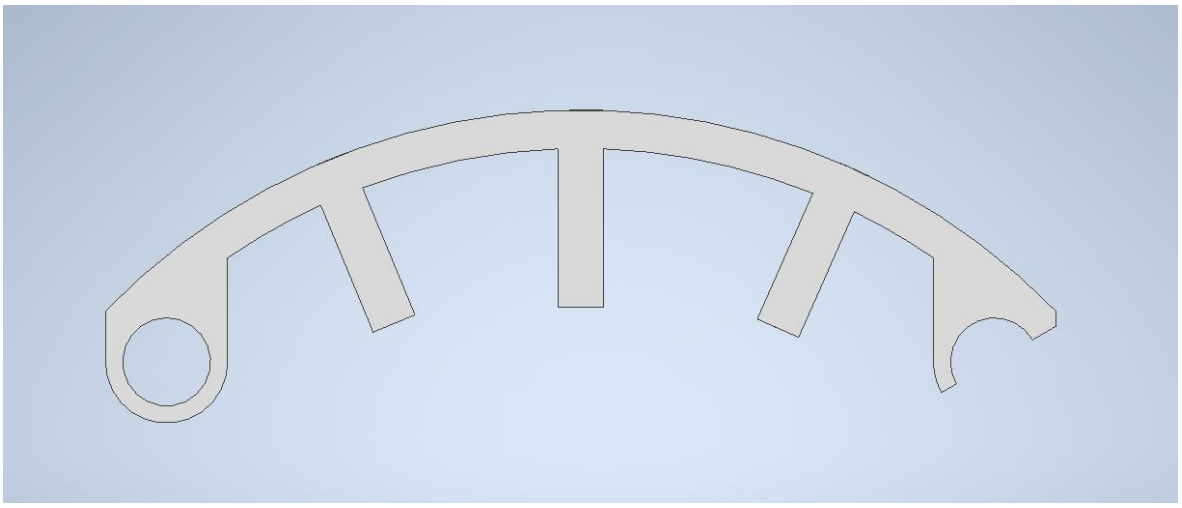


Fig. 8. Soporte

Soporte 2.5 - 3D model by arnauvicentcrespo (@arnauvicentcrespo) [d398e9f] (sketchfab.com)

3.1.4 Plano del sensor óptico

Aunque el sensor óptico sea una pieza que no será necesaria fabricar, se ha modelado en 3D con el propósito de obtener una mayor precisión en la construcción virtual de las tres piezas.

Para su creación se han empleado las medidas proporcionadas por el fabricante, permitiendo realizar un modelo con suma exactitud.

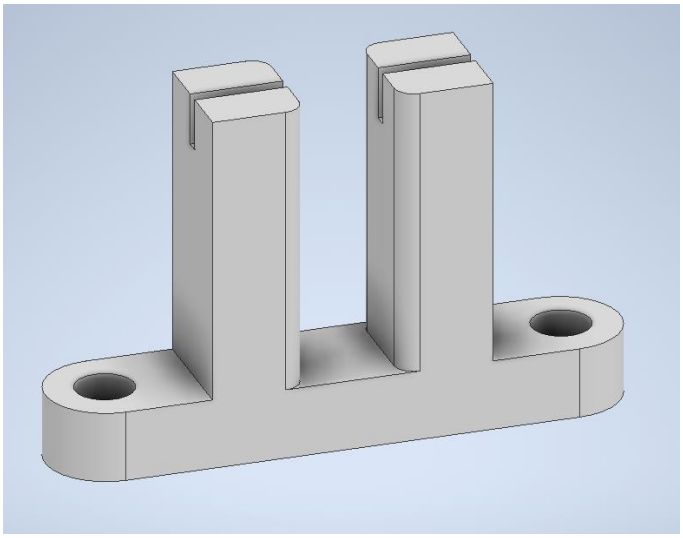


Fig. 9. Sensor óptico

Sensor óptico - 3D model by arnauvicentcrespo (@arnauvicentcrespo) [765ab75] (sketchfab.com)

3.1.5 Ensamblaje

El ensamblaje es una opción que se encuentra en inventor, donde puedes añadir las piezas que desees y montarlas imponiendo restricciones.

Para este proyecto, se ha realizado un ensamblaje para poder hacer mejores ajustes, y que en el montaje una vez imprimidas las partes, no se necesite ningún trabajo extra.

Primeramente, se ha añadido al plano de trabajo el soporte, indicándole que sería el objeto de referencia. A esta condición se le añadirá la restricción de estar fijo.

Luego se han incorporado tres sensores ópticos (esta es la razón por la que se han diseñado en 3D). Con la restricción de coincidencia, se hace que las caras de los sensores sean coplanarias a las caras de los pilares donde se deben de instalar. Por último, para acabar de fijar los sensores, se añadirá la restricción de coincidencia entre los agujeros pasantes de de los mismos junto a los agujeros pasantes de los pilares.

Finalmente, se inserta el disco perforado. Este debe de ser concéntrico con la superficie circular interior del soporte (esto también fue una de las razones para hacer el diseño del soporte circular), por lo que se le aplica la restricción de coincidencia. Sin embargo, el disco requiere de una restricción más para que quede situado en el plano y sólo pueda rotar. Para ello, se le impone la restricción de coincidencia una vez más, pero esta vez, entre una de las caras del disco y una de las caras del sensor óptico. Esta operación cuenta con un desfase de 1,5 mm debido a la necesidad de hacer el ensamblaje realista y así evitar que roce una cara con otra.

Al hacer el ensamblaje permite tener una mayor percepción del montaje en la realidad y así evitar errores que no se hubieran visto hasta la puesta en marcha del montaje. Las cosas que se apreciaron una vez hecho el ensamblaje han sido:

-Se podía ver si exactamente pasaba el haz de luz por las ranuras del disco. Por lo consiguiente, viendo esto, se ha podido comprobar con exactitud si los pilares diseñados estaban a la altura ideal.

-Que los sensores ópticos estuvieran desfasados 90º, es decir, que, en uno, el haz de luz pase por la ranura, mientras que, en el otro, su haz queda bloqueado debido a que está en una parte de la pista opaca.

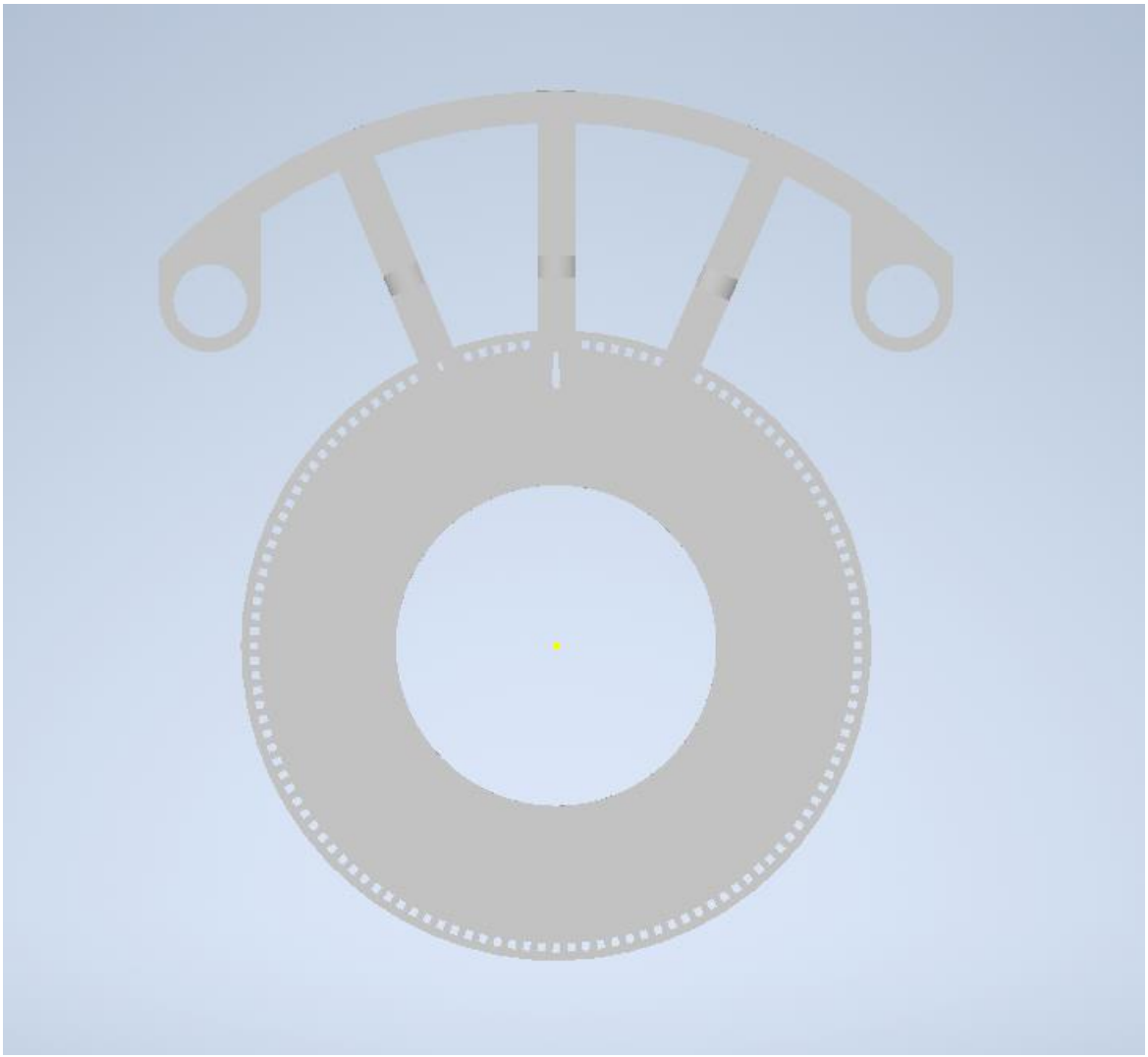


Fig. 10. Ensamblaje del encoder óptico

Ensamblaje de un encoder óptico incremental - 3D model by arnauvicentcrespo (@arnauvicentcrespo) [da03147] (sketchfab.com)

3.1.6. PrusaSlicer

Para la impresión de las piezas creadas en Inventor se procederá a importar los diseños al programa PrusaSlicer.

PrusaSlicer es una aplicación creada exclusivamente para imprimir modelos en 3D en las impresoras Prusa. Esta aplicación permite deliberar cuál va a ser la cantidad de densidad que el material tendrá al igual que las opciones de relleno de éste y así poder ajustar el tiempo de impresión al indicado. [10]

Una vez finalizada la creación de los distintos planos en inventor, se deberán de guardar como archivos ".stl". Se abrirá la aplicación de PrusaSlicer y se importarán los archivos que ya están en un formato correcto.

Para el diseño de laminación del disco perforado, se empleará una densidad de relleno del 100% es decir, macizo, dándole así una perfecta homogeneidad. El patrón de relleno será rectilíneo.

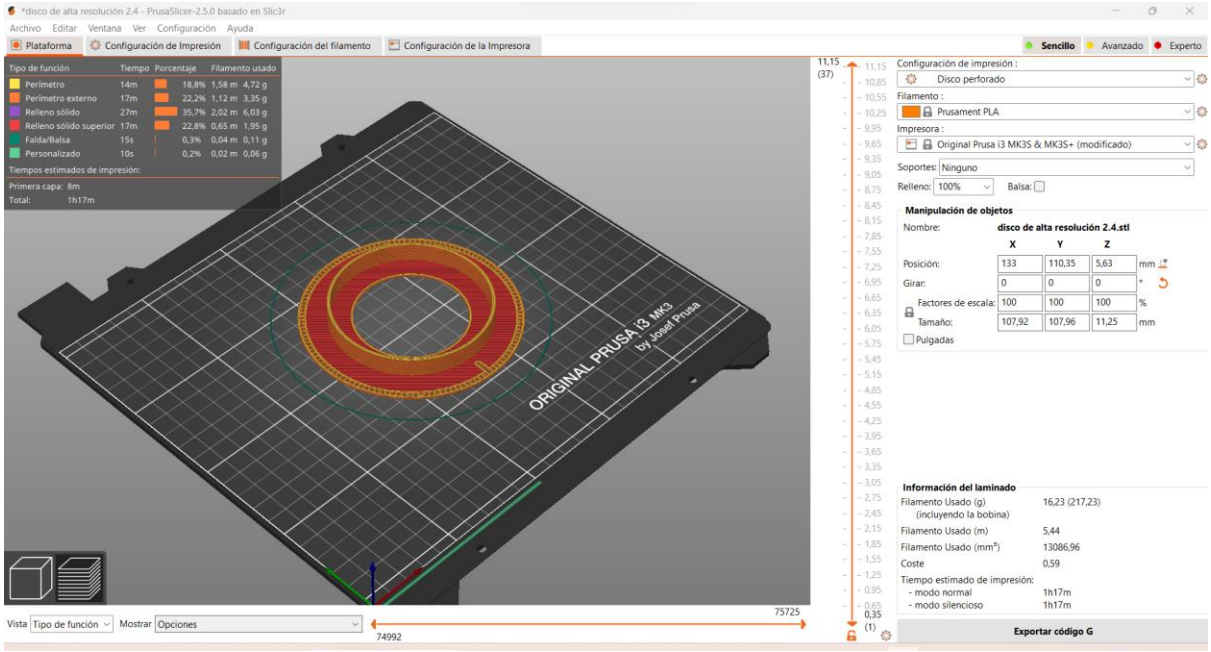


Fig. 11. Laminado del disco perforado

Para el diseño del soporte, se requerirá una densidad de relleno del 10%, ya que debido a sus dimensiones es lo suficientemente resistente. El patrón de relleno que se empleará será giroide. Por último, cabe mencionar, que debido al espesor de la pieza (en caso de la impresión, la altura, ya que la pieza estará tumbada), se deberá de añadir una balsa que permita una impresión sin ningún inconveniente.

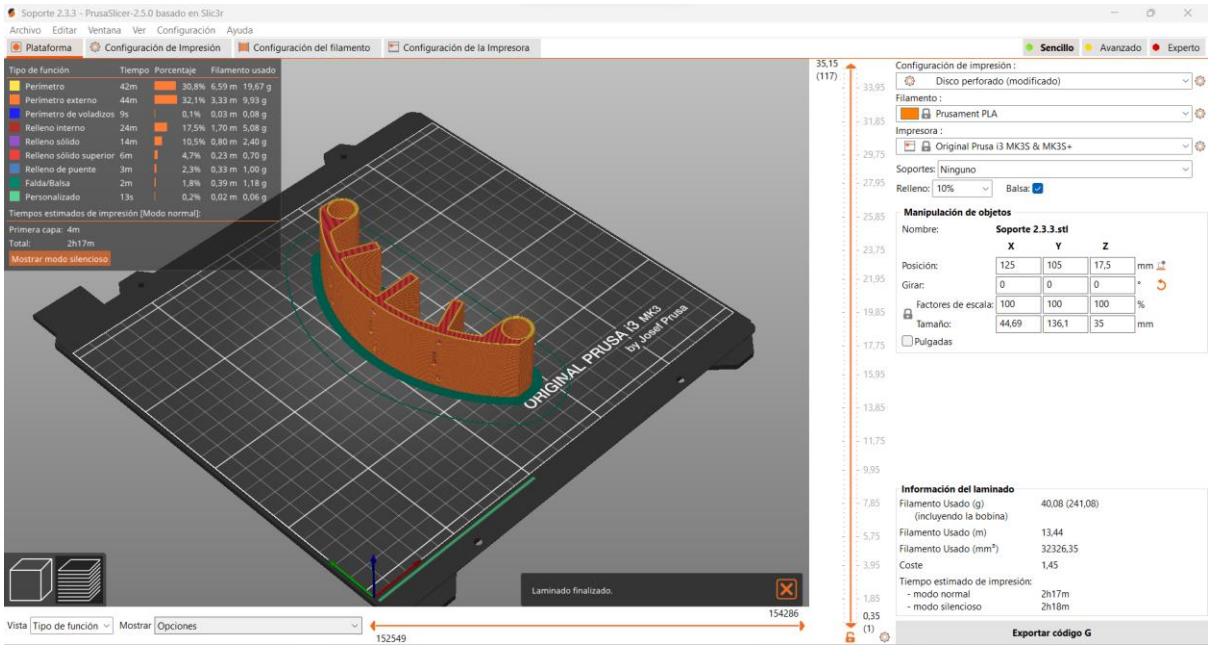


Fig. 12. Laminado del soporte

Finalmente, ya se tienen las dos piezas preparadas para la impresión.

3.2 CÁLCULO DE RESISTENCIA A TRACCIÓN

Para este apartado se verá una de las limitaciones más importantes que se debe cumplir para el correcto funcionamiento del disco perforado.

Siempre se tendrá que realizar un estudio de resistencias para obtener la viabilidad de la pieza, y así poder asignar las dimensiones y material necesario. En el presente trabajo, se debe de hacer un estudio de resistencia a tracción debido a las altas velocidades a las que está sometido el disco. Hay que calcular la máxima tensión de tracción (provocada por la aceleración centrípeta), que puede aguantar la zona más peligrosa, es decir, las uniones en la pista principal.

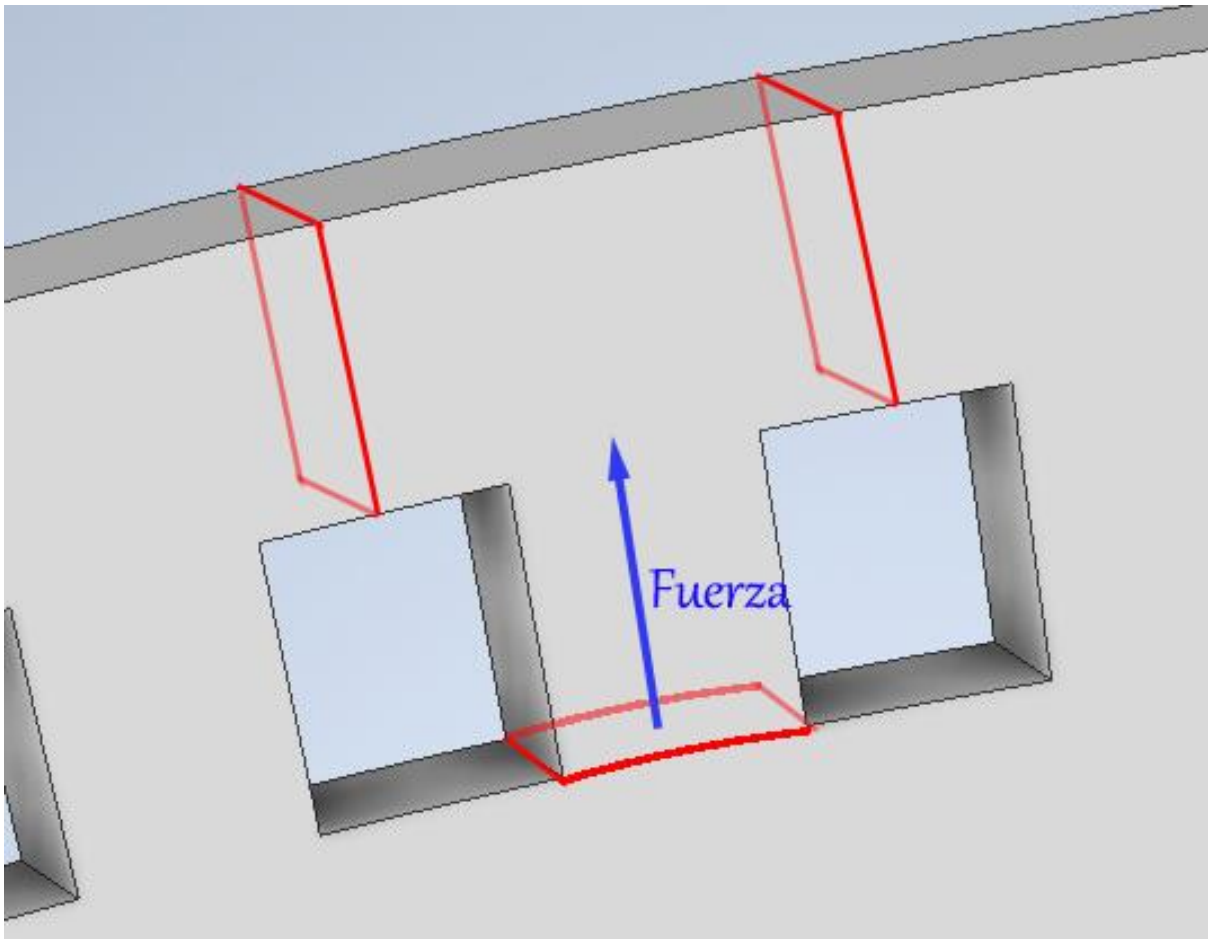


Fig. 13. Diagrama de fuerzas

Para ello, primero se calculará la masa que hay entre la pista, donde está el foco de interés, hasta el final del disco.

$$Mext = \left(\mathrm{Rext}_{\mathrm{disco}}^2 - \mathrm{Rext}_{\mathrm{pista}}^2\right) * \pi * \frac{\mathrm{e}}{\mathrm{Q}} \quad (1)$$

$$Mpista = \left(\mathrm{Rext}_{\mathrm{pista}}^2 - \mathrm{Rint}_{\mathrm{pista}}^2\right) * \pi * \frac{\mathrm{e}}{\mathrm{Q}} * 0{,}5 \quad (2)$$

$$Mtotal = \mathrm{Mext} + \mathrm{Mpista} \quad (3)$$

Radio exterior del disco perforado (Rext_disco) =54 mm

Radio exterior de la pista principal (Rext_pista) =52.5 mm

Radio interior de la pista principal (Rint_pista) =51 mm

Espesor del disco perforado (e) =1.25 mm

Cantidad de agujeros en la pista principal (Q) =128

Masa del disco exterior a la pista que soporta cada unión en la base (Mext)

Masa de la pista que soporta cada unión en la base (Mint)

Masa total que soporta cada unión en la base (Mtotal)

Para calcular la fuerza de tracción a la que está sometida la unión, de la pista con el disco exterior, se realizarán los siguientes pasos:

Primero, se calculará la fuerza centrífuga en la base:

$$F = \frac{\mathrm{Mtotal}}{1000} * \omega^2 * \left(\frac{\mathrm{Rint_{pista}}}{1000}\right) \quad 4$$

Velocidad angular (w)= 100π rad/s

Por último, se calculará la tensión superficial a la que está sometida, para así finalmente, saber si es menor de la tensión limite puesta por el fabricante (33 MPa).

$$P = \frac{\mathrm{F}}{\mathrm{A_{agujero}}} \quad 5$$

Área de la cavidad de la pista principal (A_agujero) =1.565mm^2

Fuerza centrífuga(F)=36.66 N

La tensión superficial calculada es: 23.43 N/mm^2, que es menor que la tensión límite.

