Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería Mecánica

Banco de Ensayos para Turbina Térmica destinado a Prácticas de Laboratorio.

Autor:

Adrián Martín Pintos

Tutor:

Dr. Melgar Bachiller, Andrés

Departamento de Ingeniería Energética

y Fluidomecánica

Valladolid, julio 2020

Resumen

El siguiente trabajo consiste la definición de especificaciones, diseño y construcción de un equipo para prácticas de laboratorio que permite ensayar de forma rápida diseños realizados por el alumnado de conjuntos estátor y rotor de una turbina térmica, fabricados mediante corte láser de chapa calibrada de acero.

El ensayo consiste en impulsar con aire comprimido la turbina y al mismo tiempo medir, a través de sensores gestionados por un controlador, la velocidad angular y las propiedades térmicas del aire, para, finalmente, obtener las magnitudes que caracterizan la turbina como potencia, par o rendimiento calculados mediante un software.

Se pretende que estos resultados sirvan como herramienta de estudio, comprensión e investigación por parte del alumnado de estas máquinas fluidomecánicas.

Summary

The following work consists of defining the specifications, design and construction of a laboratory equipment that allows to quickly test designs made by the students of stator and rotor assemblies of a thermal turbine, manufactured by laser cutting of calibrated sheet steel.

The test consists of propelling the turbine with compressed air and at the same time measuring, through sensors controlled by a controller, the angular speed and thermal properties of the air, to finally obtain the magnitudes that characterize the turbine as power, torque or performance calculated by software.

These results are intended to serve as a study, understanding and research tool for the students of these fluid-mechanical machines.

Palabras clave

Turbomáquina, Banco de Ensayos, Software, Fluidomecánica, Aceleración Libre, Diseño, Fabricación.

Keywords

Turbomachine, Test Bench, Software, Fluid Mechanics, Free Acceleration, Design, Manufacturing.

Índice general

Nomenclatura

α *aceleración angular*

α_1 *ángulo que forma la velocidad absoluta del fluido a la salida del estator*

α_2 *ángulo que forma la velocidad absoluta del fluido a la salida del rotor*

β_1 *ángulo que forma la velocidad relativa al rotor a la entrada del rotor*

β_2 *ángulo que forma la velocidad relativa al rotor a la salida del rotor*

γ *coeficiente de dilatación adiabática*

ω *velocidad angular* $(^{rad}/_{s})$

η_{iso} *rendimiento isoentrópico*

η_{mec} *rendimiento mecánico*

η_{total} *rendimiento total*

$\overrightarrow{c_1}$ *velocidad absoluta a la salida del estator* $(^{m}/_{s})$

$\overrightarrow{c_2}$ *velocidad absoluta a la salida del rotor* $(^{m}/_{s})$

C_0 *celeridad del fluido a la entrada del estator* $(^{m}/_{s})$

C_1 *celeridad del fluido a la salida del estator* $(^{m}/_{s})$

C_2 *celeridad del fluido a la salida del rotor* $(^{m}/_{s})$

C_p *calor específico a presión constante* $\left(^{J}/_{kg \cdot K}\right)$

h_0 *entalpía sensible a la entrada del estátor* $\left(^{J}/_{kg}\right)$

h_{00} *entalpía a la entrada del estátor* $\left(^{J}/_{kg}\right)$

h_1 *entalpía sensible a la salida del estátor* $\left(^{J}/_{kg}\right)$

h_{10} *entalpía a la salida del estátor* $\left(^{J}/_{kg}\right)$

h_2 *entalpía sensible a la salida del rotor* $\left(^{J}/_{kg}\right)$

h_{20} *entalpía a la salida del rotor* $\left(^{J}/_{kg}\right)$

$h_{2s'}$ *entalpía sensible de la evolución ideal* $\left({}^{J}/_{kg}\right)$

I *momento de inercia de la masa móvil respecto al eje de giro* $(kg \cdot m^2)$

J_F *cota funcional*

$\dot{m}$ *flujo másico* $({}^{kg}/_{s})$

M_{max} *par máximo que soporta la turbina* (Nm)

M_i *par indicado* $(N \cdot m)$

M_e *par efectivo* $(N \cdot m)$

M_p *par de perdidas* $(N \cdot m)$

$M_{p\ aceleración}$ *par de perdidas en aceleración* $(N \cdot m)$

$M_{p\ deceleración}$ *par de perdidas en deceleración* $(N \cdot m)$

P_0 *presión a la entrada del estator* (Pa)

P_1 *presión a la salida del estator* (Pa)

P_2 *presión a la salida del rotor* (Pa)

P_e *potencia efectiva* (W)

P_i *potencia indicada* (W)

T_0 *temperatura a la entrada del estator* (K)

T_1 *temperatura a la salida del estator* (K)

T_2 *temperatura a la salida del rotor* (K)

$\overrightarrow{u_1}$ *velocidad absoluta del estator en la entrada* $({}^{m}/_{s})$

$\overrightarrow{u_2}$ *velocidad absoluta del estator en la salida* $({}^{m}/_{s})$

$\overrightarrow{w_1}$ *velocidad del fluido relativa al rotor a la entrada* $({}^{m}/_{s})$

$\overrightarrow{w_2}$ *velocidad del fluido relativa al rotor a la salida* $({}^{m}/_{s})$

W_u *potencia específica* (W)

Capítulo 1 Introducción y objetivos

1.1 Antecedentes y justificación.

El siguiente trabajo está fundamentado en la asignatura de Máquinas Hidráulicas y Térmicas perteneciente al 4º curso del grado de Ingeniería Mecánica de la Universidad de Valladolid.

Como contenido docente, en esta asignatura se presentan conceptos sobre motores térmicos, de los cuales, una parte importante, usan las turbinas como elemento fundamental. A lo largo de la asignatura se realiza un análisis termodinámico y fluidomecánico para, finalmente, sentar las bases del diseño y cálculo de las condiciones de funcionamiento, dentro y fuera de los parámetros ideales de operación.

Las actividades docentes de carácter práctico basadas en turbinas son difíciles de abordar, debido a alto coste de estos equipos, la gran cantidad de subsistemas ligados a estos, su complejidad y las grandes potencias de los equipos producidos industrialmente, incluso en los tamaños comercializados más reducidos.

En la actualidad, gracias al desarrollo del CAD, CAM y la tecnología de corte de metal mediante láser, es posible fabricar piezas de geometrías planas complejas con costes, tiempos de ejecución y tolerancias muy reducidas. Esto permite disponer de elementos con precisión dimensional del orden de decimas de milímetro, una distribución de masas equilibrada y permitiendo ajustes suficientes sin auxilio de otro proceso de fabricación posterior.

El concepto de aceleración libre permite ensayar un conjunto móvil sin necesidad de equipos externos, reduciendo el tamaño y los subsistemas ligados al ensayo. Este modelo de ensayo consiste en accionar el conjunto y dejar que desacelere monitorizando su velocidad en todo momento, deduciendo de esta manera su potencia y perdidas mecánicas.

Por otro lado, la capacidad de almacenamiento de energía por parte de los gases hace posible accionar un conjunto de mayor potencia por un periodo de tiempo determinado. El acceso a aire comprimido es común en los laboratorios técnicos, permitiendo alimentar sistemas con facilidad y mediante un fluido inocuo.

Con todo esto, se plantea la definición de una práctica que consiste en que el alumnado diseñe el conjunto de rotor y estátor de una turbina radial en base a los conocimientos adquiridos durante las primeras 5 semanas de la asignatura, disponiendo de 5 semanas para la definición y finalmente en las últimas 5 semanas se procederá a su ensayo.

Las **áreas del conocimiento** impartidas en el grado, así como las fuentes externas involucradas en esta idea, son variadas y han resultado indispensables para su desarrollo. Las principales son:

- *Diseño mecánico y fabricación:* son nociones indispensables para diseñar y ejecutar el proyecto mecánico en su conjunto, el cual es la parte principal y fundamental donde pretenden extraen y monitorizar los datos a tratar.
- *Mecánica de fluidos y termodinámica:* como herramienta básica a la hora de aportar el sustento teórico necesario del funcionamiento de las maquinas accionadas por fluidos, aplicando esta óptica para obtener un criterio constructivo válido.
- *Electrónica e informática:* sirviendo como medio para implementar los receptores necesarios y tratando sus señales para generar los datos que son procesados mediante un software que realiza los cálculos de las propiedades que se desean analizar.

Complementando estos principios, se hace posible concebir la idea general de realizar un conjunto mecánico como se muestra en la *Figura 1*,basado en el diseño de una turbina simplificada, la cual sea capaz de rotar a una alta velocidad angular y permita estudiar su comportamiento, siendo captado por sensores y tratado posteriormente para generar una interfaz que muestre las propiedades físicas requeridas al usuario con motivo de facilitar y afianzar la comprensión de los conceptos teóricos del funcionamiento de este tipo de máquinas.