3.3 FABRICACIÓN

La fabricación de las piezas se dividirá en dos partes, la impresión en 3D de las piezas, y el acoplamiento de ellas al motor.

3.3.1 Impresora en 3D

Para la fabricación de las piezas se ha empleado una impresora en 3D, exactamente una Original Prusa i3 MK3S+. Se seleccionó este método de fabricación por el gran ahorro económico, ya que en caso de que haya una pieza con un mal ajuste, el precio de hacerla de nuevo no supone una pérdida sustancial.

Este ahorro económico se puede observar igualmente si lo comparamos con otros procesos que tienen la misma finalidad como el moldeo por inyección. Sin embargo, las piezas obtenidas presentan gran grado de detalle, no equiparable al obtenido por moldeo por inyección, pero si el necesario para su función.

Para la elección del material que se emplearía en la impresión en 3D, se ha hecho una comparación, entre material ABS, PLA, ASA, PET y PETG [11]:

-El material ABS:

Es un termoplástico que contiene una base de elastómeros a base de polibutadieno. Esta composición permite una mayor resistencia a los golpes y una mayor flexibilidad. Aunque se puede encontrar el material en forma de polvo o líquida, la empleada para la impresión en 3D será la sólida, es decir, el filamento de ABS.

Para su empleo en la impresión en 3D requerirá una temperatura entre 230ºC y 260ºC, por ende, también necesitará un equipo de impresión en 3D más avanzado. Este material tiene una gran capacidad de soportar temperaturas muy adversas, ya sean bajas temperaturas como -20ºC o altas temperaturas como 80ºC. Estas últimas características desembocan en que es un material un poco más caro que otros, como por ejemplo el PLA.

En resumen, es un material con un alto punto de fusión, resistente, flexible.

Fig. 14. Material ABS [12]

-El material PLA

También conocido como ácido poliláctico, es un termoplástico biodegradable y no tóxico que se produce a partir de recursos naturales renovables, como el almidón de maíz, la caña de azúcar y la remolacha azucarera. Al no ser tóxico, no emite gases nocivos durante la impresión, lo que lo hace más seguro.

Una de las principales ventajas respecto a otros materiales de impresión que se emplean, es que, este tiene un bajo punto de fusión. Para calentarlo se requiere una temperatura entre 190ºC a 230ºC. Es un material difícil de tratar debido a su velocidad de enfriamiento.

Tiene una gran debilidad al agua y es más frágil que otros materiales, lo que lo hace susceptible a la rotura en comparación a otros materiales.

En resumen, es un material consistente pero frágil, que tiene un bajo punto de fusión, no es muy adecuado para exteriores y produce piezas de buena calidad de acabado y una excelente precisión dimensional.

Fig. 15. Material PLA [13]

-El material ASA

También llamado Acrilonitrilo Estireno Acrilato, es un material termoplástico, con propiedades similares al ABS. Es un material pensado para la sustitución de este último, ya que tiene unas excelentes propiedades físicas y químicas.

Es muy resistente a los ambientes exteriores (resistente al agua) y a los rayos UV, que lo hace ideal para piezas expuestas a la luz solar. Tiene una gran resistencia a productos químicos. Es robusto mecánicamente, lo que permite que sea perfecto para aplicaciones con necesidad de resistencia. Presenta un buen acabado de piezas y presenta una mayor resistencia térmica que el ABS.

Finalmente cabe mencionar que se deberá de calentar la plataforma con anterioridad para evitar deformaciones.

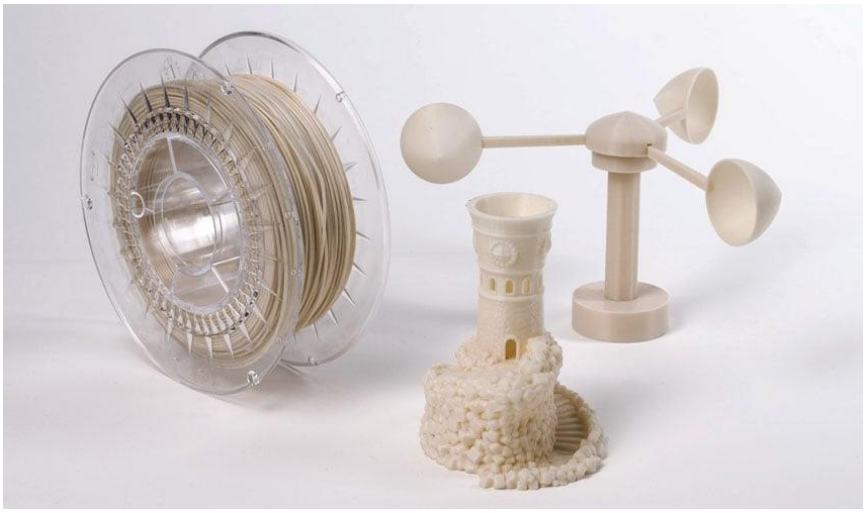


Fig. 16. Material ASA [11]

-El material PET

El tereftalato de polietileno es un plástico exclusivo para piezas destinadas al contacto con alimentos.

Presenta rigidez, y una gran resistencia química. La impresión debe de realizarse entre 75ºC y 90ºC.

Por último, cabe destacar que de este material parten muchas variantes, como el PETG, PETE y PETT.

Fig. 17. Material PET [12]

Finalmente, se procede al proceso de elección.

En primera estancia se tiene el material ABS, que tenía características bastante acordes a las necesitadas para la impresión, sin embargo, el precio extra que se debe de pagar por ellas no compensa. Debido a ello, se descarta este material.

Seguidamente al ABS, se encuentra el ASA, que presenta una mayor cantidad de mejoras. Sin embargo, las características nuevas que aporta este material son innecesarias, ya que el banco de ensayos donde se pondrán las piezas se encuentra en un ambiente cerrado. Debido a ello, las piezas no estarán sometidas ni a los rayos UV ni a ninguna condición climática (no requerirá que tenga buena resistencia al agua). Por consiguiente, tampoco estarán sometidas a ningún producto químico.

Luego se encuentra el material PET. Presenta características que no son necesarias, y ya que este material está hecho exclusivamente para envases de alimentos, queda descartado.

En último lugar queda el material que se escogerá para la impresión en 3D. Es un material barato (más barato que el ABS, y el ASA), es rígido, es fácil de trabajar con él y está desarrollado para crear objetos de pequeñas dimensiones (como es en este caso). Tiene ciertas desventajas, como podrían ser la baja resistencia térmica, aunque es una característica que no influye; y lo quebradizo que es el material, aunque presenta una resistencia a tracción más que suficiente (33MPa). [14]

La impresión de las piezas ha tenido una duración total de cuatro horas y cuarenta minutos: Dos horas y diez minutos el disco perforado, y dos horas y media el soporte.

El tiempo de impresión de ambas piezas no es muy distinto debido a la diferencia de densidad que se ha puesto en cada una de ellas. Como se ha comentado anteriormente, el disco perforado es macizo, por lo que su impresión durará más que si tuviera una menor densidad.

3.3.2 Montaje

Una vez obtenidas las piezas, lo primero será la incorporación del disco perforado en el eje. Para ello, también se deberá de acoplar al buje de bloqueo cónico. Aunque el diseño ha sido preciso, se ha tenido que dar una vuelta a la superficie de contacto en el buje con cinta adhesiva, para aumentar ligeramente el diámetro del alojamiento del disco y que el disco perforado tuviera un movimiento solidario con el eje.

Posteriormente, el soporte se incorporará en dos barras paralelas al eje del motor y a una distancia de la cara frontal del estator que permita que el disco perforado se encuentre en su mediatriz (el centro de los espacios dejados por los sensores ópticos). Tras tener toda la estructura montada, se procederá a incorporar los sensores ópticos.

Los sensores ópticos se incorporarán al soporte a partir de 6 tornillos de 3.2 mm de diámetro. Además, se deberán de pasar los cables de los tres sensores por las cavidades hechas en el soporte.

3.3.2.1 Circuito eléctrico

Para hacer funcionar los sensores ópticos se empleará una fuente de alimentación de 5V, debido a ello, se deberá de calcular la resistencia que habrá que incorporar al circuito para que la intensidad que les llegue a los sensores no les cause ningún problema.

Teniendo la hoja de especificaciones conocemos que el consumo de cada uno de los sensores es de 1.3 V, y que la intensidad máxima que pueden soportar estos son 7 mA.

El circuito se realizará en serie, ya que será preferible que todos los sensores se activen simultáneamente. Al ser en serie, la resistencia que se pondrá en el circuito deberá de ser aproximadamente de 158 Ohms. Se elige un valor normalizado próximo al calculado de 150 Ohms.

Además, se deberá de tener en cuenta que los sensores ópticos funcionan como diodos por lo que se deberá prestar especial atención a su conexión: Ánodo con cátodo, ánodo con extremo positivo de la fuente de tensión y cátodo con extremo negativo de la fuente de tensión.

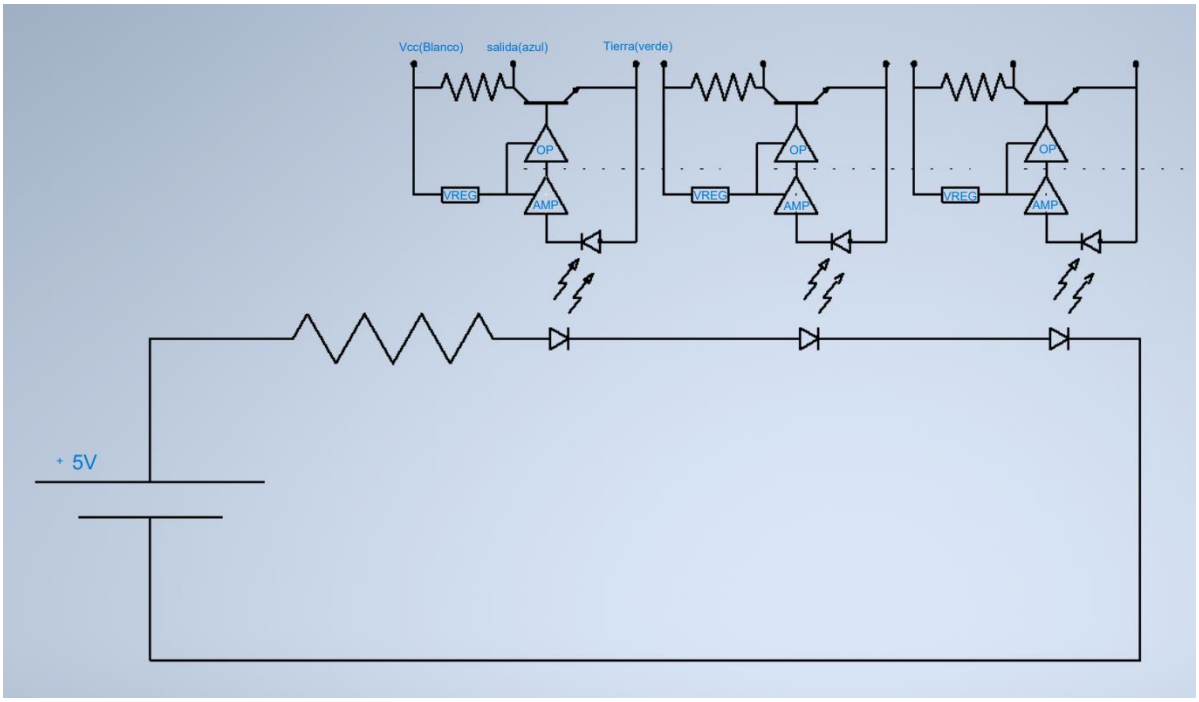


Fig. 18. Circuito eléctrico

3.3.2.2 Piezas de la placa

Una vez calculada la resistencia que se empleará y conociendo la tensión a administrar habrá que realizar el montaje de todos los elementos en una placa electrónica. Para ello, se requerirán las siguientes piezas:

-Una placa electrónica de 24x18. Esta placa permitirá el acople de los elementos empleados y facilitará la conexión entre ellos.

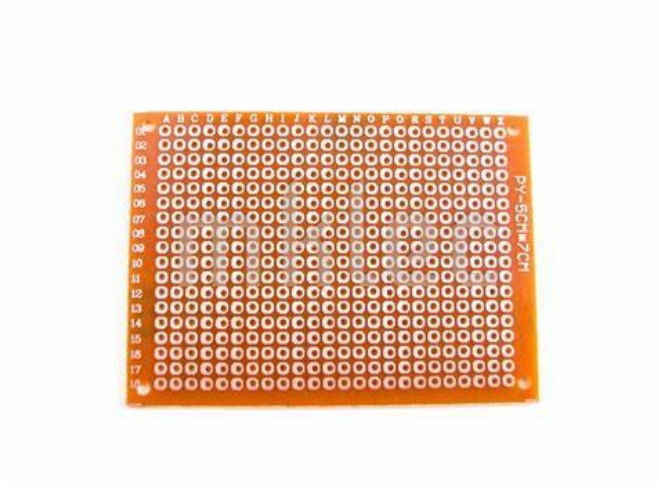


Fig. 19. Placa electrónica (Foto propia)

-Dos módulos de bornes de conexión de tornillo, con cuatro celdas. Estos bornes de conexión de tornillo asegurarán una buena conexión entre los cables y la posibilidad de separar la placa del conjunto portante de los sensores ópticos para ajuste o mantenimiento. Se emplearán dos, ya que se harán ocho conexiones:

Fig. 20. Bornes de conexión de tornillo (Foto propia)

-Dos pines de encabezado de 16 conexiones cada uno. Necesarios debido a la cantidad de calor que se emitirá cuando se tenga que soldar los cables de los sensores ópticos (los cables que transmiten la señal desde el sensor óptico) a los pines de entrada de la placa ESP32. Actuarán como enlace para que no se pueda perjudicar a la placa ESP32 durante la soldadura y, al mismo tiempo, se pueda integrar ésta sencillamente en el conjunto.

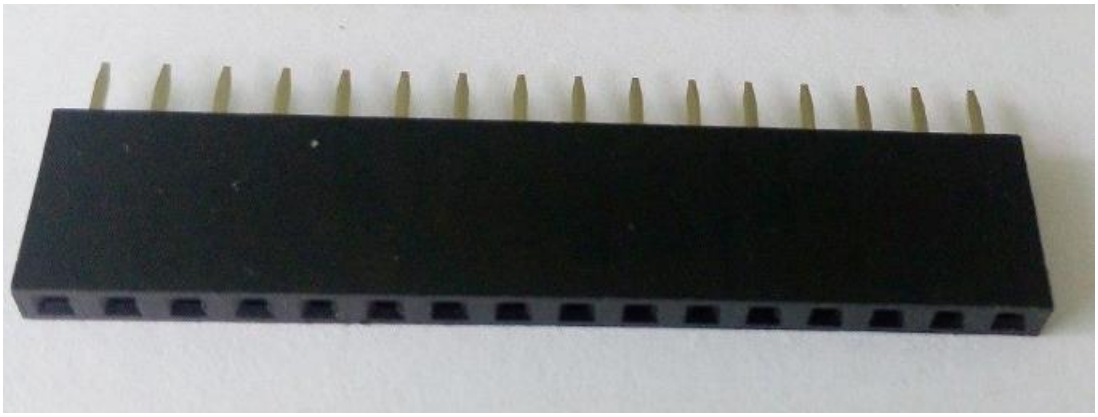


Fig. 21. Header pins (Foto propia)

Por supuesto, no puede faltar la placa ESP32 que irá acoplada a la tira de pines de encabezado.

Además de las piezas mencionadas, también se necesitarán cables para efectuar las conexiones necesarias y un soldador para conectar los pines necesarios del ESP32 a los sensores ópticos.

3.3.2.3 Montaje de la placa

A continuación, se procederá a explicar cómo se ha hecho el montaje de la placa que permitirá el correcto funcionamiento de los sensores ópticos.

Se empezará conectando todas las conexiones vistas en la Fig. 18. Para ello se emplearán los bornes de conexión de tornillo, que al tener dos, se dispondrá de un total de 8 celdas. De izquierda a derecha seguirán el siguiente orden:

En la primera celda se conectarán el borne negativo de la fuente de tensión, el cátodo del último sensor conectado en serie y las tres tomas a tierra de los tres sensores (identificadas con el color verde).

En la segunda, tercera y cuarta celda se conectarán los terminales de los sensores ópticos que llevan las señales.

En la quinta celda se conectará el ánodo del primer sensor óptico conectado en serie.

En la sexta celda se conectará el cátodo del primer sensor con el ánodo del segundo sensor.

En la séptima, de igual forma, se conectará el cátodo del segundo con el ánodo del tercero.

Finalmente, en la octava, se conectará el borne positivo de la fuente de tensión con los terminales correspondientes a la tensión de alimentación de los sensores ópticos (se identifican con el color blanco).

La resistencia calculada con anterioridad se conectará con soldadura con la celda quinta y la octava.

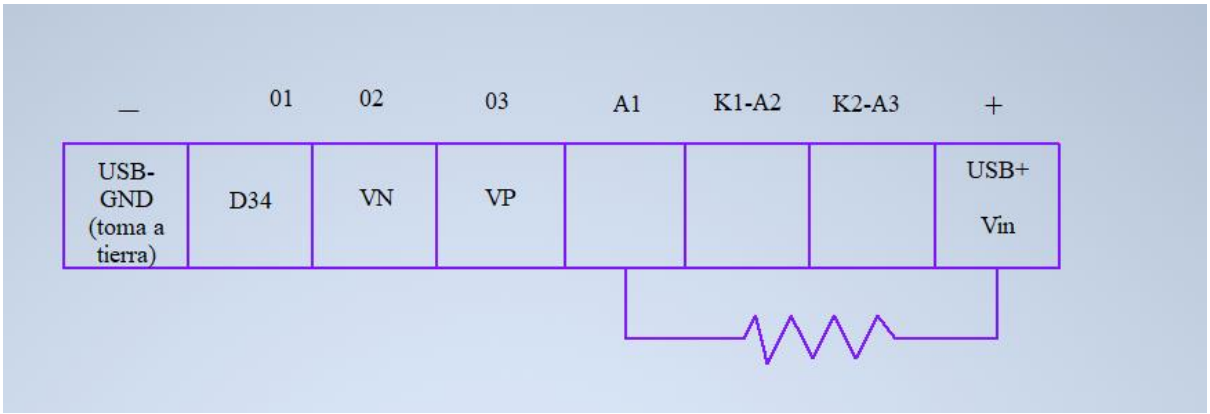


Fig. 22. Esquema de conexiones en bornes de conexión de tornillo

Por otra parte, el soldador se empleará para efectuar las conexiones de los cables situados en las celdas segunda, tercera y cuarta con los pines del ESP32. Los pines a conectar serán de izquierdas a derechas, el pin D34, el pin VN y el pin VP. Al primer pin se le asignará la señal de referencia para conocer el tiempo por vuelta, al segundo pin y el tercero las señales que están desfasadas 90º digitales.

3.3.3 Puesta en marcha y validación

Antes de emplear el ESP32 se deberá de comprobar si el montaje efectuado ha sido correcto o si ha habido algún error tanto en la precisión de montaje de los sensores ópticos, en el diseño del disco perforado o del mismo soporte. Para ello, se empleará un osciloscopio, para poder ver las señales proporcionadas por los sensores y así concluir si el proyecto ha funcionado según lo previsto o no.

Se procederá a ver las tres señales necesarias, se deberá de poder apreciar un desfase de 90º entre las señales que corresponderían a los pines VP y VN, que la señal aportada por el sensor de referencia (correspondiente a la señal G34) sólo tenga un flanco de subida y de bajada por vuelta, y por último, que las tres señales sean cuadradas.