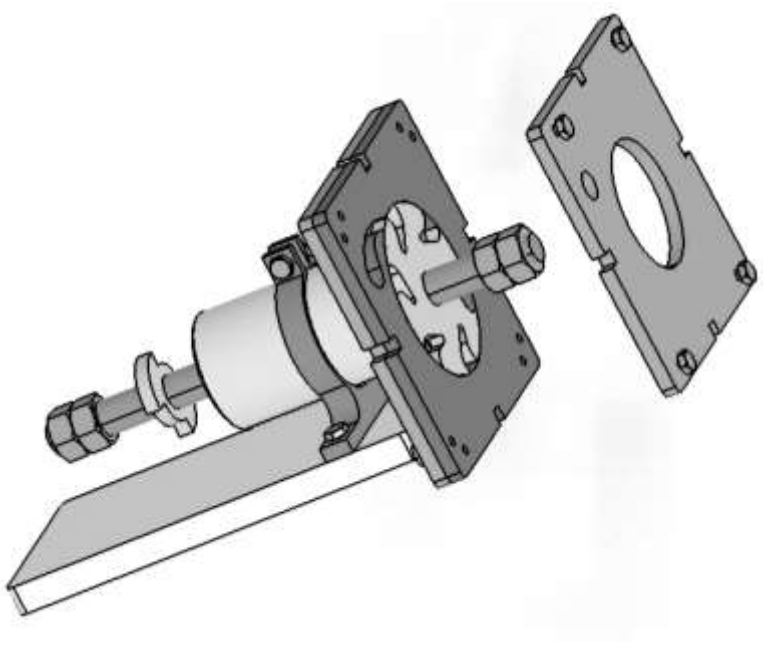


Figura 1. Conjunto mecánico del banco de ensayos.

1.2 Objetivos

En cuanto a las metas fijadas en este trabajo, se distinguen las siguientes:

- *Definición mecánica y eléctrica*: acotación de las condiciones de funcionamiento del conjunto y de las magnitudes que lo caracterizan, derivando en la concreción del diseño mecánico y eléctrico.
- *Construcción*: consistente en materializar la turbina propulsada mediante aire capaz de soportar altas revoluciones, compuesta por piezas normalizadas o de geometría sencilla que faciliten tanto su fabricación como su montaje, permitiendo que el usuario interactúe y manipule el mecanismo de manera intuitiva y segura.
- *Implementación electrónica y de software:* basado en el diseño previo, materializar la captación y el tratamiento de datos que ilustren las magnitudes que caracterizan la turbina.
- *Realización de pruebas de verificación*: se pretende ensayar el conjunto para conocer sus limitaciones y verificar el correcto funcionamiento del mismo.
- *Construcción de una herramienta útil:* como finalidad transversal en este trabajo, pretendiendo ser un complemento versátil, útil e ilustrativo de la docencia en el ámbito del estudio de las máquinas térmicas.

1.3 Distribución de la memoria

De modo preliminar, la visión del contenido del que trata cada capítulo de esta memoria se resume de la siguiente forma:

- *Capítulo 1. Introducción y objetivos:* se expone la razón por la cual este trabajo es puesto en marcha y cuáles son sus objetivos a alcanzar finalizado el mismo.

- *Capítulo 2. Fundamentos teóricos:* pretende dar una base teórica que mejore la comprensión del desarrollo del proyecto, concatenando conceptos que resultan en una línea deductiva mediante la cual se vertebra el trabajo en una orientación técnica. Del mismo modo, expone el funcionamiento de las herramientas y componentes implementados para generar el conjunto.

- *Capítulo 3. Diseño mecánico, fabricación y montaje:* siendo este punto donde se condensa la información del apartado anterior en un diseño y los procesos para ser ejecutado, exponiendo su criterio constructivo, características técnicas y funcionamiento.

- *Capítulo 4. Diseño y montaje electrónico:* expone el ensamble de elementos electrónicos necesarios, su justificación y funcionamiento.

- *Capítulo 5. Microprocesador, software y tratamiento de datos:* expone la explicación de la secuencia de transferencia de datos y los cálculos efectuados, así como la forma en la que se imprimen mediante el hardware en forma de interfaz de usuario.

- *Capítulo 6. Estudio económico:* refleja los costes de ingeniería, materiales y ejecución del proyecto.

- *Capítulo 7. Ensayos:* se plasma la intención, condiciones y ejecución de distintas pruebas del conjunto para, en un capítulo posterior, analizar las evidencias obtenidas.

- *Capítulo 8. Resultados y conclusiones:* tanto del epígrafe anterior como del resto del proceso, se extrae un balance global en forma de resultados y, a raíz de estos, se evalúan y comparan con los objetivos iniciales, deduciéndose a modo de corolario las conclusiones. También se añaden las líneas futuras de desarrollo que completen y enriquezcan el presente modelo.

- *Capítulo 9. Bibliografía*: se acopian los títulos y referencias del material consultado durante la elaboración que hayan sido especialmente ilustrativos.

Capítulo 2
Fundamentos teóricos de las turbomáquinas.

2.1 Introducción.

En primer lugar, para conformar una idea general acerca del tema principal a tratar, las turbomáquinas térmicas, se incluye a continuación los conocimientos esenciales sobre esta rama de la ingeniería.

Se define turbomáquina como todos aquellos conjuntos mecánicos que interactúan con un fluido de manera continua, transformando el estado termodinámico del mismo mediante la variación del momento cinético debida a elementos solidarios a un eje dispuestos de manera radial.