Comprobación de que el sensor de referencia funciona:

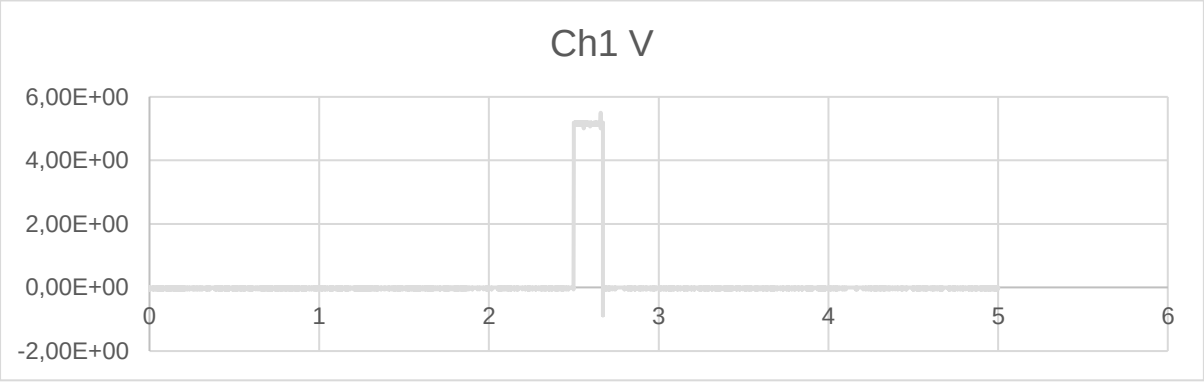


Fig. 23. Señal del sensor de referencia mostrada por un osciloscopio

Comprobación de que funcionan correctamente los sensores, viendo que están desfasadas las ondas aproximadamente 90º una de otra.

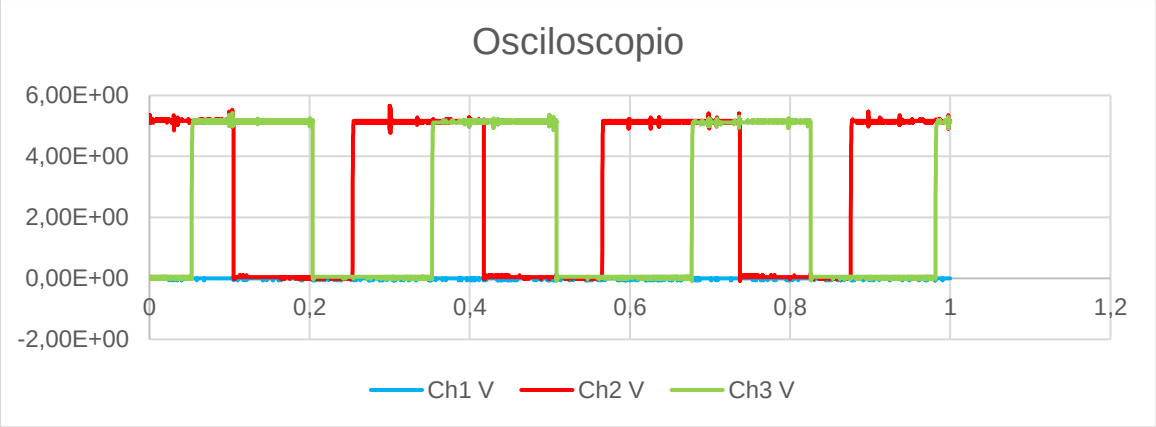


Fig. 24. Señales de los sensores ópticos mostradas por el osciloscopio

Finalmente, se corrobora que el montaje ha sido correcto y que la información que reciba el ESP32 será la prevista en el diseño. El desfase entre las señales de los canales 2 y 3, que se corresponden con las señales A y B del encoder óptico, se puede comprobar que no es exactamente ¼ del periodo de cada una o 90º eléctricos. Ese sería el caso ideal, pero no es necesario llegar a una precisión excesiva en ese desfase. Conseguir una precisión mayor exigiría una precisión de montaje de los sensores ópticos en su bastidor que no es fácil de alcanzar con el material y boquillas disponibles para la impresión 3D. El hecho de que el desfase sea inferior a 90º sólo incide en el diseño en que, a la hora de detectar la secuencia de flancos en el cambio de las entradas del ESP23, se tendrá que asegurar un tiempo de lectura consecutivo que sea inferior al desfase mínimo observado convertido en tiempo a la velocidad máxima de operación del sensor, lo que no es problemático dada la alta capacidad de procesamiento del ESP32 en operaciones de lectura digital por comparación con la frecuencia de las señales de los canales A y B ópticos a la velocidad de operación máxima.

CAPÍTULO 4: PLACA ESP32, FUNCIONAMIENTO Y BASES

En este capítulo se procederá a explicar que es el ESP32, sus características y sus funciones. También se hará un inciso sobre la plataforma que se empleará para trabajar con la placa ESP32, explicando brevemente por qué se ha seleccionado y el cómo será trabajar con ella.

4.1 ¿QUE ES EL ESP32?

El ESP 32 proviene de la misma familia que el ESP 8266 y fue creada por Espressif Systems. El ESP 32 es un SoC (System on Chip). Contiene la mayoría de los componentes que tiene un ordenador. Una de las singularidades más importantes que presenta, es que permite conectarse tanto por vía Wifi como por vía Bluetooth (conectividad integrada ya en la placa), haciéndolo un dispositivo perfecto para proyectos de IoT (Internet of Things). También contiene un procesador gráfico, memoria de trabajo, interfaces… [15]

El ESP 32 está compuesto por el siguiente diagrama de bloques:

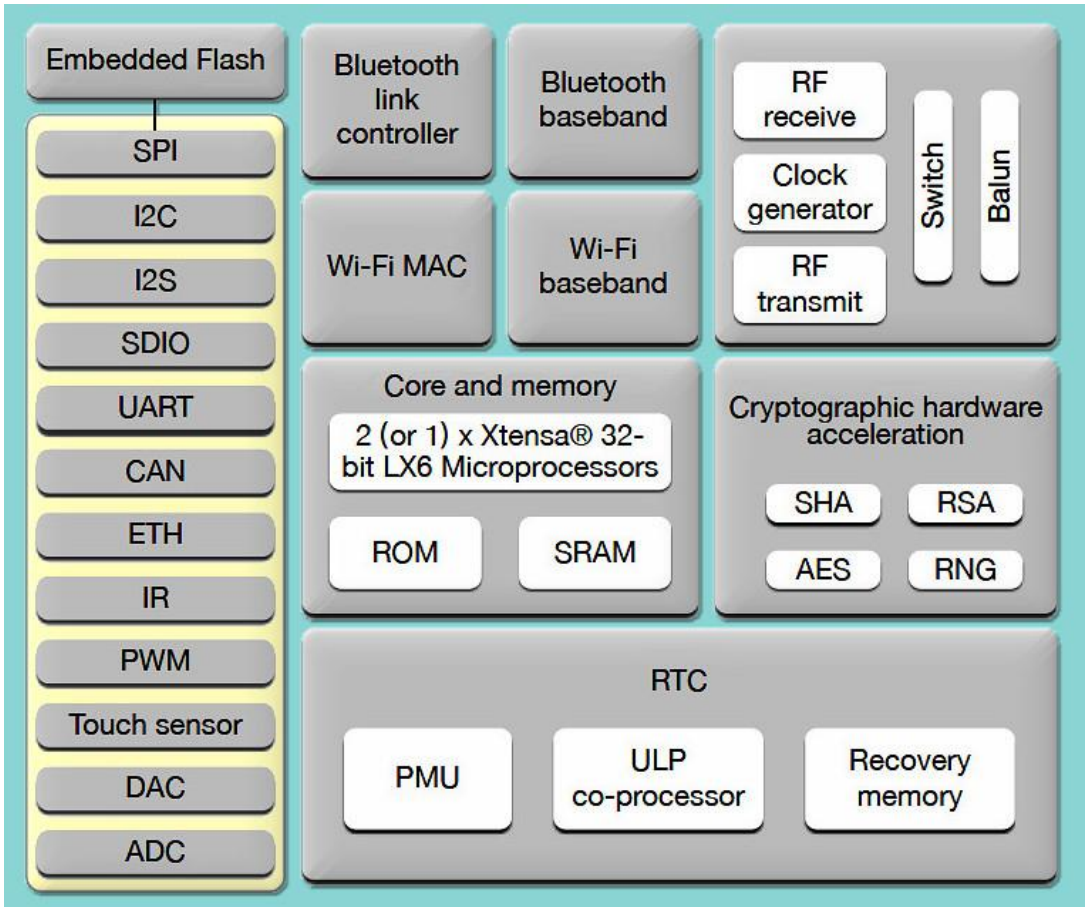


Fig. 25. Bloques del ESP32 [15]

4.2 CARACTERÍSTICAS

El ESP32 tiene una cierta cantidad de características que lo han hecho ideal para este proyecto. [16]

4.2.1 Diseño robusto

El ESP32 es capaz de evitar que sea dañado por fallos en circuitos exteriores, y además no tendrá ningún problema de funcionamiento debido a que trabaja con altas temperaturas.

4.2.2 Ahorro energético

A causa de que está creado para IoT, el ESP32 tiene un bajo consumo de energía. Esto se debe a que el chip integrado tiene un coprocesador de baja potencia que se emplea para sustituir a la CPU en funciones que no requieren mucha potencia informática y así ahorrar energía. Un ejemplo de ello sería la comprobación de periféricos.

4.2.3 Alto nivel de integración

El ESP32 tiene la capacidad de tener una arquitectura de doble núcleo que permite ejecutar dos procesos simultáneamente. Además, cuenta con una velocidad de reloj de hasta 240 MHz y 520 KB de SRAM.

4.2.4 Conexión

Como se ha comentado anteriormente, a diferencia de una placa Arduino, el ESP32 tiene la conectividad integrada ya en la placa. Esto permite interactuar con otros sistemas para proporcionar funcionalidad Wi-Fi y Bluetooth sin costes adicionales y contando con numerosos ejemplos de integración disponibles.

4.2.5 Programación

Una de las mayores ventajas de emplear el ESP32 es que se puede programar con el lenguaje de programación de Arduino IDE (c). Esto implica una gran cantidad de información para el desarrollo del actual proyecto además de una gran facilidad para su programación (al ser un lenguaje de programaciónmuy extendido y una solución típica de sistemas embebidos).

4.3 PLATAFORMA DE TRABAJO

Para trabajar con una placa ESP32, se empleará el editor de código fuente desarrollado por Microsoft, llamado Visual Studio Code. Este editor cuenta con la facilidad de trabajar con diferentes tipos de placas compatibles con Arduino, tanto las placas oficiales de Arduino como otras placas provenientes de otros fabricantes como es nuestro caso.

En este caso, también se podría haber trabajado con el entorno de desarrollo Arduino IDE, sin embargo, por comodidad se empleará Visual Studio.

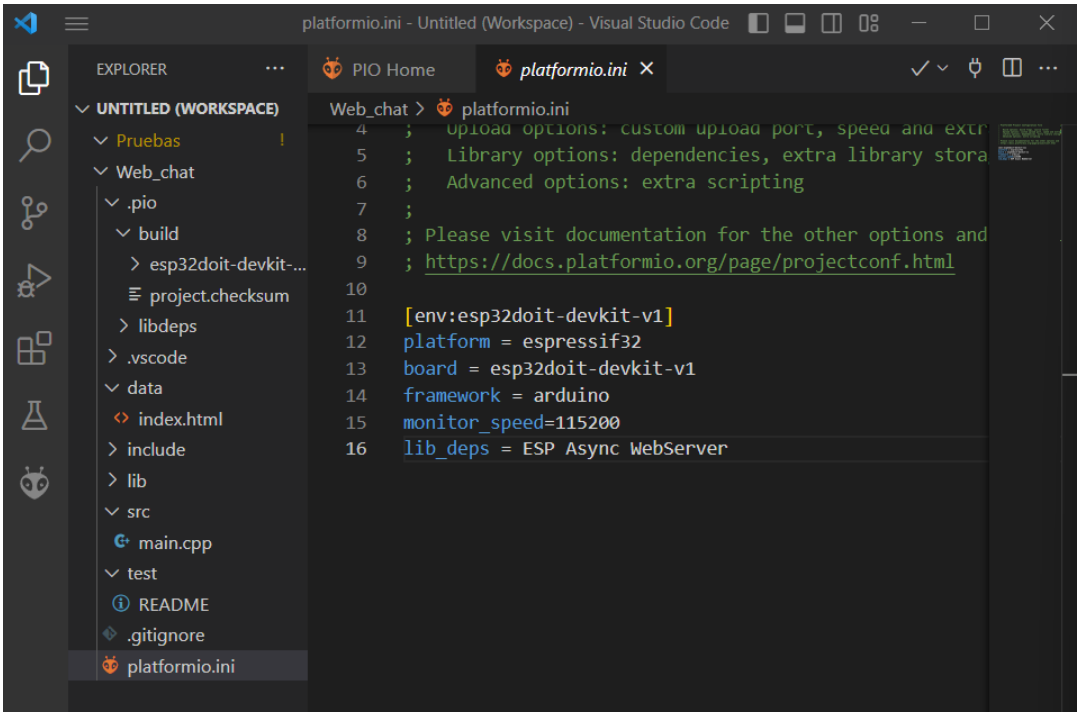
Por contraparte, Visual Estudio es compatible con una extensión de Arduino. Aunque no contenga el entorno de desarrollo integrado (IDE), permite proporcionar las herramientas necesarias para desarrollar y cargar código en Arduino, como se ha mencionado anteriormente. Esto incluye la depuración integrada y la gestión de bibliotecas.

Visual Studio Code admite una gran variedad de lenguajes de programación como Python, JavaScript, Java, C, C++, PHP…. Los que se emplearán con la extensión de Arduino serán principalmente C y C++.

4.3.1 Preparación

El primer paso por realizar será descargar el Visual Studio Code. Una vez descargada, se deberá instalar el paquete PlatformIO IDE (versión 3.1.1). Este paquete permite trabajar en Visual Studio Code, como si se estuviera trabajando en Arduino IDE.

Posteriormente, se deberá de añadir cierta información en Visual Studio Code, más exactamente en el apartado de platformio.ini, para que permita operar con el microcontrolador ESP32:



Código 1. Pantalla explicativa del Platfotmio.ini

En “platform” se pondrá el microcontraldor con el que se va a trabajar, es decir, la placa ESP32 como se ha comentado con anterioridad.

En la función “board” se pondrá la placa de desarrollo DOIT Esp32 DevKit V1, ya que integra el módulo ESP-WROOM32 que es el que se encuentra en el ESP32 con el que se hará el estudio.

En “framework” se pondrá arduino, debido a la facilidad con la que permite trabajar, además de todas las librerías que son compatibles.

Se deberá de poner cuál será la velocidad del monitor “monitor_speed”. Está velocidad suele ser de 115200 baudios (Cantidad de bits que pueden ser transmitidos por segundo) ya que es una velocidad de transmisión de datos relativamente alta y estable, lo que permite una comunicación rápida y fiable con el microcontrolador ESP32 y el ordenador al que está conectado.

4.3.2 Funcionamiento

Lo primero que se encuentra al entrar en Visual Studio Code, tras haber abierto el programa nuevo con <Platformio>, son las funciones de “void setup” y “void loop”. Estas funciones son el pilar dentro del código ya que, si faltara una de ellas, el programa no permitiría compilar. Si se compara con el programa C/C++, estas funciones serían el equivalente a la función main, sin embargo, en este caso hay dos.

-La función “void setup”, sólo se ejecutará una vez al principio del programa.

-La función “void loop” se ejecutará repetidamente hasta que se apague la placa o se reinicie el programa de Arduino.

Un detalle bastante importante, es que las funciones “setup” and “loop” están precedidas por la palabra “void”, que representa que dicha función no presenta ningún valor de retorno.

```
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

Código 2. Funciones básicas arduino

CAPÍTULO 5: CREACIÓN DE WEB Y LECTURA DE DATOS ASÍNCRONA

En este capítulo se explican los pasos para realizar una página web y para poder ver de forma asíncrona los datos aportados por los sensores ópticos y así obtener la gráfica que indique en un histórico la velocidad a la que gira el motor.

Cabe destacar, que para la creación de la web que permite ver los datos de forma asíncrona en una gráfica, el código empleado ha sido basado en un código ya creado en la página web Random Nerds Tutorials. Sin embargo, han sido modificados varios aspectos para permitir su correcto funcionamiento. [17]

5.1 INICIALIZACIÓN DEL PROGRAMA

El primer paso será añadir las bibliotecas que se emplearán en el programa:

```
#include <Arduino.h>
#include <Wire.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFS.h>
#include <WiFi.h>
```

Código 3. Bibliotecas empleadas

La primera en encontrarse es <Arduino.h>, que incluye funciones para configurar el microcontrolador, definir constantes, tipos de datos y funciones básicas para el control de E/S (entrada/salida). Además, también incluye la función “setup()” y “loop()”. Esta biblioteca permite trabajar de forma similar al entorno de desarrollo de Arduino, Arduino IDE.

La biblioteca que se encuentra en la segunda línea de código es: <Wire.h>, que permite comunicarse con dos dispositivos por bus I2C (Inter-Integrated Circuit). Permite la comunicación entre un controlador y circuitos periféricos integrados.

La biblioteca incluida en la tercera línea de código es: <ESPAsyncWebserver.h>, se emplea para el desarrollo de aplicaciones Web en el ESP32 al igual que su versión antigua el ESP8266, empleando el framework (marco de desarrollo) de Arduino. Esta biblioteca emplea el modelo de programación asíncrono, lo que indica que no se bloqueará la ejecución del programa mientras se esté manejando una petición HTTP. Esta biblioteca será fundamental para el desarrollo de la página web y para el cálculo en tiempo real de la velocidad del motor.

La biblioteca <SPIFFS.h>, permite emplear directamente un archivo con la información que se necesite en lugar de añadirlo en el código. En el presente caso de estudio se empleará para añadir archivos que conforman una web como es el caso del html.

Finalmente, encontramos la biblioteca <WiFi.h>, que contiene las funciones necesarias para conectarse a la red WiFi desde el ESP32.

5.1.1 Variables

Hay que definir los pines que se utilizarán para la obtención de información del ESP32. Estos pines del ESP32 están conectados con los sensores ópticos que se han empleado para medir la velocidad del motor. Estos pines han sido escogidos ya que sólo sirven para transmitir información del exterior al interior, y no al contrario.