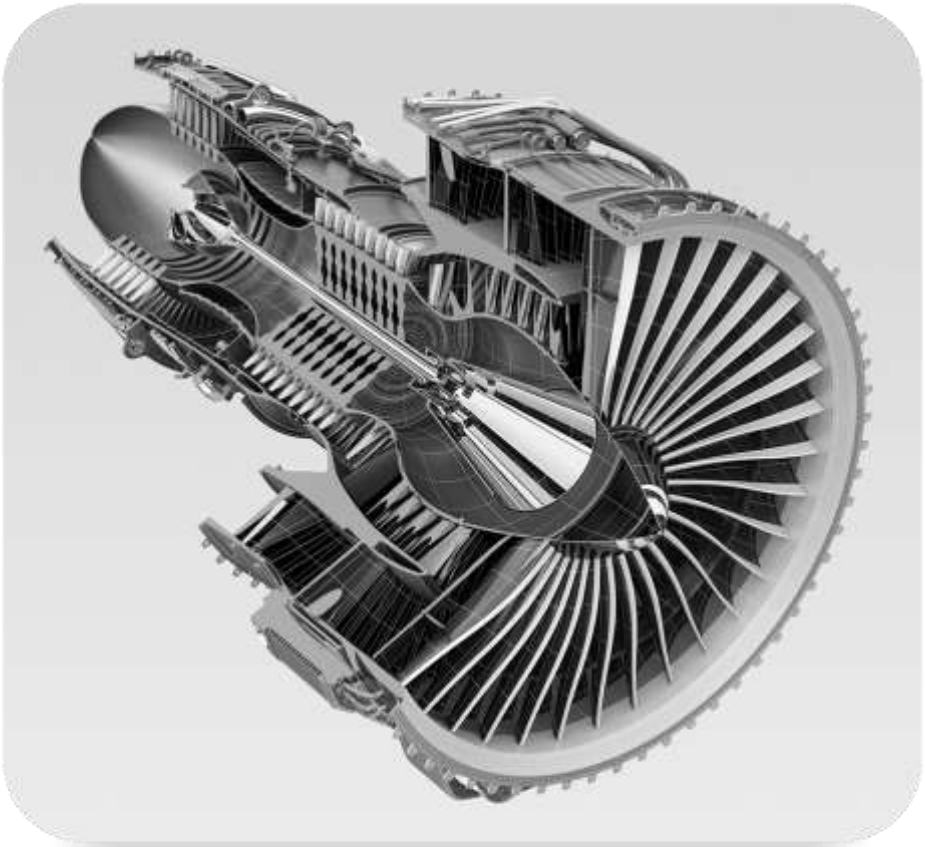


Figura 2. Sección de un turbofan.

Con efecto de acotar la definición anterior, se plantea una posible clasificación en función del tipo de fluido con el cual trabajan y su evolución. Si el fluido es incompresible se denominan máquinas hidráulicas y térmicas cuando si lo es. Por otra parte, si se extrae trabajo del conjunto se denominan máquinas motoras y en el caso contrario máquinas generadoras.

Atendiendo a estos parámetros, este estudio se centrará en las **turbomáquinas térmicas motoras**.

Describiendo brevemente el funcionamiento y la composición del conjunto, se parte de un fluido el cual ha sido llevado a un nivel de mayor energía, haciendo posible que evolucione hacia un estadio inferior energéticamente. Esto se consigue, comúnmente, mediante un proceso de combustión (elevando su temperatura) y con ayuda de una turbomáquina térmica generadora (aumentando su presión).

Partiendo del estado energético inicial, el fluido transforma parte de su entalpía en velocidad dentro del **estátor**, dirigiendo el flujo hacia el **rotor**. Este último transforma, mediante el cambio de dirección del fluido, el salto energético en trabajo, pudiendo producirse en este también un proceso de expansión.

A las superficies específicas situadas en ambas partes por las cuales el fluido evoluciona se denominan **álabes**. Según la disposición de estos elementos y la dirección que toma el flujo, se distinguen en axial o axial-radiales.