```
11  //Inicialización de los pines que se tomarán del ESP32 para visualizar la información de los sensores ópticos
12  #define VP_PIN 36
13  #define VN_PIN 39
14  #define D34_PIN 34
```

Código 4. Definición de Pines empleados en el ESP32

Para saber exactamente qué número pertenecía a cada PIN, se ha cortejado con la siguiente tabla:

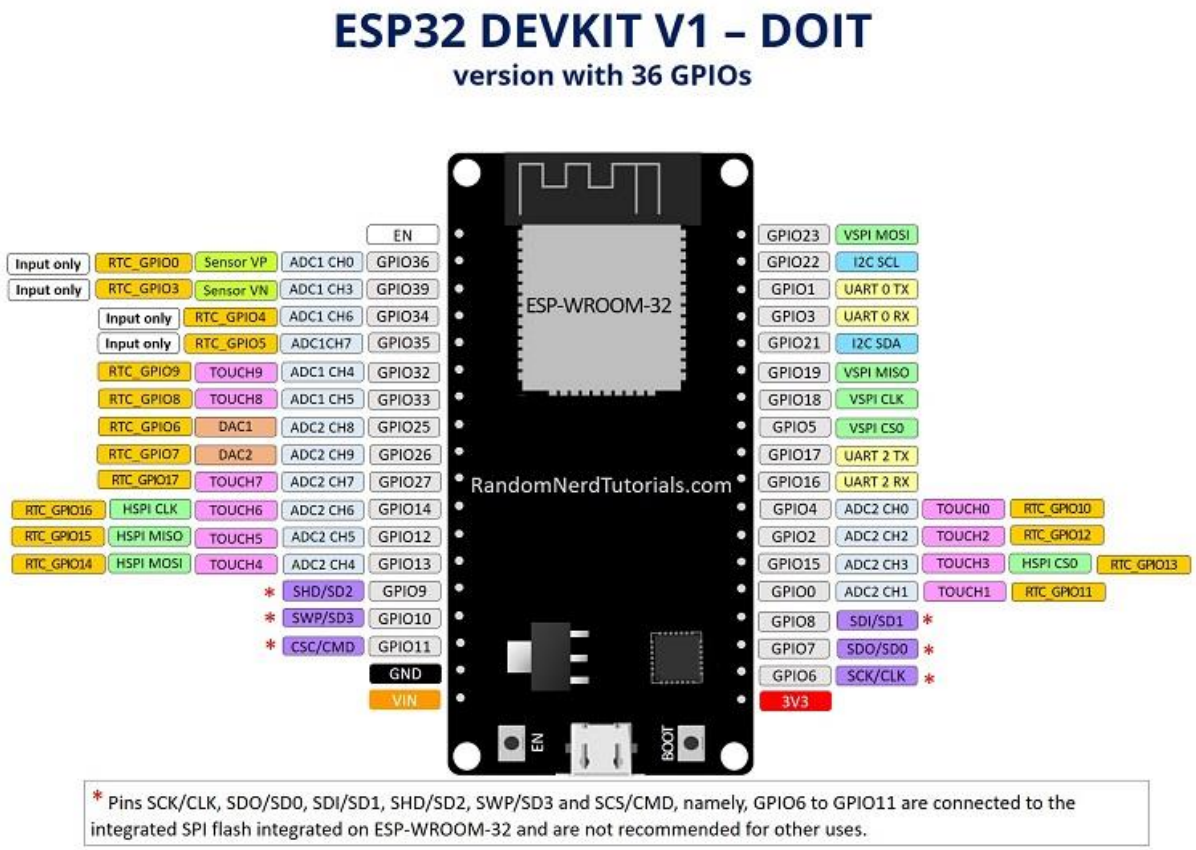


Fig. 26. Tabla de Pinouts [16]

5.2 EL HTML

Es el código que se utiliza para estructurar y construir una página web y sus contenidos.

Antes de empezar con las funciones dentro de lo que vendría a ser el “int main” del código, se empezará a explicar cuál es la estructura que se ha seguido para construir una página web.

Lo primero que se puede observar en el código [Código 5. Archivo html y la sección <cabeza>.Código 6. Sección cuerpo], es que el html se ha creado dentro de una carpeta llamada <datos>. Seguidamente

de cuando se haya creado el archivo, se procederá a declarar el tipo de este archivo, que como bien se ha mencionado con anterioridad es html. El archivo se declara en la línea de código 1.

En la segunda línea de código se le impone que el idioma empleado será el español.

En la tercera línea se empezará con los ajustes iniciales de la página web, es decir, con la sección <cabeza>. Se emplea para incluir información sobre la página que no se muestra directamente al usuario, como los scripts y los estilos CSS (Cascading Style Sheets).

Dentro de la sección cabeza se encuentra:

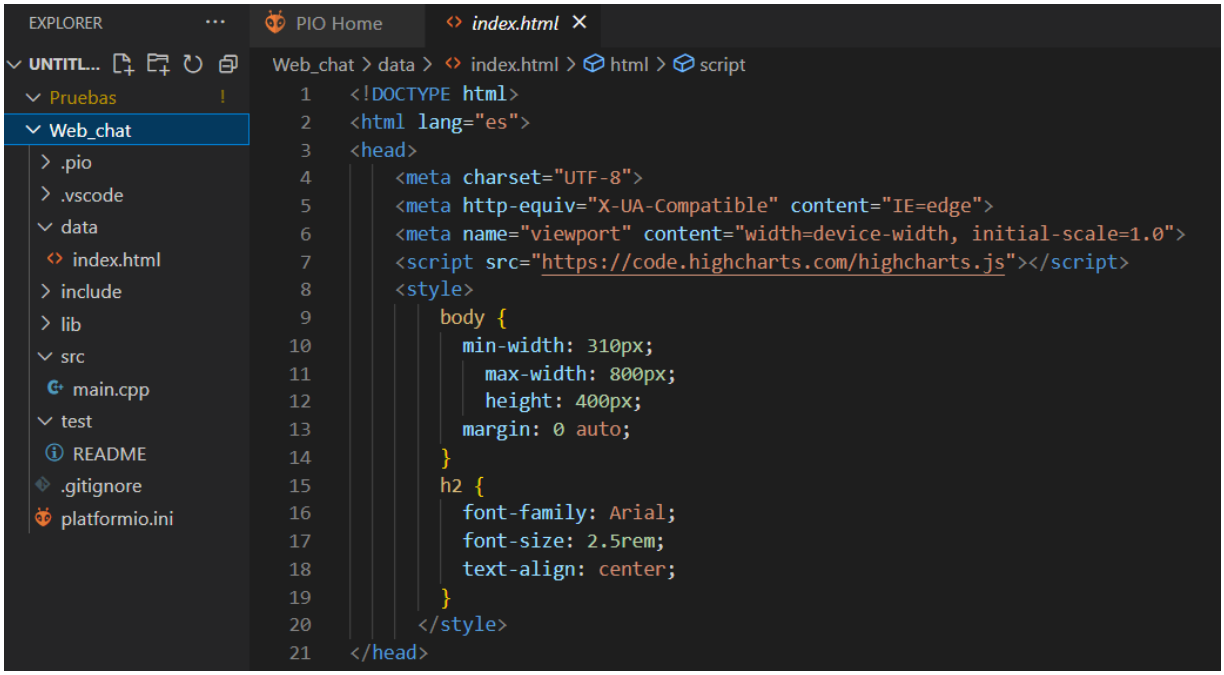
“<meta charset="UTF-8">”: Especifica el conjunto de caracteres que se utilizarán para la página. En este caso, se utiliza UTF-8, que es un conjunto de caracteres que admite una amplia gama de idiomas y caracteres especiales.

“<meta http-equiv="X-UA-Compatible" content="IE=edge">”: Especifica la versión del motor de navegación que se utilizará para mostrar la página web. Se utiliza "IE=edge", que indica que utiliza la versión más reciente de Internet Explorer.

“<meta name="viewport" content="width=device-width, initial-scale=1.0">”: Especifica las dimensiones de la pantalla y la escala inicial de la página. Se emplea "width=device-width" con el motivo de que la página se ajuste automáticamente al ancho de la pantalla del dispositivo, y "initial-scale=1.0" para que la página se muestre inicialmente a escala completa.

“<script src="https://code.highcharts.com/highcharts.js"></script>”: Permite añadir Highcharts, que es una potente librería que permite generar gráficas en Java Script, desde complejas a simples.

<style>: Se define un bloque de estilos CSS para dar formato a la página. Con esto se define el ancho mínimo y máximo de la página, la altura de la página, y el estilo de la fuente para los encabezados. (Por ejemplo, se ve que el estilo de la fuente es Arial).



Código 5. Archivo html y la sección <cabeza>.

Seguidamente, se encuentra la sección <cuerpo>. En ella, se declara el título del gráfico (línea 23). En la línea 25 se creará una sección que llevará el nombre de “chart-velocidad” y que será de la clase

contenedor. Esto último no tiene ningún efecto en la página web, más que ser un agrupador para organizar.

```
22    <body>
23        <h2>ESP velocidad del motor</h2>
24        <div id="chart-velocidad" class="container"></div>
25
26    </body>
```

Código 6. Sección cuerpo dentro del html

Para finalizar, queda la última sección dentro del html, que es el <script>. Se dividirá en dos partes:

Primeramente, se crea un nuevo gráfico utilizando la biblioteca de gráficos HighCharts. Se declara que el gráfico se renderice en el elemento con el identificador "chart-velocidad", comentado anteriormente.

Luego, se procede a crear el título que se incorporará en el eje de las "Y". Por contraparte, en el eje de las "X", no tendrá título, y lo único que aparecerá será la hora en el que se ha tomado el dato. También, se especifica el color con el que se verá la gráfica.

Se especifica que la serie de datos que se mostrará por pantalla inicialmente estará vacía. Aunque esto se debe a que se irán agregando puntos de datos en tiempo real posteriormente, como se verá a continuación.

```
27  <script>
28  var chartT = new Highcharts.Chart({
29    chart:{ renderTo : 'chart-velocidad' },
30    title: { text: 'Velocidad motor asincrono de 2,2kw' },
31    series: [{
32      showInLegend: false,
33      data: []
34    }],
35    plotOptions: {
36      line: { animation: false,
37        dataLabels: { enabled: true }
38      },
39      series: { color: '#059e8a' }
40    },
41    xAxis: { type: 'datetime',
42      dateTimeLabelFormats: { second: '%H:%M:%S' }
43    },
44    yAxis: {
45      title: { text: 'Velocidad (rpm)' }
46      //title: { text: 'Velocidad (rev/min)' }
47    },
48    credits: { enabled: false }
49  });
50
```

Código 7. Sección script dentro del html. Parte 1

En la segunda parte del script se encuentra la función <setInterval>, que se encarga de pedir los datos obtenidos en la función “main”. El gráfico se actualiza cada 1000 milisegundos, agregando los puntos obtenidos a la serie utilizada empleando la función “addPoint”, de HighChart.

```
50
51  setInterval(function ( ) {
52    var xhttp = new XMLHttpRequest();
53    xhttp.onreadystatechange = function() {
54      if (this.readyState == 4 && this.status == 200) {
55        var x = (new Date()).getTime(),
56            y = parseFloat(this.responseText);
57        //console.log(this.responseText);
58        if(chartT.series[0].data.length > 40) {
59          chartT.series[0].addPoint([x, y], true, true, true);
60        } else {
61          chartT.series[0].addPoint([x, y], true, false, true);
62        }
63      }
64    };
65    xhttp.open("GET", "/velocidad", true);
66    xhttp.send();
67  }, 1000) ;
68  </script>
69
70
71  </html>
```

Código 8. Sección script dentro del html. Parte 2

5.3 CÓDIGO PRINCIPAL

Una vez se sabe cómo se generará la página web, se procederá a ver como se obtienen los valores que se muestran por la gráfica y lo necesario para que todo funcione. Esto último, agrupa desde la conexión de la red WiFi, hasta el proceso para obtener los valores dados por los sensores ópticos y transformarlos en la velocidad que se requerirá.

Primeramente, se explicará brevemente las partes que no están incluidas dentro de las funciones principales, el “setup” y el “loop”. Sin embargo, las variables que se declaren fuera de ambas se explicarán una vez se empleen en estas mismas funciones al igual que la función que hace el cambio de unidades de la velocidad.

Para acceder a la red WiFi, se deberá de indicar el nombre de la red WiFi a la que se conecta en la línea 7 y poner la contraseña de la WiFi en la línea 9. Hay que señalar que para que posteriormente la funcione la conexión, la WiFi a la que se debe de conectar debe de ser 2G o 3G y tener acceso a internet. Esto se debe a que con la WiFI 4G y 5G presenta problemas de compatibilidad con la librería utilizada.

```
7   const char* ssid = "nombre de la red WiFi";
8
9   const char* password = "contraseña";
```

Código 9. Declaración de red WiFi

5.3.1 Función “void setup”

Como se puede ver en el *Código 10. Función setup*, la primera acción que hace es declarar que los pines de los que se va a recibir información son solo pines de entrada y no de salida (Estas señales serán binarias: 1 o 0).

Luego se procederá a comprobar que todo funcione, iniciando el chequeo por las bibliotecas SPIFFS. Seguidamente, se encuentra la comprobación de la conexión a la red WiFi, que muestra el estado de la conexión hasta que finalmente se logre (Si en el *Código 9* se hubiera puesto de forma incorrecta la contraseña, o la red WiFi fuera 4G o 5G, el proceso de conexión seguiría en bucle).

Por último, está la función que permite obtener el código para acceder a la página Web y ver los resultados. Estos resultados son enviados desde la línea 82 al html, que posteriormente los traslada a la gráfica.

En el apartado 4.3.2.4 se mostrará la función readvelocity, que es la función que transmite los datos que se incorporarán en la gráfica.

```
57  void setup()
58  {
59    //activación de los pines del ESP32 para recibir información
60    pinMode(VP_PIN,INPUT);
61    pinMode(VN_PIN,INPUT);
62    pinMode(D34_PIN,INPUT);
63
64    //Muestra por pantalla que ha habido un error, en caso de que las Bibliotecas SPIFFS no hayan funcionado
65      if(!SPIFFS.begin()){
66      Serial.println("un error ha ocurrido con los SPIFFS");
67      return;
68    }
69    //Bucle que se repite hasta que se haya conectado a la red WiFi
70    Serial.begin(115200);
71     WiFi.begin(ssid, password);
72    while (WiFi.status() != WL_CONNECTED) {
73      delay(1000);
74      Serial.println("Conectando a la red Wi-Fi..");
75    }
76    //Se obtiene el código para acceder a la página web donde se mostrarán los datos.
77     Serial.println(WiFi.localIP());
78     server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
79      request->send(SPIFFS, "/index.html");
80    });
81    //Obtenemos el valor de la función readvelocity y lo representamos en la gráfica
82    server.on("/velocidad", HTTP_GET, [](AsyncWebServerRequest *request){
83      request->send_P(200, "text/plain", readvelocity().c_str());
84    });
85    // Start server
86    server.begin();
87  }
```

Código 10. Función setup

5.3.2 Función “void loop”

La función void loop, como bien se ha explicado anteriormente en el apartado 3.3.2, es una función que se repite constantemente. Debido a su naturaleza, será la función que se empleará para obtener las señales aportadas por los sensores ópticos. Estas señales serán transformadas, para finalmente obtener la velocidad deseada, que es la que se mostrará en la gráfica de la Web.

Para tener una mayor facilidad de compresión del código que se ha empleado en esta función, se dividirá en dos partes. La primera parte será la traducción de las señales de los sensores ópticos en posiciones y la segunda la transformación de las posiciones obtenidas en la velocidad del motor.

5.3.2.1 Variables

Antes de incidir en el código se mostrarán las variables empleadas para él. Estas variables están distribuidas entre variables globales, declaradas fuera del "void loop", y las locales que están dentro:

Las que están fuera es debido a que deben ser accesibles para lectura o para escritura por funciones externas a la función <loop()>.

Entre ellas, cabe destacar que la que finalmente aparecerá por pantalla en la página web, será la "w_fil", y que esta misma no se empleará en el "loop", sino en otra función que se verá más adelante.

```
16  //Inicializacion de las variables dentro del loop
17  unsigned int dtus=40; //tiempo de muestreo, se elige 40, para que pueda mostrar bastantes puntos por vuelta
18  int cuenta=0;
19  int estado_ant1=0;//estado anterior de la variable estado, que es igual a la información dada por el pin VP_PIN del ESP32
20  int estado_ant2=0;//estado anterior de la variable estado, que es igual a la información dada por el pin VN_PIN del ESP32
21  int estado_ant3=0;//estado anterior de la variable estado, que es igual a la información dada por el pin D34_PIN del ESP32
22  float w;//velocidad positiva
23  float w2;//velocidad negativa
24  float w_fil=0; //velocidad en rpm
25  float alpha=0.2;//factor para hacer un filtro pasabajos
26  int marc=0; //Es un marcador, que en caso de que la velocidad sea positiva es 0 y si es negativa 1
27  int negativo=0;// Indica si la sucesión de velocidades anteriores es negativa o positiva
```

Código 11. Declaración de variables fuera de la función "loop"

Las variables que están declaradas dentro de la función "void loop" son principalmente variables de estado. Se emplean para tener constancia de datos anteriores, y así compararlos con los actuales, obteniendo la posición del disco perforado de acuerdo a los cambios de estado.

```
87  void loop() {
88  //Se declara la variable nextus, que permitirá obtener la velocidad controlando el tiempo que hay entre dos tomas de datos
89    unsigned int nowus, nextus;
90  //Se define que nextus es igual al tiempo en microsegundos más 40 microsegundos
91    nextus = micros()+dtus;
92    int ct =0; //declaración del conteo
93    int p0 = 0; //declaración de la posición anterior
94    int p1; //declaración de la posición actual
95    int cont0=0;//declaración de tres posiciones anteriores a la actual
96    int cont1=0;//declaración de dos posiciones anteriores a la actual
97    int incr1=0;//diferencia entre anterior vs dos posiciones anterior
98    int incr2=0;//diferencia entre dos posiciones anterior vs tres posiciones anterior
99    int cont2=0;//declaración de cuatro posiciones anteriores a la actual
100   int cont3=0;//declaración de cinco posiciones anteriores a la actual
101   int incr3=0;//diferencia entre tres posiciones anterior vs cuatro posiciones anterior
102   int incr4=0;//diferencia entre cautro posiciones anterior vs cinco posiciones anterior
```

Código 12. Declaración de variables dentro de la función "loop"

5.3.2.2 Fundamentos teóricos de la conversión de señales del encoder a velocidad angular

El código que se mostrará a continuación es el utilizado para obtener la velocidad. Sin embargo, antes de incidir en él, se debe de explicar a grandes rasgos qué se ha hecho para pasar de una señal binaria a una velocidad angular.

Primeramente, se necesitan tres señales para poder determinar la velocidad y el sentido. Esto se debe a que es un encoder óptico incremental, funcionamiento comentado en el apartado 1.2.1.

Estas señales provendrán de tres sensores ópticos, uno situado a diferente pista que los demás (sólo dará una señal por vuelta) y otros dos que estarán separados aproximadamente 90º eléctricos. Al estar desfasados ese ángulo, se podrá ver en qué dirección va la onda atendiendo a la secuencia de estados de ambas señales.

Una vez obtenidas las señales se debe de interpretar la información binaria que aportan. Para ello, se empleará una tabla de verdad, que representa los ocho casos que se pueden dar. Los casos vienen determinados a causa de que las señales son cuadradas y desfasadas 90º.

VP	VN	NOR VP	NOR VN	
0	1	0	0	Incremento
0	0	1	0	Incremento
1	0	1	1	Incremento
1	1	0	1	Incremento
0	0	0	1	Decremento
1	0	0	0	Decremento
1	1	1	0	Decremento
0	1	1	1	Decremento

Tabla 3. Tabla de verdad de los sensores

Las variables empleadas son: VP (señal del sensor 1), VN (señal del sensor 3) y NOR VP y NOR VN son las señales anteriores de los mismos sensores. En realidad, al tratarse de 4 señales binarias, las combinaciones teóricas podrían llegar a ser $2^4 = 16$, pero el propio diseño del encoder con su separación de señales en 90º eléctricos hace que cuando en una de las señales se produce un flanco (un cambo de estado binario entre la situación actual y la anterior) en la otra señal el estado no haya cambiado y sólo se deba contemplar el caso en que el estado actual y el previo sean ambos 1 o 0 lógico.

Conociendo la tabla de verdad se consigue saber si el disco perforado tiene sentido positivo o negativo. En caso de que el resultado sea “incremento”, el sentido es positivo, y si es “decremento” será negativo. Las nomenclaturas empleadas para describir el sentido del disco han sido seleccionadas debido a que posteriormente, para poder calcular la posición se hará un conteo. En caso de que el giro sea en sentido positivo, la cuenta aumentará en una unidad y en sentido contrario si el giro es en sentido negativo.

Esta cuenta llega hasta 512, ya que el disco perforado cuenta con 128 agujeros, y se dispone de 4 estados diferentes de las salidas de los sensores ópticos para el movimiento del disco correspondiente a un paso completo, lo que en total resulta en 512 estados. Sin embargo, una vez que la cuenta llegue a ese tope, o por el contrario al 0 en caso de ser giro negativo, se deberá de seguir. Esto se consigue de forma automática reiniciado la cuenta al recibir la señal de paso por cero.

Conociendo la idea principal de lo que se quiere programar, se procede a explicar el programa empleado para conseguir tal objetivo.

5.3.2.3 Adquisición de posición

Primeramente, se creará un bucle (que se ejecuta a paso temporal fijo de valor inferior a la mitad de la mínima separación temporal entre eventos de flanco en cualquiera de las señales del encoder) donde se pondrá todo el código de evaluación de la posición. Se inicializarán las variables estado con los valores

aportados por los sensores ópticos. Y finalizada la primera ejecución del ciclo, se declararán los estados anteriores.

Hay cinco casos que se puedan dar, aunque en una misma secuencia sólo se pueden dar tres:

-El primer caso se da cuando ocurren una de estas dos condiciones: La primera es que el valor aportado por el VP_PIN sea 0 y el estado anterior del VN_PIN sea 0. La segunda condición es que el valor dado por el VP_PIN sea 1 al mismo tiempo que el valor del estado anterior del VN_PIN sea 1. Si se cumplen cualquiera de las dos condiciones, se sabrá que el sentido del giro es positivo, por lo que la cuenta aumentará en 1.

-El segundo se dará también cuando se cumplan dos condiciones: La primera es que el valor actual del VN_PIN sea distinto de 0 al mismo tiempo que el valor anterior del VP_PIN sea también distinto de cero. La segunda es que el estado VN_PIN sea 0 al igual que el estado anterior del VP_PIN. En caso del cumplimiento de este caso, la cuenta decrementará en 1.

-El tercer caso es que la cuenta sea superior a 511, por lo que se deberá de reiniciar la cuenta, poniéndola a 0, para que en caso de que se siga sumando, la cuenta sea correcta. En este caso, incluye un marcador, que se inicializa a 0, indicando que la serie sigue un transcurso positivo, y por ende, la velocidad será positiva.

-El cuarto caso sería lo mismo que el anterior pero que en caso de que la cuenta sea inferior a 0, la cuenta se reiniciará desde el valor de 511. En este caso, incluye un marcador, que se inicializa a 1, indicando que la serie sigue un transcurso negativo, y por ende, la velocidad será negativa.

-El quinto caso y último es el paso por 0. Este caso viene marcado por la señal asociada al "estado 3", proveniente del pin D34_PIN. Una vez que esta señal sea 1 provocará que la cuenta se reinicie asignándosele el valor de 0.