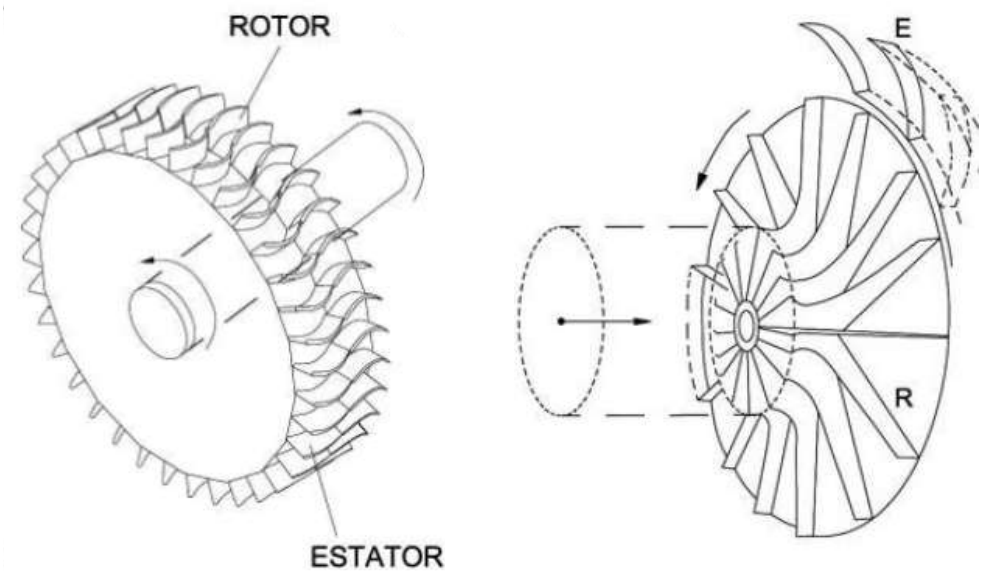


Figura 3. Escalonamiento axial y axial-radial.

(Tomado de: MUÑOZ, M. y ROVIRA, A. J. Máquinas térmicas. U.N.E.D., 2011)

Usualmente, este conjunto rotor y estátor se le denomina **escalonamiento**. Los escalonamientos se concatenan para conseguir unas mejores prestaciones de la máquina térmica.

Una vez definido el funcionamiento básico de estos conjuntos, se realizará su estudio termodinámico y cinemático, con objeto de deducir las variables que determinan su funcionamiento. Posteriormente a este análisis, se simplificarán dando forma al modelo llevado a cabo.

2.2 Estudio termodinámico.

Un punto de vista desde cual comenzar el estudio de este tipo de máquinas es mediante la termodinámica. Desde esta visión, se toman los estados termodinámicos del fluido en los puntos más importantes para así evaluar su evolución a lo largo del escalonamiento.

Se fijan tres puntos de los cuales obtener el estado energético del fluido: antes del estátor (0), entre el estátor y el rotor (1); y después del rotor (2).

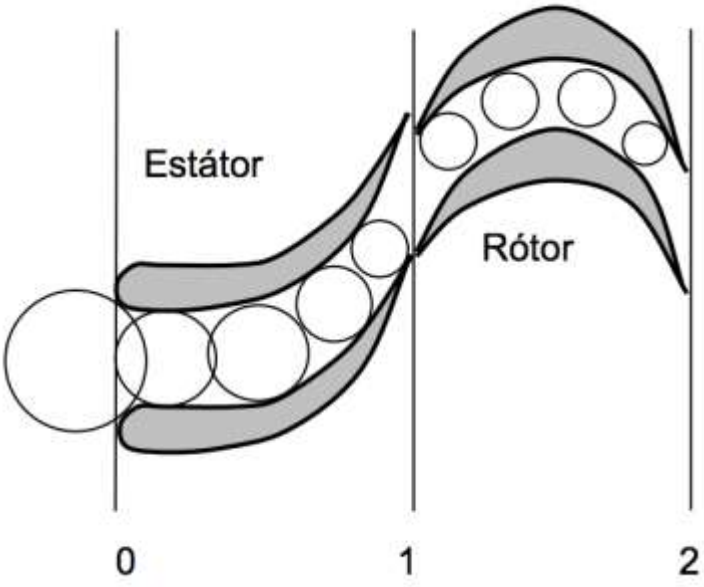


Figura 4. Sección transversal de estátor y rotor.

(Apuntes de Máquinas Térmicas, departamento de ingeniería energética y fluidomecánica)

Esta evolución, normalmente, queda reflejada en un gráfico que tiene en el eje de las abscisas la entropía y en el de las ordenadas la entalpía.

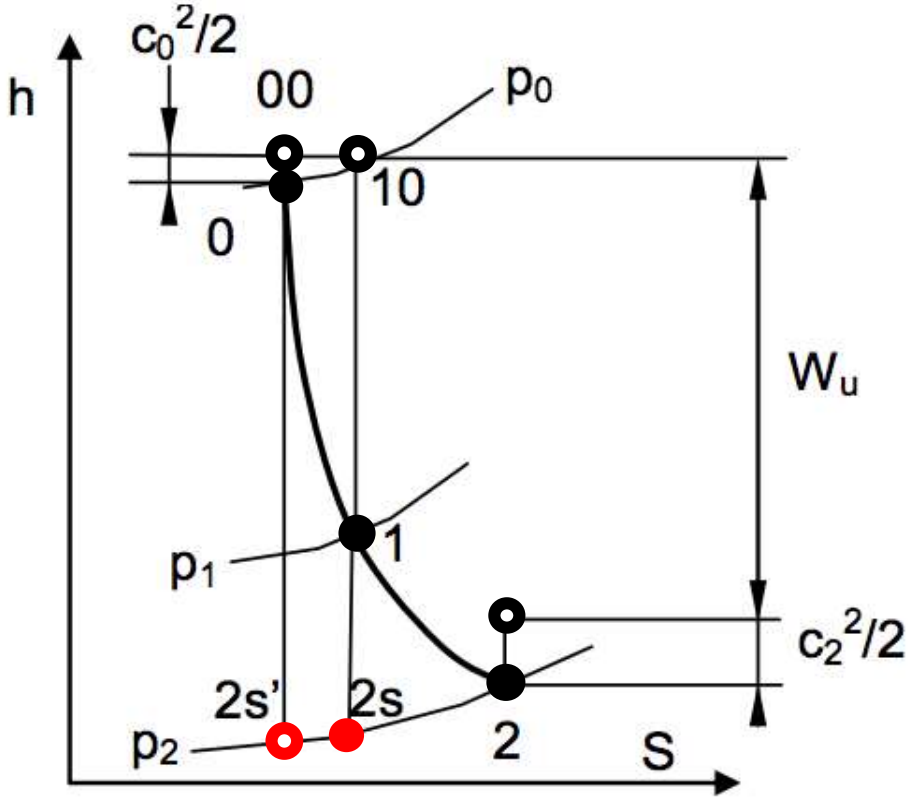


Figura 5. Diagrama de evolución termodinámica entropía-entalpía.

Se parte del fluido en el estado 0 ●, el cual posee una entalpía sensible (sin tener en cuenta la energía cinética) h_0. Teniendo en cuenta la energía debida a la velocidad del fluido en ese punto su estado sería 00 ⭘ con $h_{00} = h_0 + \frac{{C_0}^2}{2}$.

El fluido evoluciona desde 0 ● a 1 ● en el estátor, disminuyendo su presión y aumentando su velocidad ($\frac{{C_1}^2}{2} = h_{10} - h_1$), para ,posteriormente ,atravesar el rotor donde se puede producir o no una expansión (dependiendo del tipo de escalonamiento) representado el estado del fluido a la salida con el punto 2 ● y un nivel energético sensible h_2.

A través de estos estados, se podrá determinar el **trabajo específico** (W_u) extraído del escalonamiento, tomando el flujo másico constante, como:

$$W_u = h_{00} - h_{20}$$

Los puntos $2s'$ ⭘ y $2s$ ● representan la evolución del fluido en un estado ideal, una evolución que no genere entropía, y, a partir de ellos, se podrá calcular el **rendimiento isoentrópico** (η_{iso}) del escalonamiento:

$$\eta_{iso} = \frac{h_{00} - h_{20}}{h_{00} - h_{2s'}}$$

Las magnitudes η_{iso} y W_u son las que caracterizan el escalonamiento, proporcionando la información más importante de su funcionamiento.

2.3 Estudio cinemático.

La otra óptica mediante la cual se evalúa el funcionamiento de un escalonamiento es mediante la **evolución cinética** del fluido. Desde este análisis se hace posible relacionar la geometría del conjunto con su desempeño.

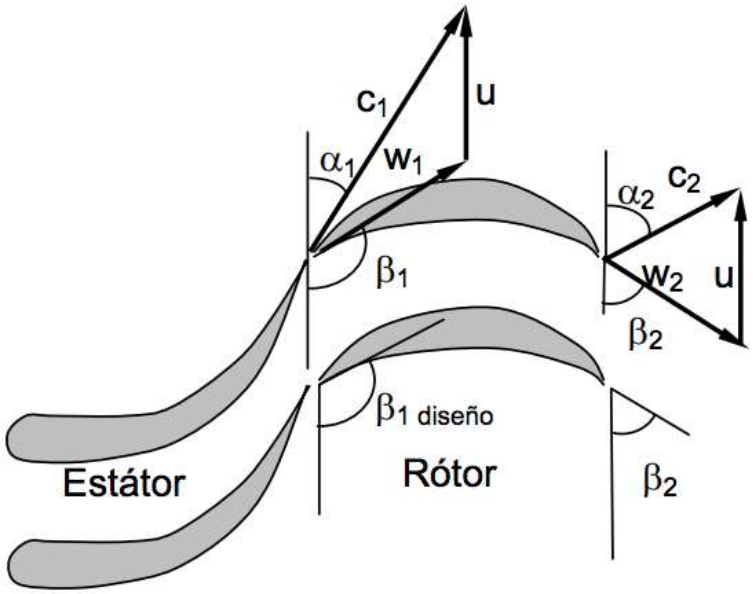


Figura 6. Sección transversal de estátor y rotor con la representación de los vectores que caracterizan la velocidad del fluido.

(Apuntes de Máquinas Térmicas, departamento de ingeniería energética y fluidomecánica)

A la salida del estátor la velocidad es de $|\vec{c}|$ siendo α_1 la dirección con la que sale del mismo, dependiendo así el estado del fluido de la geometría, que, por un lado, marcará el ángulo de salida del estátor y, por otro, la expansión que el fluido sufrirá por la variación de área.