```
106  while(true){
107    //Se asigna a las variables estados el valor que se da por el ESP32 en los pines correspondienres (valores: 0 y 1)
108    int estado1 = digitalRead(VP_PIN);
109    int estado2 = digitalRead(VN_PIN);
110    int estado3 = digitalRead(D34_PIN);
111
112    if (((estado1==0) && (estado_ant2==0))||((estado1!=0) && (estado_ant2!=0)))
113    {
114    cuenta = cuenta+1; //Incremento
115    }
116    if (((estado2!=0) && (estado_ant1!=0))||((estado2==0) && (estado_ant1==0)))
117    {
118    cuenta = cuenta-1; //Decremento
119    }
120
121    //En caso de que la cuenta supere el 511, se reinicia la cuenta, por lo que cuenta=0
122    if(cuenta>511)
123    {
124      cuenta=0;
125      negativo=0;
126    /*Además se incluye la variable negativo, ya que en caso de que la cuenta esté aumentando
127    signifacará que el movimiento del motor será positivo.*/
128    }
129
130    //En caso de que la cuenta sea menor que 0, si disminuyera más, la siguiente posición que encontrariamos es 511, no -1
131    if(cuenta<0)
132    {
133      cuenta=511;
134      negativo=1;
135    /*Además se incluye la variable negativo, ya que en caso de que la cuenta esté disminuyendo
136    signifacará que el movimiento del motor será negativo.*/
137    }
```

```
138
139     //Cuando se pase por la ranura de referencia, la cuenta se reinicia
140     if (estado3!=0)
141     {
142       cuenta=0;
143     }
144
145     //Se declara el estado anterior.
146     estado_ant1=estado1;
147     estado_ant2=estado2;
148     estado_ant3=estado3;
```

Código 13. Declaración de estados y condiciones de conteo

5.3.2.4 Obtención de la velocidad angular

Tras obtener el valor de la cuenta, ahora se requerirá obtener la velocidad angular a partir de este dato. Para ello se necesitarán dos datos, espacio y tiempo. El espacio se obtendrá con la diferencia entre la cuenta actual menos la cuenta anterior, y el tiempo se obtendrá creando una parada artificial a partir de un bucle. El tiempo estará marcado por la variable "nextus", que crea una espera en el bucle inicial hasta que esta misma variable sea menor que el tiempo que está transcurriendo en vivo. Es decir, se fuerza a que el bucle se ejecute a paso temporal fijo. Este valor se ha establecido en un valor inferior a la mitad de la mínima separación temporal entre eventos de flanco en cualquiera de las señales del encoder. Esto permite obtener los valores que se desean en un tiempo determinado mediante una cuenta de ejecuciones del bulce, dando lugar a la velocidad por relación entre diferencia de posición y diferencia de tiempo.

Lo primero que se tiene es un condicional, cuya función es que en caso de que la "ct" sea igual a 0, permitirá entrar en el proceso de cálculo de la velocidad, tanto positiva "w", como negativa "w2".

Dentro de este condicional habrá cuatro posibles casos. Para poder acceder dentro de cualquiera de estos casos se deberá cumplir unas condiciones. La primera acción que se realizará una vez se entre dentro del condicinal será definir la variable "p1" como el valor de "cuenta" y "p0" como el valor anterior de la cuenta (También se definirán todos los valores anteriores a esas cuentas).

Los dos casos que proporcionarán la velocidad positiva:

> El primero es el caso más habitual, que es en el que la posición actual es mayor que la posición anterior ("p1>=p0"), y en los casos anteriores la diferencia también es positiva. Con ello, la velocidad se podrá calcular como la diferencia entre la posición actual menos la posición anterior.
>
> El segundo caso es poco probable, pero no imposible. Este caso se da cuando la velocidad es positiva pero la posición anterior es mayor que la posición actual. Anteriormente se ha comentado que una vez se llega a los 512, la cuenta se reinicia; en ese momento es cuando se puede dar el caso. Entonces para su cálculo, se deberá de sumar 512 a la velocidad obtenida por la diferencia para obtener el valor correcto.

Por otra parte, los últimos dos casos, que proporcionarán la velocidad negativa.

> El primero, siendo también el más habitual, se da cuando la posición actual es menor que la posición anterior. Esto da paso al cálculo de la velocidad, que es la diferencia entre ambas, "p4-p3", obteniéndose así la velocidad negativa.

El segundo caso se produce cuando la posición actual es mayor que la posición anterior, aunque la velocidad sea negativa. Esto es a causa de que la cuenta ha llegado a un número menor que 0 y se ha tenido que reiniciar, haciendo que la cuenta vuela a ser 511. La velocidad se obtiene igual que antes, pero restando 512 al total que se ha obtenido.

Sin embargo, se puede apreciar un problema, y es que hay que saber distinguir en qué momento se está dando un giro positivo o negativo. Esto es a causa de que la segunda condición de la velocidad negativa es igual a la primera condición de la velocidad positiva, es decir, que la cuenta actual sea mayor que la anterior. Por ello, se ha desarrollado un mecanismo para evitar que ocurra, cuya solución ha sido, registrar cinco casos anteriores. Con ello se puede ver la tendencia que sigue el movimiento, ya que, si se ha dado que en cuatro veces la diferencia entre la posición inicial y final es positiva, quiere decir que estamos en el primer caso de la velocidad positiva y no en el segundo caso de la velocidad negativa. Esto sólo sirve para saber si la tendencia es positiva o negativa.

La solución aportada sólo funcionará en velocidades bajas, por lo que, para implementar el código, se optará por emplear la variable "negativo", anteriormente mencionada. Esto permite saber si las velocidades registradas en casos anteriores son positivas o negativas. La lógica empleada para este implemento es que, si la velocidad es positiva, más pronto o más tarde, la cuenta superará el valor de 511. Al contrario, se da el mismo suceso, si la cuenta disminuye de 0 se sabrá que la velocidad es negativa. Aprovechando que en el código ya se habían definido ambos casos, se declara la variable "negativo", que llevará el registro de si la velocidad es positiva o negativa.

```
150     if(ct==0){
151       p1 = cuenta;
152       /*En caso de que el cuenteo sea 0, que sería el caso inicial,
153        se declara que la posición 1 es la cuenta que se lleva.*/
154
155       if((p1>=p0)&& (incr1>0) && (incr2>0) && (incr3>0) && (incr4>0)) {
156         w = (p1-p0);
157         /*Si la posición actual es mayor que la anterior,
158         se declara que la velocidad es la diferencia de estas posiciones,
159         en un tiempo determinado, es decir, el tiempo nextus*/
160
161       }
162       if((p1>=p0)&& (incr1<0) && (incr2<0)&& (incr3<0) && (incr4<0)){
163         w2 = p1-511-p0;
164         /*En caso de que sea menor la posición actual, que la anterior,
165         sólo se tendrá que restar 512 a la actual, haciendo que
166         la dieferencia sea positiva*/
167       }
```

Código 14. Cálculo de velocidad positiva

```
168       if((p1<p0) && (incr1>0) && (incr2>0)&& (incr3>0) && (incr4>0)){
169         w = 511+p1-p0;
170         /*En caso de que sea menor la posición actual, que la anterior,
171         sólo se tendrá que sumar 512 a la actual, haciendo que
172         la diefrencia sea positiva*/
173       }
174       if((p1<=p0)&& (incr1<0) && (incr2<0)&& (incr3<0) && (incr4<0)){
175         w2 = (p1-p0);
176         /*Si la posición actual es mayor que la anterior,
177         se declara que la velocidad es la diferencia de estas posiciones,
178         en un tiempo determinado, es decir, el tiempo nextus*/
179       }
```

Código 15. Cálculo de velocidad negativa

```
180       ct++;
181       p0=p1;
182       incr1=p0-cont1;
183       cont1=p0;
184       incr2=cont1-cont0;
185       cont0=cont1;
186       incr3=cont0-cont2;
187       cont2=cont0;
188       incr4=cont2-cont3;
189       cont3=cont2;
190
191       if(negativo!=0)
192       {
193         marc=1;
194       }
195       if(negativo==0)
196       {
197         marc=0;
198       }
199     }else ct++;
200     if(ct==500)ct=0;
```

Código 16. Solución para distinguir velocidad positiva de negativa

5.3.2.4 Función de conversión de unidades y transferencia de valores a la página web

Para realizar el cambio de unidades y transmitir la velocidad a la página web se ha diseñado la siguiente función. Que obtiene las velocidades proporcionadas por la función “loop”, y luego las transforma con el fin de obtener la velocidad en revoluciones por minuto (rpm). Finalmente, este valor es enviado a la función “setup”, que finalmente se representará por pantalla.

Se emplea un filtrado para reducir la variabilidad de la velocidad que proporcionará la gráfica. Con este filtrado se consigue eliminar los valores atípicos respecto al valor más concurrente.