La velocidad $\vec{c}$ se puede descomponer en la velocidad del flujo relativa al rotor ($\vec{u}$) y la velocidad del rotor en ese punto ($\vec{w}$). La salida posee un ángulo de β_2, el cual depende de la geometría de los álabes del rotor, mientras que β_1 y α_2 dependerán de las condiciones de funcionamiento, las cuales pueden variar.

A través de este planteamiento es posible caracterizar parte del funcionamiento del escalonamiento. Complementando el estudio cinemático con el termodinámico se establecen las relaciones necesarias para el diseño geométrico, las cuales serán expuestas en puntos posteriores.

2.4 Aceleración libre.

La **aceleración libre** consiste en accionar un conjunto hasta una velocidad determinada para, posteriormente, cesar la acción sobre el mismo, dejándolo desacelerar libremente. El cuerpo aumenta su velocidad debido a que la fuerza resultante ejercida sobre el mismo es mayor que cero. Cuando la acción cesa, el cuerpo se detiene debido a que existe una fuerza resultante que se opone al movimiento. Es posible hallar estas fuerzas calculando la aceleración mediante la monitorización de la velocidad y el tiempo.

Si es un cuerpo en rotación y el momento de inercia (I) es conocido, es posible calcular el **par efectivo** (M_e) que desarrolla el conjunto, así como el **par de pérdidas** (M_p) y el **par indicado** (M_i) mediante la **aceleración angular** (α).

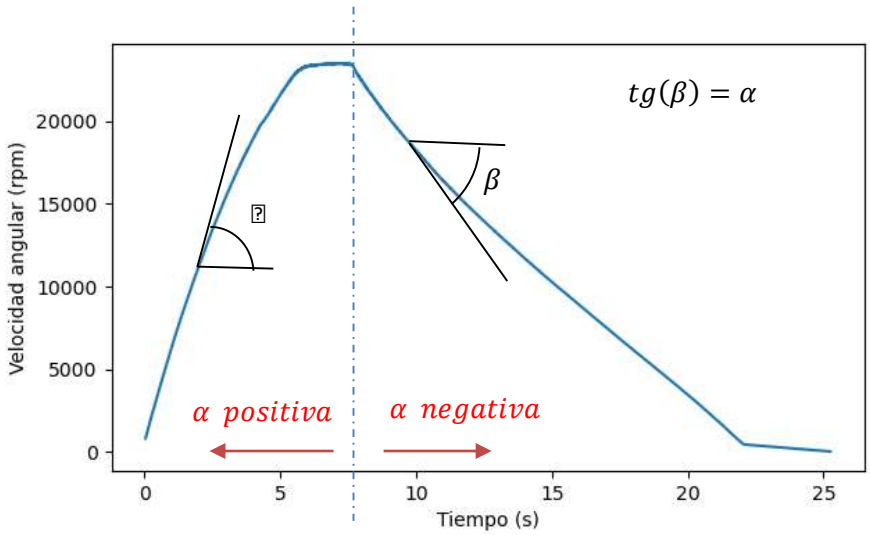


Figura 7. Gráfica tiempo-velocidad angular de un ensayo real.

Este método permite conocer características del conjunto sin la necesidad de equipos externos, reduciendo el tamaño de las instalaciones necesarias para el ensayo. Por otro lado, se encuentra limitado por la incapacidad de probar distintos estados a velocidad constante.

Las expresiones que rigen este método quedan divididas en dos partes, la parte donde la aceleración es positiva y donde es negativa:

- Positiva: $M_e = M_i - M_{p\,aceleración} = I \cdot \alpha$
- Negativa: $M_{p\,deceleración} = I \cdot \alpha$

Con motivo de hacer posible la relación entre el proceso de aceleración y desaceleración es necesario que dichas magnitudes sean dependientes de la velocidad angular.

2.5 Simplificación.

En base a lo expuesto en los puntos anteriores, se pretende identificar las magnitudes necesarias para caracterizar una turbina. Con motivo de la simplificación del modelo, se suponen una serie de **hipótesis**:

- El fluido de operación (aire) se comporta como **gas perfecto**, haciendo que los calores específicos (C_p y C_v) no dependan de la temperatura ni de la presión. Se realiza esta afirmación debido a la poca variación de las propiedades en el rango de temperaturas y presiones de trabajo. Se fija $\gamma = 1{,}4$ y $C_p = 1007\ {}^{J}/_{kg \cdot K}$.
- El proceso es **cuasi-estacionario**, por tanto, el flujo másico deberá ser el mismo en la entrada y salida de la turbina. Las posibles pérdidas del conjunto son ínfimas en comparación con la masa de aire que lo atraviesa.
- La **energía cinética es nula** a la entrada de la turbina. Los sensores son situados en puntos de remanso para medir presión y temperatura con velocidad despreciable.
- Las **pérdidas** dependen únicamente de la velocidad angular, siendo similares en el proceso de aceleración y deceleración.
- La presión de salida del conjunto (P_2) es contante en el tiempo y es la presión atmosférica.