```
29  AsyncWebServer server(80);
30  String readvelocity() { //envía el valor de la velocidad
31    if(marc==0)
32    {
33        w=w/10*60;
34        w_fil=alpha*(w)+(1-alpha)*w_fil;
35    }else
36    {
37        w2=w2/10*60;
38        w_fil=alpha*(w2)+(1-alpha)*w_fil;
39    }
40      Serial.println(w_fil);
41      return String(w_fil);
42  }
```

Código 17. Conversor de velocidades

5.4 RESULTADOS

En la [Fig. 27. Gráfica aportada por el programa] se ve la velocidad obtenida por el encoder incremental óptico. Como se puede observar, el motor gira 60 revoluciones por minuto y gira en sentido horario, es decir, la velocidad es positiva. Se ha empleado filtrado para la obtención de la velocidad, obteniéndose valores dentro de un rango aceptable.

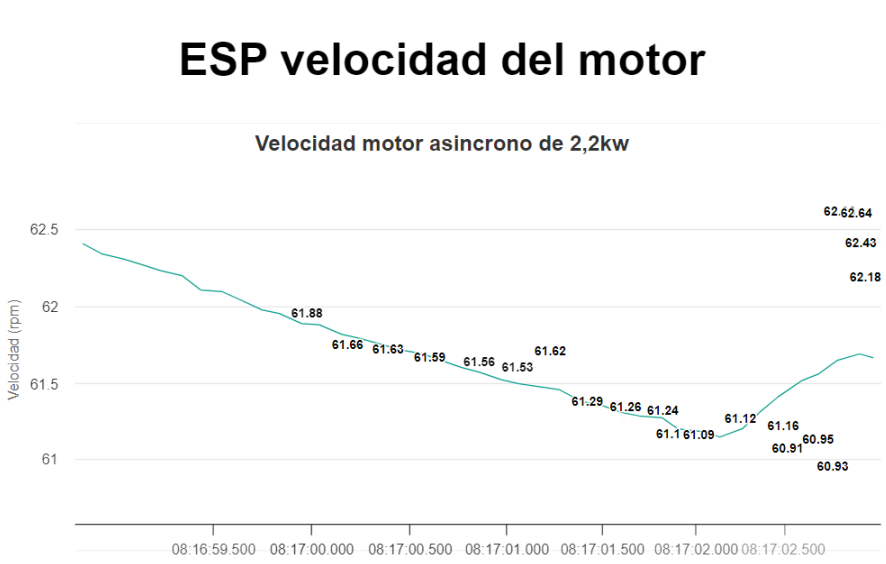


Fig. 27. Gráfica aportada por el programa

CAPÍTULO 6: AUTOMATION STUDIO

En el capítulo a tratar, se verán los pasos a seguir para poder arrancar el motor con el que se ha hecho el estudio pertinente.

En el capítulo a tratar, se verán los pasos a seguir para arrancar el motor con el que se ha hecho el estudio pertinente y se analizará la herramienta de programación empleada para este propósito

6.1 TRABAJAR CON AUTOMATION STUDIO

Automation Studio es una herramienta de programación utilizada para diseñar y simular sistemas de automatización industrial. Es un software completo que integra múltiples funcionalidades, como la creación de diagramas eléctricos, la programación de PLC (Controladores Lógicos Programables), la simulación de circuitos hidráulicos y neumáticos, la creación de interfaces de operador y la generación de documentación técnica. [18]

La herramienta proporciona una interfaz gráfica intuitiva que facilita el diseño y la configuración de sistemas de automatización, lo que la hace popular en la industria de la automatización y control de procesos.

6.2 PROCESO

En el proyecto a realizar, se emplearán las funciones más básicas del programa. No se requerirá programar nada y sólo se deberá de poner en marcha el motor a una velocidad constante, ya sea positiva o negativa.

Los pasos por seguir para hacer funcionar al motor son:

-Primero se encenderá el interruptor diferencial permitiendo el paso de la corriente al autómata.

-Se abrirá la aplicación, y se deberá de esperar hasta que se muestre por pantalla que el autómata está en modo “RUN”. Aparecerá en el borde derecho inferior de la pantalla.

Fig. 28. Programa Automation Studio: Activación del programa

-Una vez operativo el sistema, se abrirá la pestaña “monitor” para sincronizar el autómata con el motor. La sincronización, se realizará declarando el estado del motor en modo “TRUE” y forzar su arrancado.

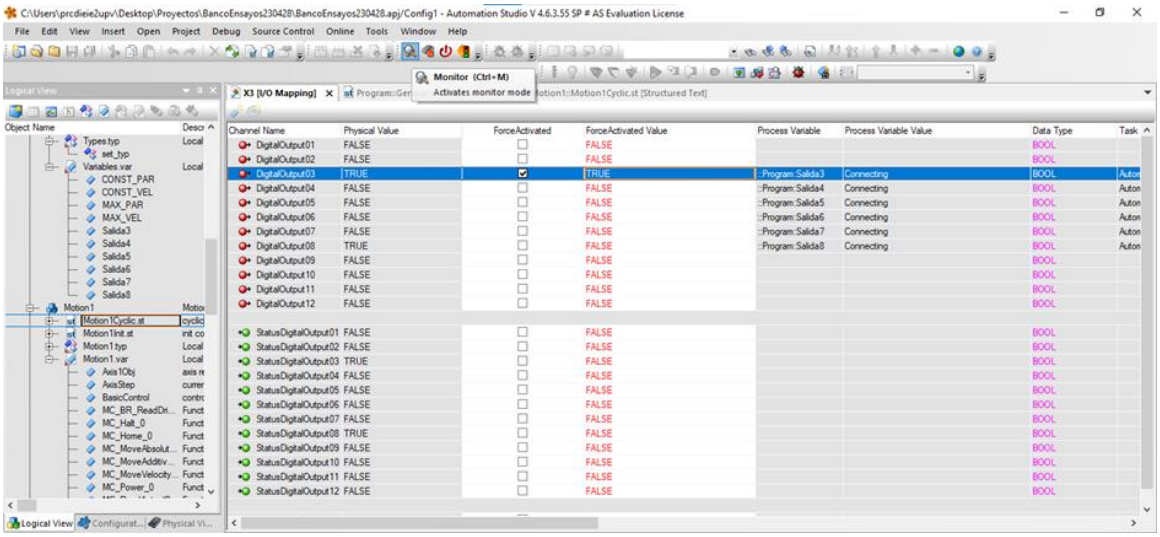


Fig. 29. Programa Automation Studio: Sincronización con motor

-Se accederá a la pestaña de “physical views” donde se podrá seleccionar el motor vinculado y así realizar la acción “test”. A continuación, aparecerá una ventana en forma de alerta; se deberá de seleccionar “Exclusive mode” para empezar con el ajuste de velocidades.

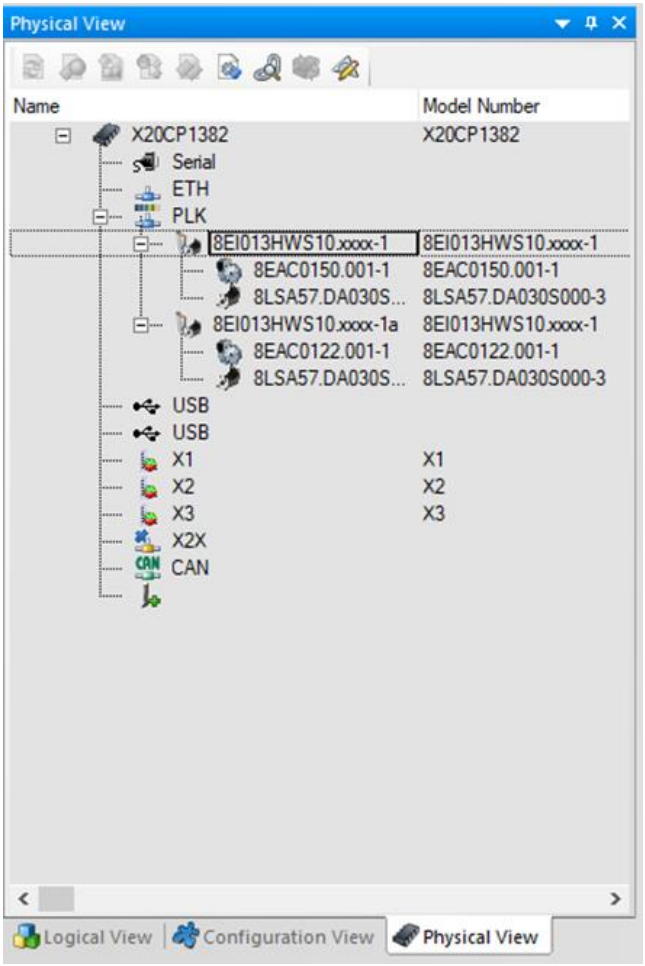


Fig. 30. Programa Automation Studio: Testeo

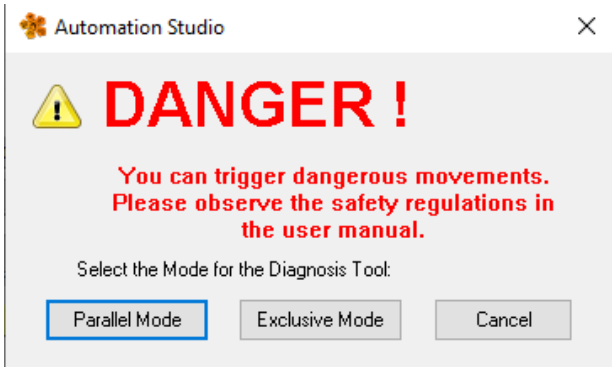


Fig. 31. Programa Automation Studio: Ventana de peligro

-Finalmente, al haber ejecutado la acción test, aparecerá una ventana con la que se podrá modificar la velocidad. Para evitar un mal funcionamiento del programa hay un semáforo en la ventana que muestra si hay un error o no. El error aparece de color rojo, y el buen funcionamiento en color verde. Una vez que el semáforo esté en verde se encenderá el motor y se procederá a la modificación de su velocidad.

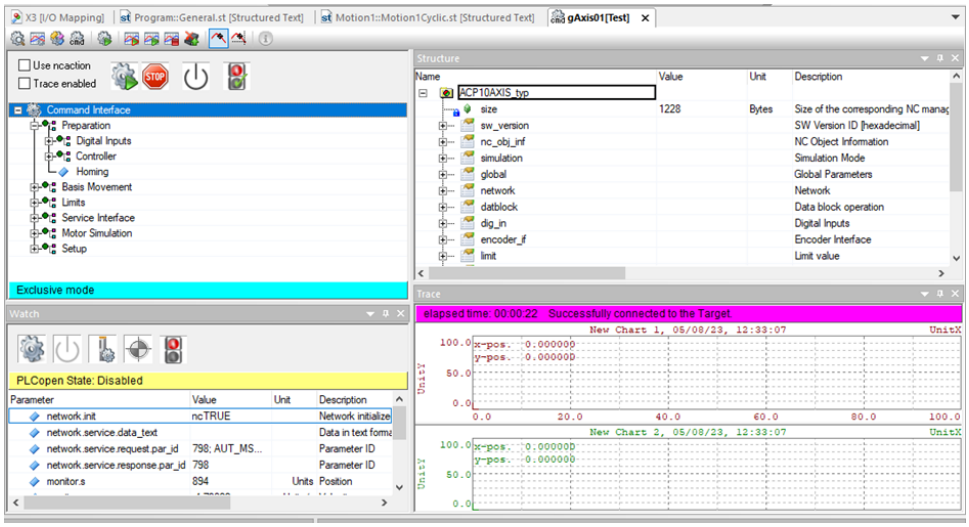


Fig. 32. Programa Automation Studio: Ventana de test

-En la ventana a la que se accede se encuentra la función “controller” que permitirá encender el motor con el botón “Switch on”.

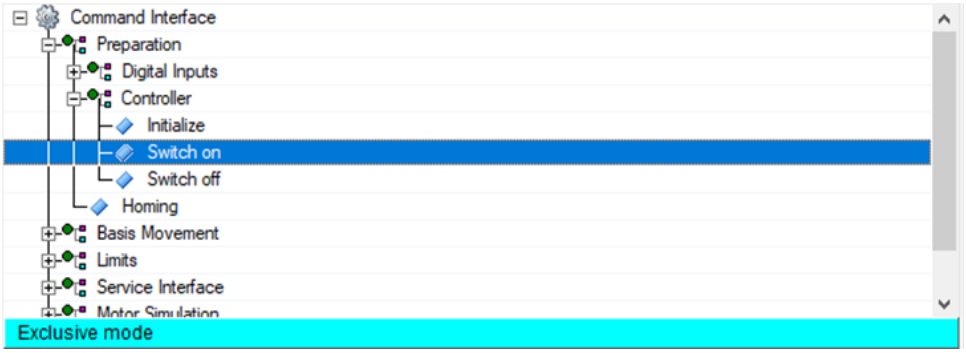


Fig. 33. Programa Automation Studio: Arranque del motor

-Se activará la función “Homing”. Esta función se refiere a la operación de mover un actuador o un eje a su posición de inicio designada. Además, puede ser utilizada para comprobar que los sensores de

posición y los actuadores está funcionando correctamente y que el sistema es capaz de mover los ejes y los actuadores a sus posiciones de inicio y finalización correctamente.

-Para modificar la velocidad del motor se accederá a “Basic Movements”. Para hacer que el motor se mueva al sentido de las agujas del reloj, se seleccionará “Positive”, y si se quiere que el movimiento sea a la inversa, se seleccionará “Negative”. A la izquierda del monitor se encontrará una ventana donde se puede modificar a cuánta velocidad irá el motor. La velocidad en el programa se declara en miles de revoluciones por segundo. En caso de que se desee parar el ensayo, se seleccionará la función “STOP”, que detendrá de inmediato el movimiento del motor.

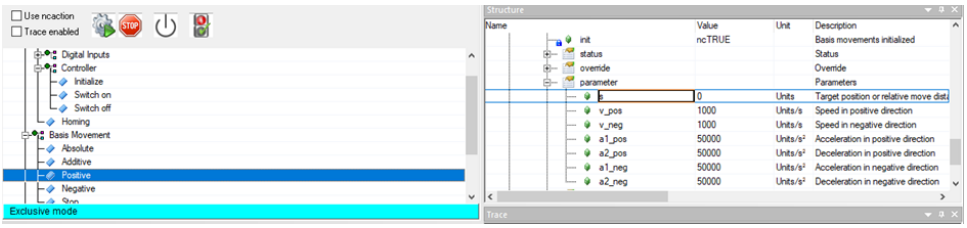


Fig. 34. Programa Automation Studio: Modifiación de parámetros del motor

-Una vez terminado el ensayo, se apagará el motor, con “Switch off” y se cerrará la ventada de “test” para poder cerrar el programa.

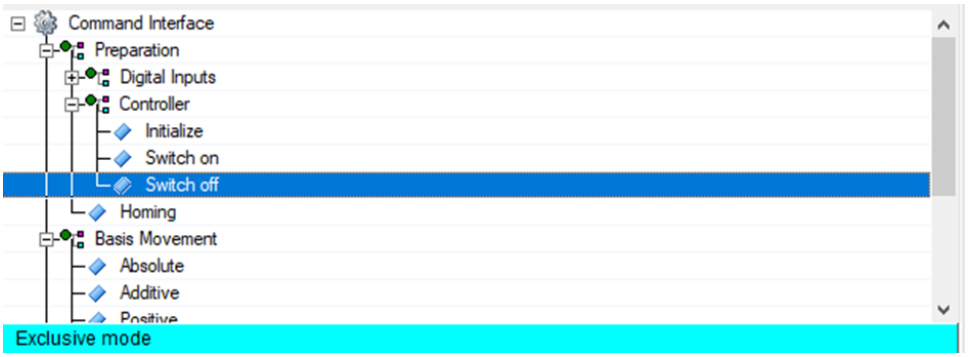


Fig. 35. Programa Automation Studio: Apagar motor

CAPÍTULO 7: RESULTADOS

En este capítulo, se procederá a mostrar el montaje final tras haber completado todos los pasos mencionados con anterioridad.

Encoder óptico acoplado al motor de ensayos:

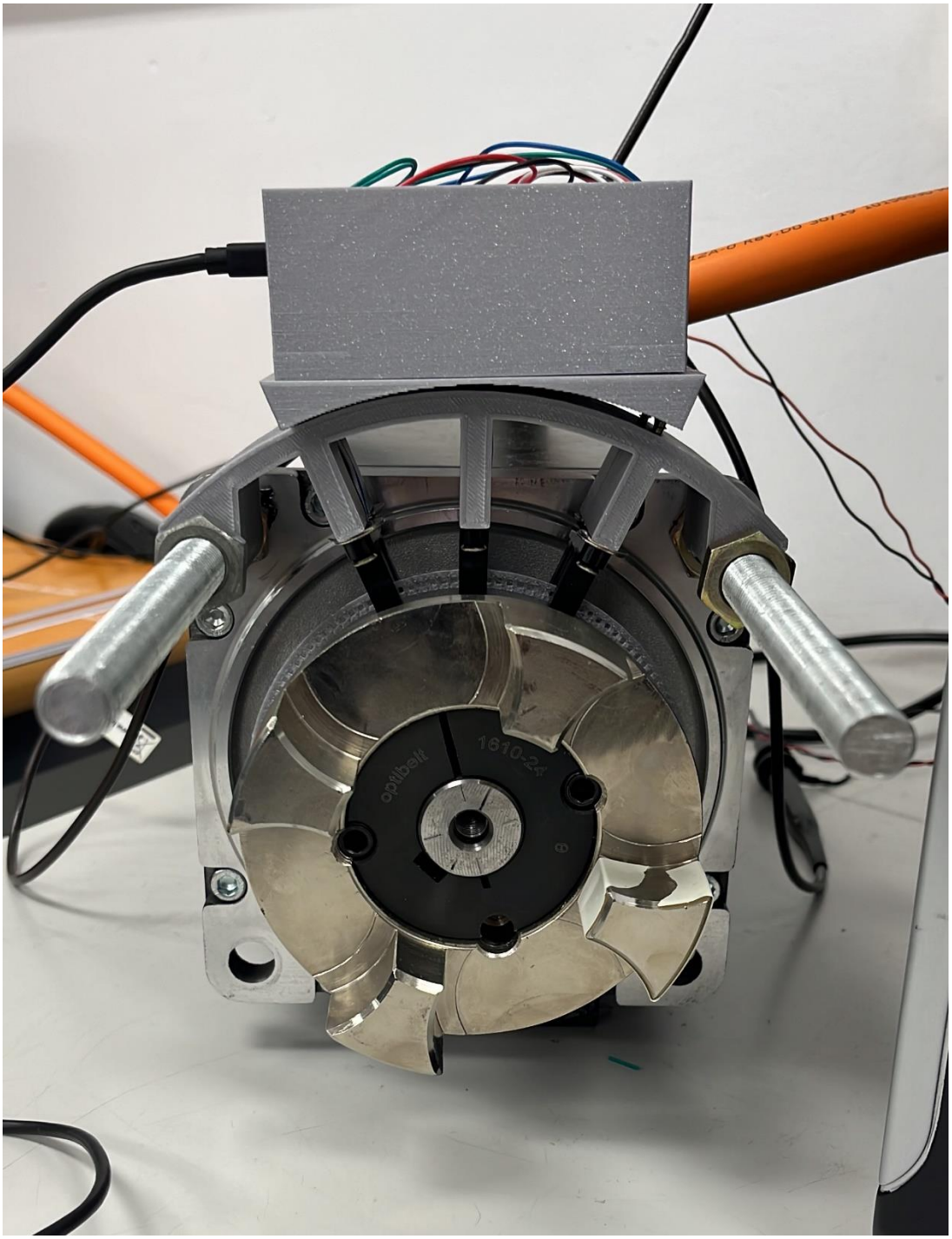


Fig. 36. Encoder óptico acoplado al motor

ESP32 integrado en la placa electrónica junto a todos los demás componentes:

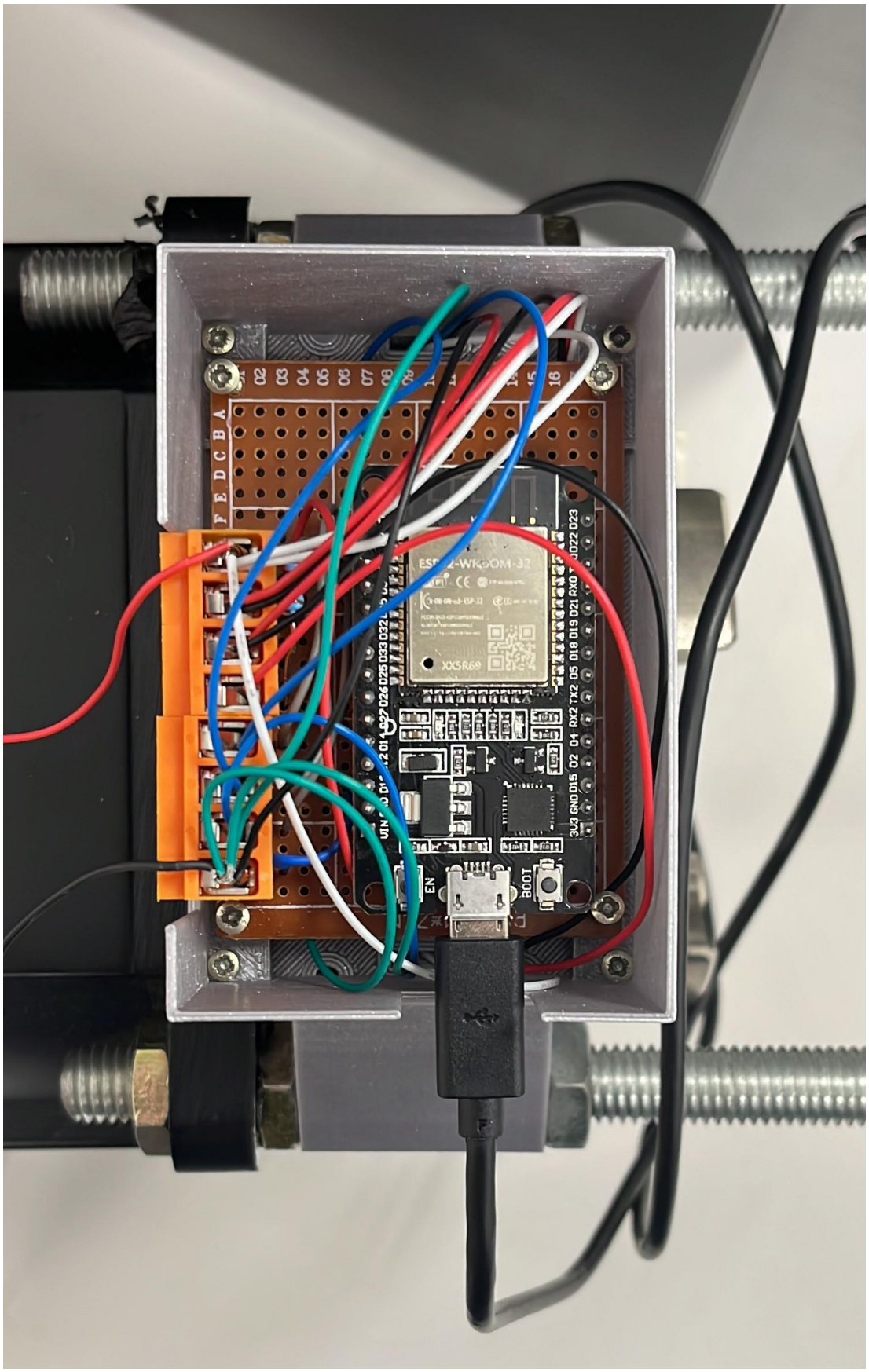


Fig. 37. Montaje placa electrónica

Gráfica obtenida tras el programa desarrollado para el microprocesador ESP32:

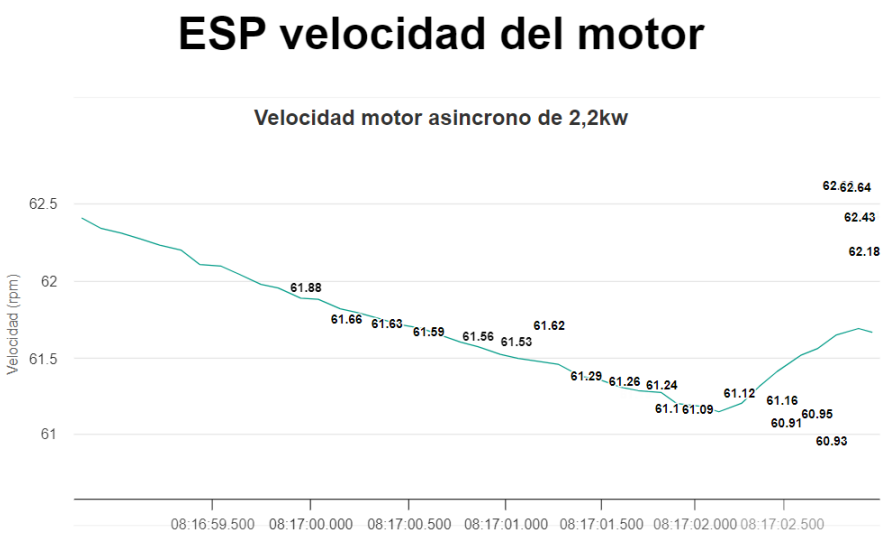


Fig. 38. Gráfica de velocidades

CONCLUSIONES

En resumen, el presente proyecto se ha podido dividir en dos grandes bloques: la creación del encoder óptico incremental y el desarrollo del programa empleado para crear la página web donde se verán los resultados.

Primeramente, se ha creado un encoder óptico incremental, capaz de medir la velocidad y posición de un motor eléctrico asíncrono compuesto por imanes permanentes. El primer paso para la creación de este encoder, ha sido la elección del encoder que más se ajusta a las características impuestas. Posteriormente se han desarrollado los planos que se emplearán en su fabricación en una impresora en 3D.

En este proceso cabe destacar la implicación del programa Inventor para la obtención de estos planos. Además, para el montaje que se ha hecho a posteriori, integrando el microprocesador ESP32 con el encoder, se han necesitado conocimientos en electrónica aplicados al cableado de este. Cabe destacar, que para la obtención del material empleado en el fabricación del encoder, han sido necesarios conocimientos físicos para obtener sus características y hacer las comprobaciones pertinentes respecto a la resistencia de este.

En la segunda parte, se ha analizado el microprocesador ESP32 que ha permitido la creación de la página web y la transmisión de datos entre el encoder y la página. Explicadas las características más notorias del ESP32, se ha realizado el código empleado para el desarrollo de la página web.

Para la realización del código empleado, se ha tenido que profundizar en conocimientos de programación en la plataforma Visual Studio Code (equivalentes a los conocimientos de programación en los que se profundizaría en el entorno de desarrollo Arduino IDE). El código utilizado traduce la información binaria en velocidades, al mismo tiempo que crea una página web asíncrona donde representa los datos en una gráfica.

También se ha explicado brevemente el proceso que hay que seguir para trabajar en el banco de ensayos y poner a prueba lo obtenido en el trabajo.

Por último, el desarrollo del presente Trabajo Final de Grado ha supuesto un trabajo multidisciplinar que ha abarcado todos los ámbitos involucrados en el ser un ingeniero industrial. Se ha puesto en práctica todos los conocimientos adquiridos en la carrera, como el manejo de programas específicos como Inventor o el pensamiento crítico basado en conocimientos de electrónica, eléctrica, física, automática e informática para obtener la mejor decisión.

BIBLIOGRAFÍA

[1] ONU, *Objetivos y metas de desarrollo sostenible,* 2015. [En línea]. Página web: www.un.org. Disponible en:

https://www.un.org/sustainabledevelopment/es/education/

[2] Dynapar, *Magnetic encoders, Dynapar,* 2022. Disponible en:

https://www.dynapar.com/Technology/Encoder_Basics/Magnetic_Encoder/

[3] EcuRed contributors, *Efecto Hall,* EcuRed, 2019. [En línea]. Disponible en:

https://www.ecured.cu/index.php?title=Efecto_Hall&oldid=3491196

[4] Editores de TodoElectronica.com, *Electronicasi.* [En línea]. Página web: www.electronicasi.com. Disponible en:

https://www.electronicasi.com/ensenanzas/electronica-elemental/electronica-basica/corriente-alterna-ac/

[5] Editores de Euroencoders, *Euroencoders.* [En línea]. Página web: www.euronencoder.com. Disponible en:

https://euroencoder.com/encoders-magneticos/.

[6] ACP, *Cómo funciona un sensor magnético, ACP,* 2021. [En línea]. Disponible en:

https://acpautomatismos.com/como-funciona-un-sensor-magnetico/

[7] R. T. A. B. E. R. N. E. R. ROSALENY, *Encoders Ópticos,* [En línea]. Disponible en:

https://infoplc.net/files/documentacion/instrumentacion_deteccion/infoPLC_net_ENCODERS_OPTICOS.pdf

[8] K. Craig, *Actuators & Sensors in Mehatronics: Optical Encoders,* ME. [En línea]. Disponible en:

http://engineering.nyu.edu/mechatronics/Control_Lab/Criag/Craig_RPI/SenActinMecha/S&A_Optical_Encoders.pdf

[9] R. S. de Editores, «OPB916».[En línea]. Página web: es.rs-online.com. Disponible en:

OPB916BZ

[10] E. de Prusa Slicer, *Prusa Slicer,* 2023. [En línea]. Página web: www.prusa3d.com. Disponible en:

https://www.prusa3d.com/page/prusaslicer_424/

[11] Susana, *3Dnative,* 2021. [En línea]. Página web: www.3dnatives.com. Disponible en:

https://www.3dnatives.com/es/plasticos-impresion-3d-22072015/#

[12] Editores de NovaTecnos, *NovaTecnos,* 2019. [En línea]. Página web: th.bing.com. Disponible en:

https://th.bing.com/th/id/OIP.vVB7YykTEWTSHhOct6-HlAHaFj?pid=ImgDet&rs=1

[13] Editores de TublogEn3D, *TublogEn3D,* 2017. [En línea]. Página web: tublogen3d.com. Disponible en:

https://tublogen3d.com/filamentos/guia-filamentos-impresion-3d/

[14] A. Prado, *Scribd*. [En línea]. Página web: www.scribd.com. Disponible en:

https://www.scribd.com/document/461991359/Ficha-tecnica-PLA#

[15] P. Bertoleti, Proyectos com ESP32 y LoRa, Editora NCB, 2019. [En línea]. Disponible en:

Proyectos com ESP32 y LoRa - Pedro Bertoleti - Google Libros

[16] Editores de Todo Sobre Circuitos, *TODO SOBRE CIRCUITOS*, 2020. [En línea]. Página web: www.circuitos-electricos.com. Disponible en:

https://www.circuitos-electricos.com/esp32-especificaciones-y-disenos/

[17] R. Santos, *RANDOM NERDS TUTORIALS*, 2009. [En línea]. Página web: randomnerdtutorials.com. Disponible en:

https://randomnerdtutorials.com/esp32-esp8266-plot-chart-web-server/

[18] Edición B&R, *Trabajar con Automation Studio*, 2018.

[19] Editores de Mokka-sensors, *Mokka-Sensors*. [En línea]. Página web: mokka-sensors.com. Disponible en:

https://mokka-sensors.com.br/2020/03/27/encoders-incrementais-mokka-sensors/.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

DESARROLLO DE UN SISTEMA DE REALIMENTACIÓN DE POSICIÓN Y VELOCIDAD PARA BANCO DE ENSAYOS DE MOTORES SÍNCRONOS CON IMANES PERMANENTES

PRESUPUESTO

Trabajo Final de Grado

ARNAU VICENT CRESPO

Junio, 2023

Índice de presupuesto

CAPÍTULO 1: INTRODUCCIÓN

Se procederá a obtener el gasto económico que se ha necesitado para la elaboración de este proyecto.

Se dividirá en cuatro unidades de obra de las que se realizará un estudio. Se obtendrá, el cuadro de precios unitarios, el cuadro de precios descompuestos y el presupuesto de inversión que costará realizar esta actividad.

CAPÍTULO 2: ELABORACIÓN DE LAS UNIDADES DE OBRA

1. Circuito eléctrico (Ud01)

La primera unidad de obra es el circuito eléctrico. Estará compuesta por materiales necesarios para la construcción de este, además de la mano de obra necesaria para llevarlo a cabo.

Materiales:

CONCEPTO	UNIDAD	CANTIDAD	PRECIO (€)
DOIT ESP32 DEVKIT	u	1	6,9
PLACA DE CIRCUITO IMPRESO	u	1	14,12
CABEZAL DE PINES DE 40 PINES	u	1	0,25
BORNES DE CONEXIÓN DE TORNILLO	u	2	0,208
ESTAÑO PARA SOLDADURA	g	50	0,0496
SOLDADOR ELÉCTRICO	u	1	18,69
CARGADOR USB DE 5V	u	1	4,5
CABLE USB PARA ANDROID	u	1	2,29

Mano de obra:

CONCEPTO	UNIDAD	CANTIDAD	PRECIO (€)
IGENIERO JUNIOR	h	20	35
TECNICO DE LABORATORIO	h	2	20

2. Programación (Ud02)

La segunda unidad de obra es la programación del ESP32. Esta unidad de obra está compuesta por el programa con el que se ha desarrollo el código empleado para obtener la página web y el programa que permite el funcionamiento del banco de ensayos. A esta unidad también se incluye el tiempo dedicado para el desarrollo y confección del código.

Materiales:

CONCEPTO	UNIDAD	CANTIDAD	PRECIO(€)
VISUAL STUDIO CODE	u	1	0
AUTOMATION STUDIO	u	1	0
COMPUTADOR PERSONAL	mes	1	15

Los programas se han adquirido de forma gratuita ya que se han empleado las licencias universitarias proporcionadas por la Universidad Politécnica de Valencia.

El microprocesador ESP32 podría haberse contado en esta unidad de obra. Sin embargo, ha sido incluido en la primera unidad de obra.

El precio con del computador personal proviene del precio total que cuesta el computador, partido la cantidad años de vida útil que tiene. A esto se le deberá de dividir entre 12, para obtener el precio que cuesta por mes el computador. En este caso, el computador personal cuesta 900€ y tiene 5 años de vida útil.

Mano de obra:

CONCEPTO	UNIDAD	CANTIDAD	PRECIO (€)
IGENIERO JUNIOR	h	80	35
SUPERVISOR DEL PROYECTO	h	15	65

3. Encoder óptico (Ud03)

La tercera unidad de obra es programas la referente a la creación del encoder óptico. Se ha necesitado del programa Inventor para el desarrollo del diseño; de los sensores ópticos para la toma de medidas; de la impresora Prusa Sliece para la impresión en 3D y por último, el tiempo empleado en su construcción y desarrollo.

Materiales:

CONCEPTO	UNIDAD	CANTIDAD	PRECIO (€)
INVENTOR	u	1	0
PRUSA SLICER	u	1	0
SENSORES ÓPTICOS	u	3	6,52
IMPRESORA PRUSA i3 MK3S+	mes	0.5	11,47
PLA(material de impresión)	g	68,56	0,03646
COMPUTADOR PERSONAL	mes	1	15

El programa se ha adquirido de forma gratuita, ya que se ha empleado la licencia de estudiante proporcionada por la Universidad Politécnica de Valencia.

La aplicación para el confeccionado de la impresión en 3D, Prusa Slicer, es una aplicación gratuita.

El precio asignado a la impresora prusa i3 MK3S+, es el precio total que ha costado la impresora (825,62€), dividido por su esperanza de vida (6 años), poniendo el caso de que la impresora se empleara 6 horas al día.

El material de la impresora 3D que se ha utilizado es el material del soporte más el del disco perforado.

Mano de obra:

CONCEPTO	UNIDAD	CANTIDAD	PRECIO (€)
IGENIERO JUNIOR	h	80	35
SUPERVISOR DEL PROYECTO	h	15	65

4. Redacción de memoria (Ud04)

Para la última unidad de obra se tendrá la redacción y confección del documento técnico pertinente al desarrollo del proyecto.

Materiales:

CONCEPTO	UNIDAD	CANTIDAD	PRECIO (€)
COMPUTADOR PERSONAL	mes	1.5	15
MICROSOFT WORD	u	3	6,52

Mano de obra:

CONCEPTO	UNIDAD	CANTIDAD	PRECIO (€)
IGENIERO JUNIOR	h	120	35
SUPERVISOR DEL PROYECTO	h	20	65

CAPÍTULO 3: CUADRO DE PRECIOS UNITARIOS DEL PROYECTO

CONCEPTO	UNIDAD	DESCRIPCIÓN	PRECIO (€)
Ud01	u	Circuito eléctrico que conecta los sensores ópticos con el ESP32	789,646
Ud02	u	Programación del ESP32	3.790
Ud03	u	Creación del encoder óptico	3.816.984
Ud04	u	Redacción de memoria	5.541.25

CAPÍTULO 4: PRECIOS DESCOMPUESTOS

En este capítulo se verán los precios de cada unidad de obra tras aplicarles los costes indirectos. Estos costes indirectos son costes que no están presentes dentro de cada unidad de obra, pero que son muy difíciles de cuantificar, por ello, se añaden en forma de porcentaje. Un ejemplo de estos costes indirectos sería el gasto de luz durante ese tiempo, o el precio del internet empleado. Por ello, se considerará adecuado asignar un 10% a los costes indirectos en este proyecto.

CONCEPTO	UNIDAD	DESCRIPCIÓN	CANTIDAD	PRECIO (€)	IMPORTE (€)
Ud01	u	Circuito eléctrico que conecta los sensores ópticos con el ESP32	1		
DOIT ESP32 DEVKIT	u	Microprocesador	1	6,9	6,9
PLACA DE CIRCUITO IMPRESO	u	Placa electrónica	1	14,12	14,12
CABEZAL DE PINES DE 40 PINES	u	Cabezal de pines con 40 pines machos	1	0,25	0,25

BORNES DE CONEXIÓN DE TORNILLO	u	Bornes de conexión de tornillo de 8 pines	2	0,208	0,416
ESTAÑO PARA SOLDADURA	g	Estaño	50	0,0496	2.48
SOLDADOR ELÉCTRICO	u	Soldador eléctrico	1	18,69	18,69
CARGADOR USB DE 5V	u	Fuente de alimentación de 5V	1	4,5	4,5
CABLE USB PARA ANDROID	u	Cable USB tipo C	1	2,29	2,29
IGENIERO JUNIOR	h		20	35	700
TECNICO DE LABORATORIO	h	Ayudante	2	20	40
		Costes directos	1.00		789,646
	%	**Costes indirectos**	10		78,9646
		Coste total			868.6106

CONCEPTO	UNIDAD	DESCRIPCIÓN	CANTIDAD	PRECIO (€)	IMPORTE (€)
Ud02	u	Programación del ESP32	1		
VISUAL STUDIO CODE	u	Plataforma de programación	1	0	0
AUTOMATION STUDIO	u	Herramienta de programación	1	0	0
COMPUTADOR PERSONAL	mes	Dispositivo electrónico para realizar el programa	1	15	15
IGENIERO JUNIOR	h		80	35	2800
SUPERVISOR DEL PROYECTO	h		15	65	975
		Costes directos	1,00		3.790
	%	**Costes indirectos**	10		379
		Coste total			4169

CONCEPTO	UNIDAD	DESCRIPCIÓN	CANTIDAD	PRECIO (€)	IMPORTE (€)
Ud03	u	Creación del encoder óptico	1		
INVENTOR	u	Herramienta de diseño en 3D	1	0	0
PRUSA SLICER	u	Herramienta de confeccionado de piezas en 3D	1	0	0
SENSORES ÓPTICOS	u	Optoacopladores	3	6,52	18,75
IMPRESORA PRUSA i3 MK3S+	mes	Impresora en 3D	0.5	11,47	5,73347
PLA	g	Material de impresión	68,56	0,03646	2,5
COMPUTADOR PERSONAL	mes	Dispositivo electrónico para realizar el diseño	1	15	15
IGENIERO JUNIOR	h		80	35	2.800
SUPERVISOR DEL PROYECTO	h		15	65	975
		Costes directos	1,00		3.816,984
	%	**Costes indirectos**	10		381,6984
		Coste total			4.198,6824

CONCEPTO	UNIDAD	DESCRIPCIÓN	CANTIDAD	PRECIO (€)	IMPORTE (€)
Ud04	u	Redacción de memoria	1		
COMPUTADOR PERSONAL	mes	Dispositivo electrónico para realizar la memoria	1.5	15	15
MICROSOFT WORD	u	Herramienta de escritura	3	6,52	18,75
IGENIERO JUNIOR	h		120	35	4.200
SUPERVISOR DEL PROYECTO	h		20	65	1.300
		Costes directos	1,00		5.541.25
	%	**Costes indirectos**	10		554,125
		Coste total			6095,375

CAPÍTULO 5: PRESUPUESTO DE EJECUCIÓN Y POR CONTRATA

CONCEPTO	COSTE (€)
PRESUPUESTO DE EJECUCIÓN MATERIAL	15.330,9856
Gastos generales (12%)	1.839,7183
Beneficio industrial (6%)	919,85
PRESUPUESTO DE EJECUCIÓN POR CONTRATA	18.090,563
IVA (21%)	3.799,02
PRESUPUESTO BASE DE LICITACIÓN	21.889,58

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

DESARROLLO DE UN SISTEMA DE REALIMENTACIÓN DE POSICIÓN Y VELOCIDAD PARA BANCO DE ENSAYOS DE MOTORES SÍNCRONOS CON IMANES PERMANENTES

ANEXO 1

CÓDIGO DEL PROGRAMA DE VISUAL STUDIO CODE Y DOCUMENTOS HTML PARA LA INTERFAZ DE LA WEB

Trabajo Final de Grado

ARNAU VICENT CRESPO

Junio, 2023

CÓDIGO DE PROGRAMACIÓN DE VISUAL STUDIO CODE:

```
#include <Arduino.h>
#include <Wire.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFS.h>
#include <WiFi.h>

const char* ssid = "BandRTestBench";
const char* password = "diedosii_servid";
//DIGIFIBRA-ee7H//7KcTYNGTF9

//Inicialización de los pines que se tomarán del ESP32 para visualizar la
información de los sensores ópticos
#define VP_PIN 36
#define VN_PIN 39
#define D34_PIN 34

//Inicializacion de las variables dentro del loop
unsigned int dtus=40; //tiempo de muestreo, se elige 40, para que pueda
mostrar bastantes puntos por vuelta
int cuenta=0;
int estado_ant1=0;//estado anterior de la variable estado, que es igual a la
información dada por el pin VP_PIN del ESP32
int estado_ant2=0;//estado anterior de la variable estado, que es igual a la
información dada por el pin VN_PIN del ESP32
int estado_ant3=0;//estado anterior de la variable estado, que es igual a la
información dada por el pin D34_PIN del ESP32
float w;//velocidad positiva
float w2;//velocidad negativa
float w_fil=0; //velocidad en rpm
float alpha=0.2;//factor para hacer un filtro pasabajos
int marc=0; //Es un marcador, que en caso de que la velocidad sea positiva es
0 y si es negativa 1
int negativo=0;// Indica si la sucesión de velocidades anteriores es negativa
o positiva

AsyncWebServer server(80);
String readvelocity() { //envía el valor de la velocidad
  if(marc==0)
  {
      w=w/10*60;
      w_fil=alpha*(w)+(1-alpha)*w_fil;
  }else
  {
       w2=w2/10*60;
      w_fil=alpha*(w2)+(1-alpha)*w_fil;
```

```
  }
    Serial.println(w_fil);
    return String(w_fil);
}


void setup()
{
  //activación de los pines del ESP32 para recibir información
  pinMode(VP_PIN,INPUT);
  pinMode(VN_PIN,INPUT);
  pinMode(D34_PIN,INPUT);

  //Muestra por pantalla que ha habido un error, en caso de que las
Bibliotecas SPIFFS no hayan funcionado
    if(!SPIFFS.begin()){
    Serial.println("un error ha ocurrido con los SPIFFS");
    return;
  }
  //Bucle que se repite hasta que se haya conectado a la red WiFi
  Serial.begin(115200);
   WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando a la red Wi-Fi..");
  }
  //Se obtiene el código para acceder a la página web donde se mostrarán los
datos.
   Serial.println(WiFi.localIP());
   server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html");
  });
  //Obtenemos el valor de la función readvelocity y lo representamos en la
gráfica
  server.on("/velocidad", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readvelocity().c_str());
  });
  // Start server
  server.begin();
}



void loop() {
//Se declara la variable nextus, que permitirá obtener la velocidad
controlando el tiempo que hay entre dos tomas de datos
  unsigned int nowus, nextus;
//Se define que nextus es igual al tiempo en microsegundos más 40
microsegundos
```

```
  nextus = micros()+dtus;
  int ct =0; //declaración del conteo
  int p0 = 0; //declaración de la posición anterior
  int p1; //declaración de la posición actual
  int cont0=0;//declaración de tres posiciones anteriores a la actual
  int cont1=0;//declaración de dos posiciones anteriores a la actual
  int incr1=0;//diferencia entre anterior vs dos posiciones anterior
  int incr2=0;//diferencia entre dos posiciones anterior vs tres posiciones
anterior
  int cont2=0;//declaración de cuatro posiciones anteriores a la actual
  int cont3=0;//declaración de cinco posiciones anteriores a la actual
  int incr3=0;//diferencia entre tres posiciones anterior vs cuatro posiciones
anterior
  int incr4=0;//diferencia entre cautro posiciones anterior vs cinco
posiciones anterior



  while(true){
    //Se asigna a las variables estados el valor que se da por el ESP32 en los
pines correspondienres (valores: 0 y 1)
    int estado1 = digitalRead(VP_PIN);
    int estado2 = digitalRead(VN_PIN);
    int estado3 = digitalRead(D34_PIN);

    if (((estado1==0) && (estado_ant2==0))||((estado1!=0) &&
(estado_ant2!=0)))
    {
    cuenta = cuenta+1; //Incremento
    }
    if (((estado2!=0) && (estado_ant1!=0))||((estado2==0) &&
(estado_ant1==0)))
    {
    cuenta = cuenta-1; //Decremento
    }

    //En caso de que la cuenta supere el 511, se reinicia la cuenta, por lo
que cuenta=0
    if(cuenta>511)
    {
      cuenta=0;
      negativo=0;
    /*Además se incluye la variable negativo, ya que en caso de que la cuenta
esté aumentando
    signifacará que el movimiento del motor será positivo.*/
    }
```

```
    //En caso de que la cuenta sea menor que 0, si disminuyera más, la
siguiente posición que encontrariamos es 511, no -1
    if(cuenta<0)
    {
      cuenta=511;
      negativo=1;
    /*Además se incluye la variable negativo, ya que en caso de que la cuenta
esté disminuyendo
    signifacará que el movimiento del motor será negativo.*/
    }

    //Cuando se pase por la ranura de referencia, la cuenta se reinicia
    if (estado3!=0)
    {
      cuenta=0;
    }

    //Se declara el estado anterior.
    estado_ant1=estado1;
    estado_ant2=estado2;
    estado_ant3=estado3;

    if(ct==0){
      p1 = cuenta;
      /*En caso de que el cuenteo sea 0, que sería el caso inicial,
       se declara que la posición 1 es la cuenta que se lleva.*/

      if((p1>=p0)&& (incr1>0) && (incr2>0) && (incr3>0) && (incr4>0)) {
        w = (p1-p0);
        /*Si la posición actual es mayor que la anterior,
        se declara que la velocidad es la diferencia de estas posiciones,
        en un tiempo determinado, es decir, el tiempo nextus*/

      }
      if((p1>=p0)&& (incr1<0) && (incr2<0)&& (incr3<0) && (incr4<0)){
        w2 = p1-511-p0;
        /*En caso de que sea menor la posición actual, que la anterior,
        sólo se tendrá que restar 512 a la actual, haciendo que
        la dieferencia sea positiva*/
      }
      if((p1<p0) && (incr1>0) && (incr2>0)&& (incr3>0) && (incr4>0)){
        w = 511+p1-p0;
        /*En caso de que sea menor la posición actual, que la anterior,
        sólo se tendrá que sumar 512 a la actual, haciendo que
        la diefrencia sea positiva*/
      }
      if((p1<=p0)&& (incr1<0) && (incr2<0)&& (incr3<0) && (incr4<0)){
        w2 = (p1-p0);
```

```
        /*Si la posición actual es mayor que la anterior,
        se declara que la velocidad es la diferencia de estas posiciones,
        en un tiempo determinado, es decir, el tiempo nextus*/
      }
      ct++;
      p0=p1;
      incr1=p0-cont1;
      cont1=p0;
      incr2=cont1-cont0;
      cont0=cont1;
      incr3=cont0-cont2;
      cont2=cont0;
      incr4=cont2-cont3;
      cont3=cont2;

      if(negativo!=0)
      {
        marc=1;
      }
      if(negativo==0)
      {
        marc=0;
      }
    }else ct++;
    if(ct==500)ct=0;


    //El proceso se detiene hasta que nextus sea menor que el tiempo que ha
pasado en microsegundos
    while(nextus > micros()){}

    nextus += dtus;

    }


}
```

DOCUMENTO HTML DE LA INTERFAZ DE LA WEB:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="https://code.highcharts.com/highcharts.js"></script>
    <style>
        body {
          min-width: 310px;
            max-width: 800px;
            height: 400px;
          margin: 0 auto;
        }
        h2 {
          font-family: Arial;
          font-size: 2.5rem;
          text-align: center;
        }
      </style>
</head>
<body>
    <h2>ESP velocidad del motor</h2>
    <div id="chart-velocidad" class="container"></div>

</body>
<script>
var chartT = new Highcharts.Chart({
  chart:{ renderTo : 'chart-velocidad' },
  title: { text: 'Velocidad motor asincrono de 2,2kw' },
  series: [{
    showInLegend: false,
    data: []
  }],
  plotOptions: {
    line: { animation: false,
      dataLabels: { enabled: true }
    },
    series: { color: '#059e8a' }
  },
  xAxis: { type: 'datetime',
    dateTimeLabelFormats: { second: '%H:%M:%S' }
  },
  yAxis: {
    title: { text: 'Velocidad (rpm)' }
    //title: { text: 'Velocidad (rev/min)' }
  },
```

```
  credits: { enabled: false }
});

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var x = (new Date()).getTime(),
          y = parseFloat(this.responseText);
      //console.log(this.responseText);
      if(chartT.series[0].data.length > 40) {
        chartT.series[0].addPoint([x, y], true, true, true);
      } else {
        chartT.series[0].addPoint([x, y], true, false, true);
      }
    }
  };
  xhttp.open("GET", "/velocidad", true);
  xhttp.send();
}, 1000) ;
</script>


</html>
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

DESARROLLO DE UN SISTEMA DE REALIMENTACIÓN DE POSICIÓN Y VELOCIDAD PARA BANCO DE ENSAYOS DE MOTORES SÍNCRONOS CON IMANES PERMANENTES

ANEXO 2

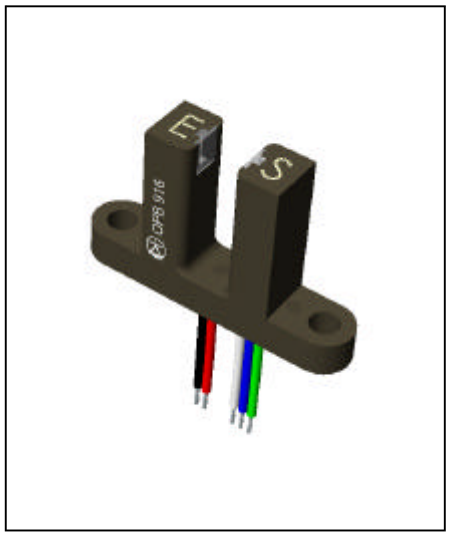
CATÁLOGO DE LOS SENSORES ÓPTICOS

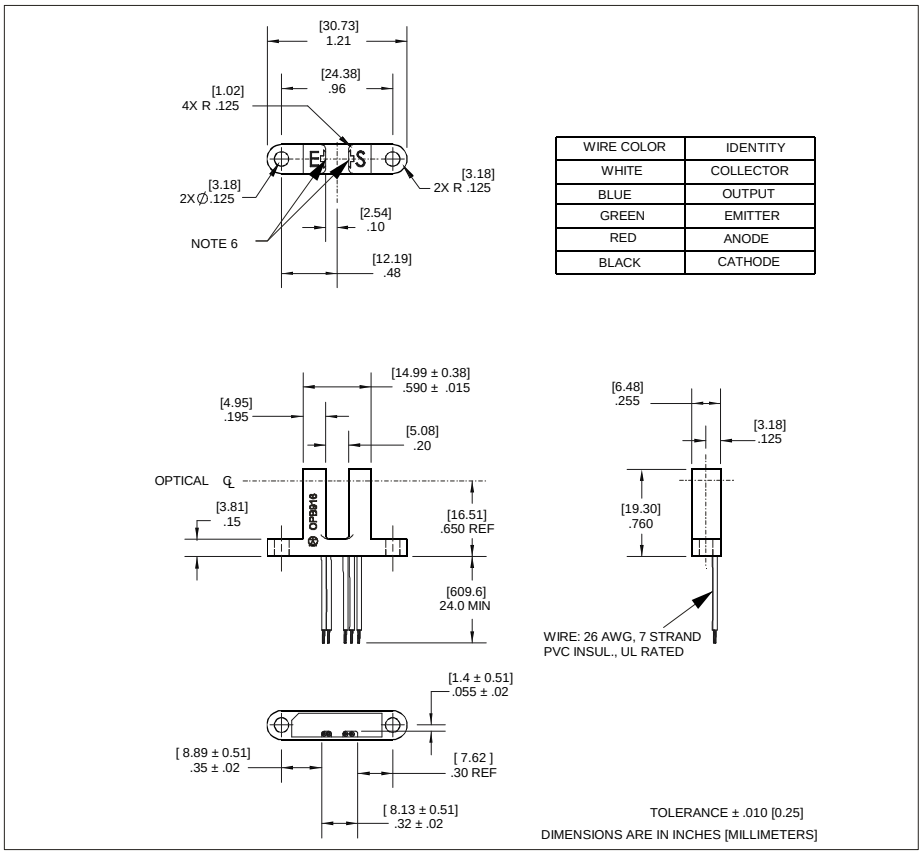
Trabajo Final de Grado

ARNAU VICENT CRESPO

Junio, 2023

Photologic® Slotted Optical Switches
Type OPB916 Series





WIRE COLOR	IDENTITY
WHITE	COLLECTOR
BLUE	OUTPUT
GREEN	EMITTER
RED	ANODE
BLACK	CATHODE

Features

- Choice of output configuration
- 24” min 26 AWG wires
- Low power consumption

Description

The OPB916 consists of an infrared emitting diode and a Photologic® photo integrated circuit mounted in an opaque housing with clear windows for dust protection. The deep slot allows for a longer reach of the optical path from the mounting plane, .650” (16.51 mm).

Internal apertures are .010” x 0.06” for the Photologic’s “S side” and .050” x 0.06” for the LED, “E side”. Two logic states and two electrical output types are available.

Custom electrical, wire or cabling are available. Contact your local representative or Optek for more information.

Absolute Maximum Ratings (T_A = 25° C unless otherwise noted)

Supply Voltage V_{CC} (Not to exceed 3 sec.) . . . . . 18 V
Storage and Operating Temperature Range . . . . . -40° C to +80° C
Input Diode Power Dissipation . . . . . 100 mW(1)
Output Photologic® Power Dissipation . . . . . 90 mW(2)
Voltage at Output Lead (Open Collector Output) . . . . . 35 V
Diode Forward D.C. Current . . . . . 50 mA
Diode Reverse D.C. Voltage . . . . . 2 V

NOTES:
(1) Derate linearly 1.67 mW/° C above 25° C.
(2) Derate linearly 2.67 mW/° C above 70° C.
(3) Clear dust protection.
(4) Normal application would be with light source blocked, simulated by I_F = 0 mA.
(5) All parameters tested using pulse technique.

PRECAUTIONS: Exposure of the plastic body to chlorinated hydrocarbons and ketones such as thread lock and instant adhesive products will degrade the plastic body. Cleaning agents methanol and isopropanol are recommended. Spray or wipe do not submerge.

Visit our website at www.optekinc.com or email us at sensors@optekinc.com

PART NUMBER GUIDE

O P B 9 1 6 X X X

Optek Assembly

Photologic® Photo Integrated Circuit

Optical Center Line .90”

Electrical Specification Variations

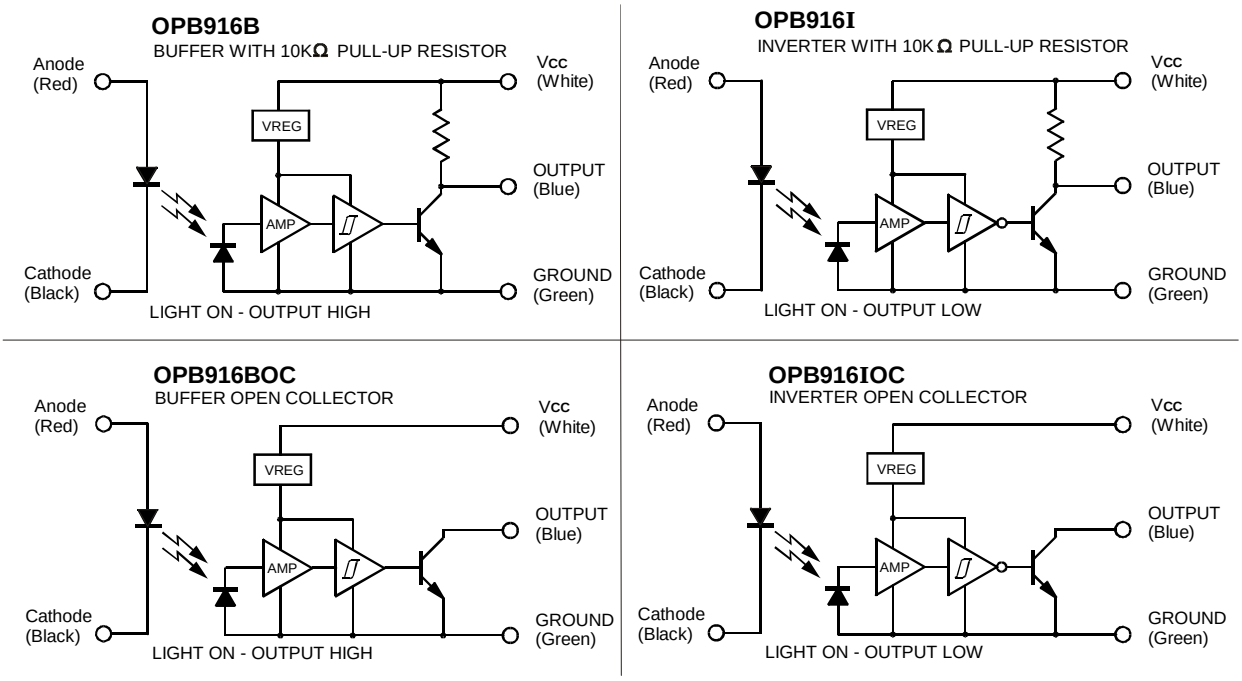
B - Buffered with 10K Ω pull-up

BOC - Buffered Open-Collector Output

I - Inverted with 10K Ω pull-up

IOC - Inverted Open-Collector Output

Schematics



Type OPB916 Series

Electrical Characteristics (T_A = 25° C unless otherwise noted)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
Input Diode						
V_F	**Forward Voltage**		1.30	1.80	V	I_F = 20 mA
I_R	**Reverse Current**			100	µA	V_R = 2 V
Phototransistor						
V_{CC}	**Operating D.C. Supply Voltage**	4.5		16.0	V	
I_{CCL}	**Low Level Supply Current:** OPB916B and OPB916BOC			7.0	mA	V_{CC} = 16 V, I_F = 0 mA(4) No Output Load
	OPB916I and OPB916IOC			7.0	mA	V_{CC} = 16 V, I_F = 10 mA No Output Load
I_{CCH}	**High Level Supply Current:** OPB916B and OPB916BOC			6.0	mA	V_{CC} = 16 V, I_F = 10 mA No Output Load
	OPB916I and OPB916IOC			6.0	mA	V_{CC} = 16 V, I_F = 0 mA(4) No Output Load
V_{OL}	**Low Level Output Voltage:** OPB916B			0.4	V	V_{CC} = 4.5 to 16 V, I_F = 0 mA(4) No Output Load
	OPB916BOC			0.4	V	V_{CC} = 4.5 to 16 V, I_F = 0 mA(4), IOL = 16 mA No Output Load
	OPB916I			0.4	V	V_{CC} = 4.5 to 16 V, I_F = 10 mA No Output Load
	OPB916IOC			0.4	V	V_{CC} = 4.5 to 16 V, I_F = 10 mA, IOL = 16 mA No Output Load
V_{OH}	**Low Level Output Voltage:** OPB916B	2.4	V_{cc} -1.5		V	V_{CC} = 4.5 to 16 V, I_F = 10 mA No Output Load
	OPB916I	2.4	V_{cc} -1.5		V	V_{CC} = 4.5 to16 V, I_F = 0 mA(4), No Output Load
I_{OH}	**High Level Output Current:** OPB916BOC		1.0	14	µA	V_{CC} = 4.5 V, V_{OH} = 30 V, I_F = 10 mA
	OPB916IOC		1.0	14	µA	V_{CC} = 4.5 V, V_{OH} = 30 V, I_F = 0 mA
$I_F(+)$	**LED Positive-Going Threshold Current:** OPB916B and OPB916I		5	10	mA	V_{CC} = 5 V No Output Load
	OPB916BOC and OPB916IOC		5	10	mA	V_{CC} = 5 V, IOL = 16 mA No Output Load
$I_F(+)/I_F(-)$	**Hysteresis**		1.5			
t_r, t_f	**Rise Time, Fall Time**		50		ns	V_{CC} = 5 V, I_F = 0 or 10 mA R_L = 300 Ω to 5 V, C_L = 50 pF
t_{PLH}, t_{PHL}	**Propagation Delay Low-High & High-Low**		3.0		µs	

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

DESARROLLO DE UN SISTEMA DE REALIMENTACIÓN DE POSICIÓN Y VELOCIDAD PARA BANCO DE ENSAYOS DE MOTORES SÍNCRONOS CON IMANES PERMANENTES

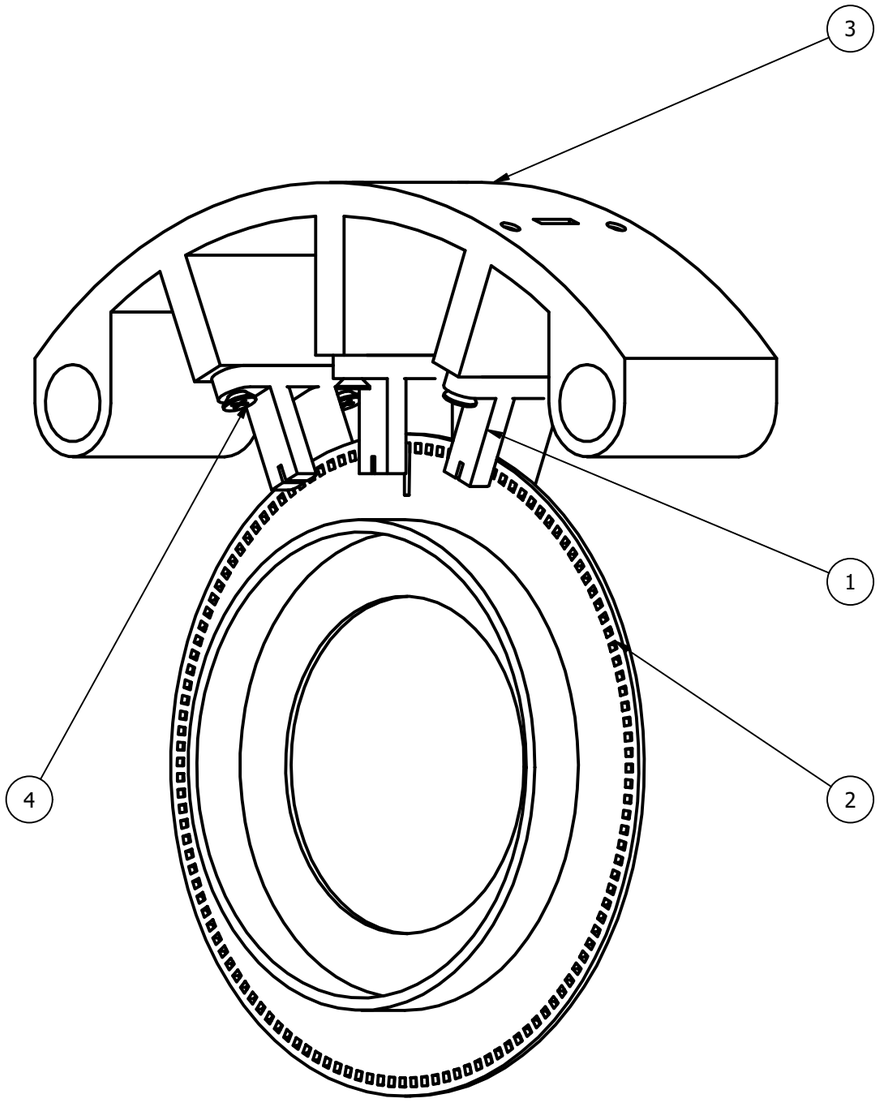
ANEXO 3

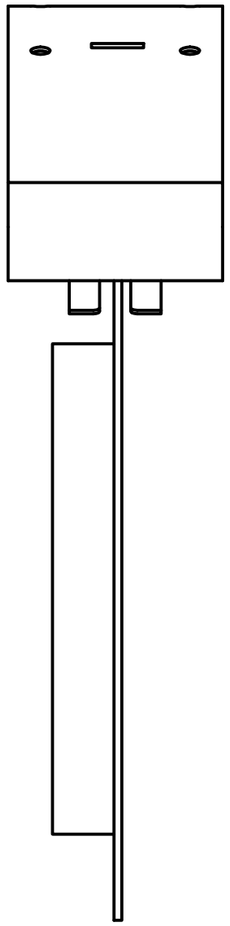
PLANOS DEL ENCODER ÓPTICO, DEL SOPORTE Y EL PLANO GENERAL

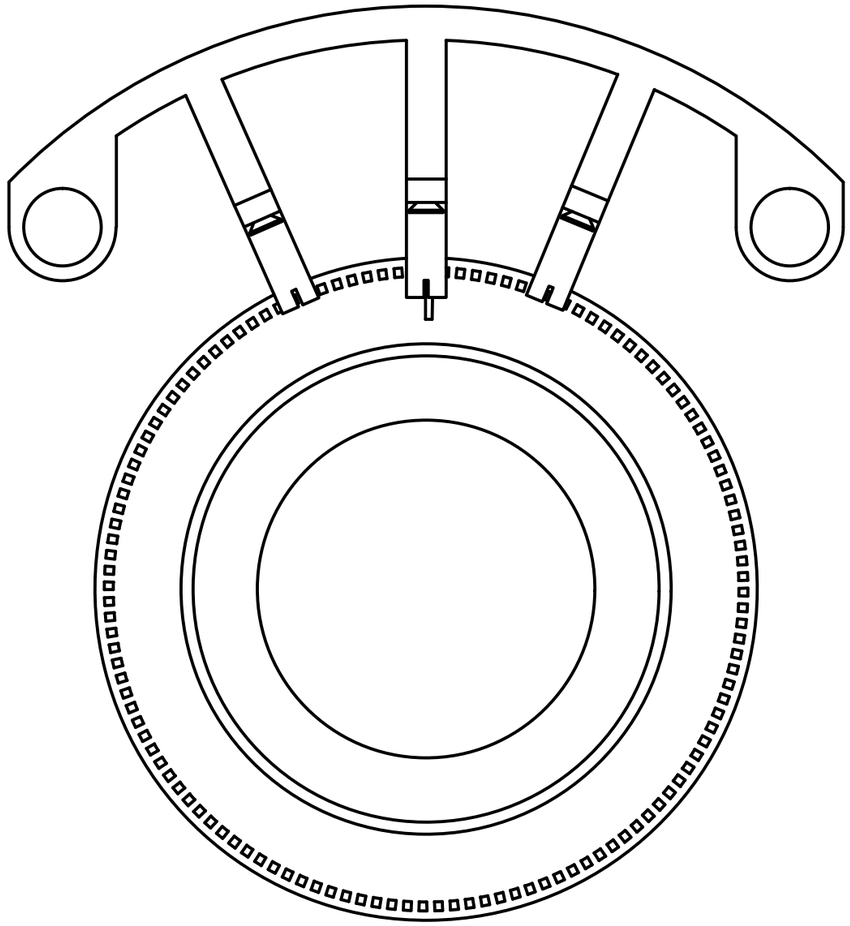
Trabajo Final de Grado

ARNAU VICENT CRESPO

Junio, 2023







LISTA DE PIEZAS			
ELEMENTO	CTDAD	Nº DE PIEZA	DESCRIPCIÓN
1	3	Sensor óptico	Sensor optoacoplador ranurado
2	1	Disco perforado de alta resolución	
3	1	Soporte	
4	6	ISO 7046-2 Z - M3x30 - 8.8 - Z1	Tornillos autorroscantes de cabeza avellanada plana con hueco cruciforme (cabeza común) - Clase A - Parte 2 con hueco cruciforme tipo H

TRABAJO FINAL DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

Título: Plano general

Tipo de documento: Académico

Escala: 1:1

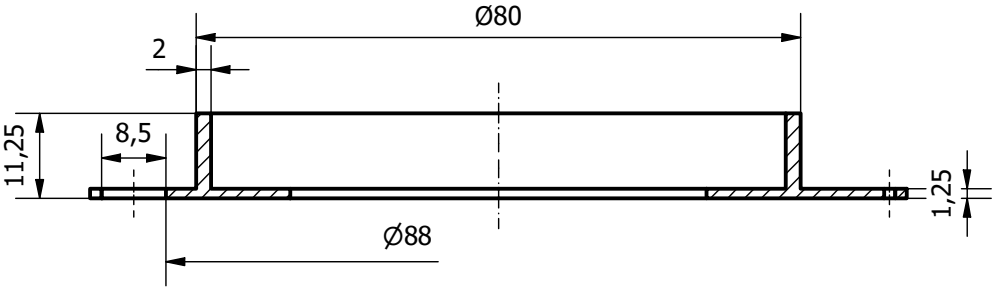
Fecha edición: Mayo 2023

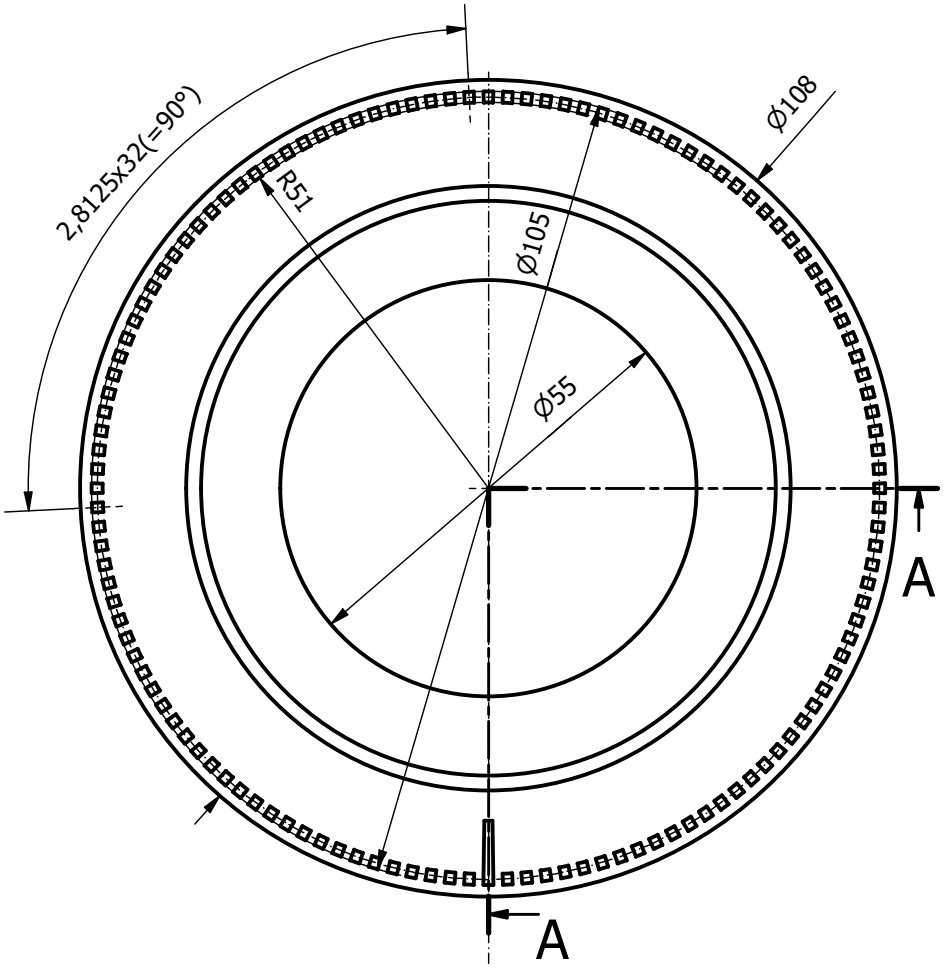
Creado por: Arnau Vicent Crespo

Aprobado por: Javier Andrés Martínez Román

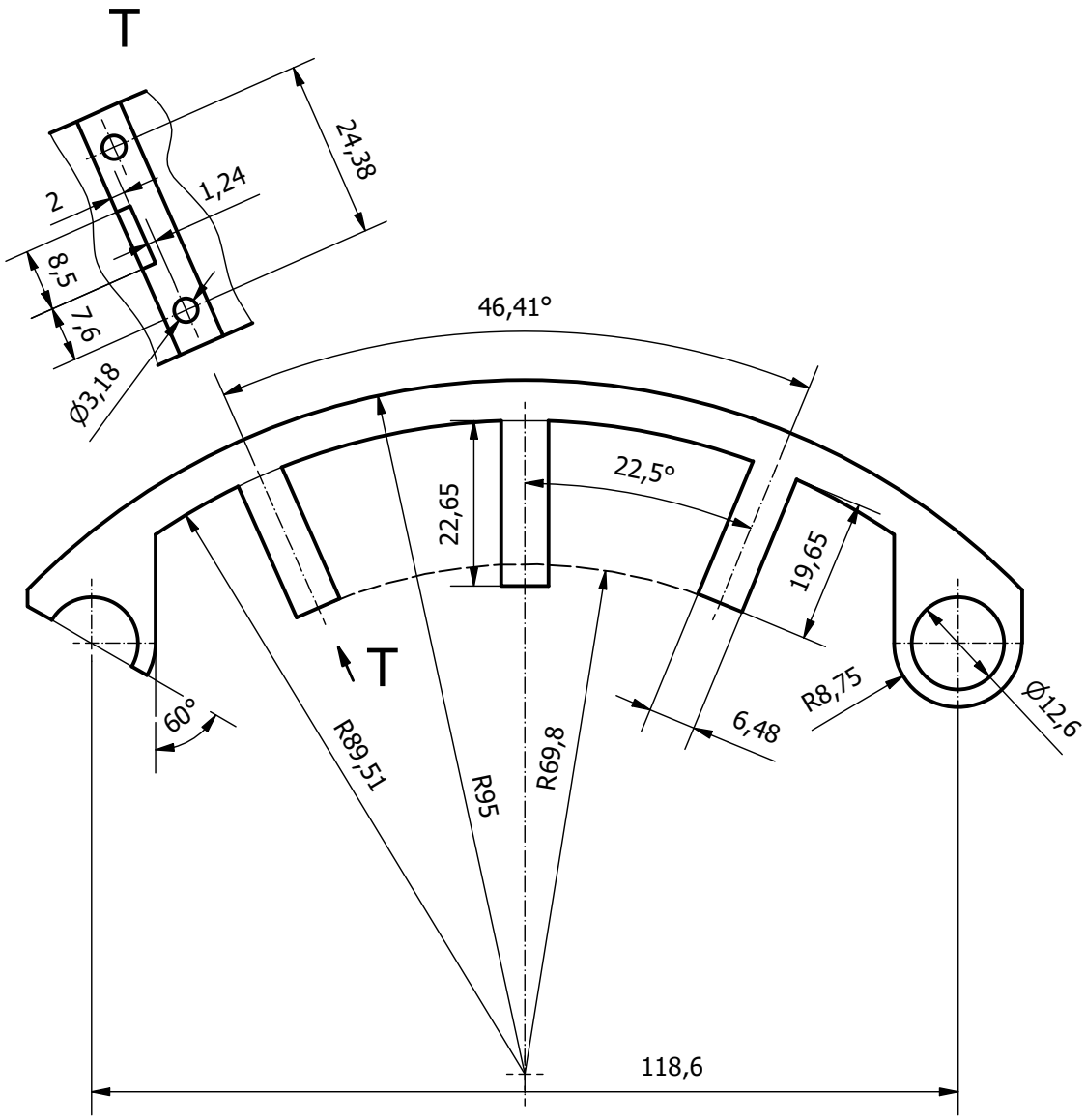
Hoja: 1/3

A-A (1:1)





TRABAJO FINAL DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES UNIVERSITAT POLITÈCNICA DE VALÈNCIA · ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA	Título: Disco perforado de alta calidad	
	Tipo de documento: Académico	Escala: 1:1
	Fecha edición: Mayo 2023	
Creado por: Arnau Vicent Crespo	Aprobado por: Javier Andrés Martínez Román	Hoja: 2/3



TRABAJO FINAL DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

Título: Soporte		
Tipo de documento: Académico		Escala: 1:1
Fecha edición: Mayo 2023		
Creado por: Arnau Vicent Crespo	Aprobado por: Javier Andrés Martínez Román	Hoja: 3/3