En base a estas hipótesis y aplicándolas a los desarrollos teóricos anteriores, se replantean las expresiones que caracterizan la turbina, teniendo en cuenta que cada una de ellas depende del tiempo:

- *Termodinámica*:
  El valor del trabajo específico (W_u) se puede calcular como:

$$W_u = h_2 - h_0 = C_p \cdot (T_2 - T_0)$$

  A través del flujo másico ($\dot{m}$) se concreta la potencia indicada de la turbina:

$$P_i = W_u \cdot \dot{m}$$

  Para hacer posible el cálculo del rendimiento isentrópico (η_{iso}), es necesario plantear la expansión adiabática desde el estado 0 al estado 2, con el fin de hallar $T_{2s'}$:

$$T_{2s'} = T_0 \cdot \left(\frac{P_2}{P_0}\right)^{\frac{\gamma-1}{\gamma}} \qquad \eta_{iso} = \frac{T_2 - T_0}{T_{2s'} - T_0} = \frac{P_i}{P_{ideal}}$$

- *Aceleración libre:* En el proceso de aceleración se deduce, basándose en las hipótesis expuestas anteriormente:

  $$M_e = M_i - M_p = I \cdot \alpha$$

  En el proceso de deceleración se deduce:

  $$M_p = I \cdot \alpha$$

  Con la velocidad angular se puede deducir las potencias:

  $$P_e = M_e \cdot \omega \;\; ; \;\; P_i = M_i \cdot \omega \;\; ; \;\; P_p = M_p \cdot \omega$$

  Es posible calcular el rendimiento mecánico (η_{iso}) y el rendimiento total (η_{total}):

  $$\eta_{mec} = \frac{P_e}{P_i} \; ; \qquad \eta_{ideal} = \frac{P_e}{P_{ideal}} \; ; \qquad \eta_{total} = \eta_{iso} \cdot \eta_{mec}$$

Las magnitudes de interés que **caracterizarán la turbina** son: $P_i(\omega)$, $M_i(\omega)$, $P_e(\omega)$, $\eta_{iso}(\omega)$, $\eta_{mec}(\omega)$ y $\eta_{total}(\omega)$

Las **variables** que deben ser **monitorizadas** a partir de las cuales es posible el cálculo de las anteriores son: $T_0(t)$, $P_0(t)$, $T_2(t)$, $\omega(t)$ y t.

Capítulo 3
Diseño mecánico, fabricación y montaje

3.1 Concepto inicial

En base a lo expuesto anteriormente, la primera cuestión a tratar es que tipo de turbomáquina térmica se pretende obtener. Esta disyuntiva se da debido a que las maquinas axiales tienen un análisis cinemático más intuitivo, pero requieren una mayor complejidad de fabricación y de manera contrapuesta se contemplan las máquinas radial-axiales.

Debido al coste de fabricación, se opta por la **máquina radial-axial**, fijando así uno de los elementos más importantes del conjunto, el cual condicionará parte del resto del diseño del mismo.

Se pretende realizar mediante **tecnología de corte láser** para conseguir esta geometría y se constituye en este punto que este proceso de fabricación sea extendido a todo el conjunto, por su precisión y coste.

Por otra parte, el estátor y el rotor deben confinarse para evitar la pérdida de presión del gas que lo acciona, proporcionando las vías de entrada y salida de aire.

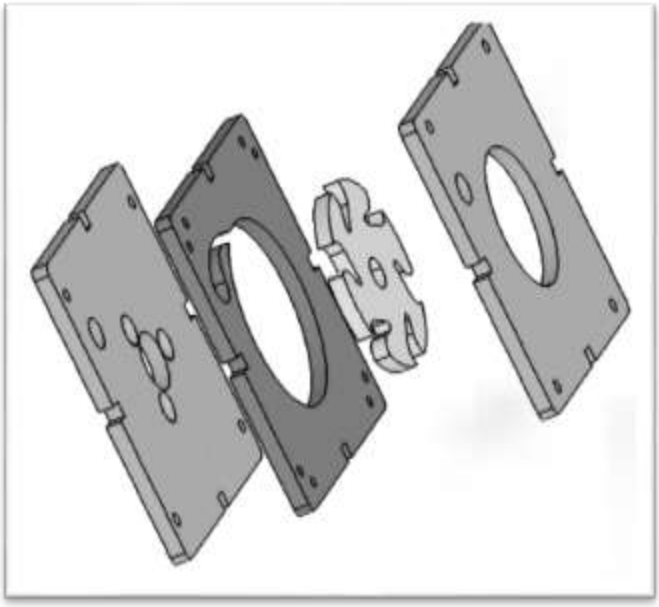


Figura 8. Rotor, estátor y tapas.

La turbina debe tener rotación libre, por tanto, es necesario fijarla a la bancada mediante rodamientos. Todo el conjunto se mantiene solidario al eje mediante

casquillos que, debido a la compresión que ejercen las tuercas de los extremos del eje, fijan cada elemento mediante fricción.

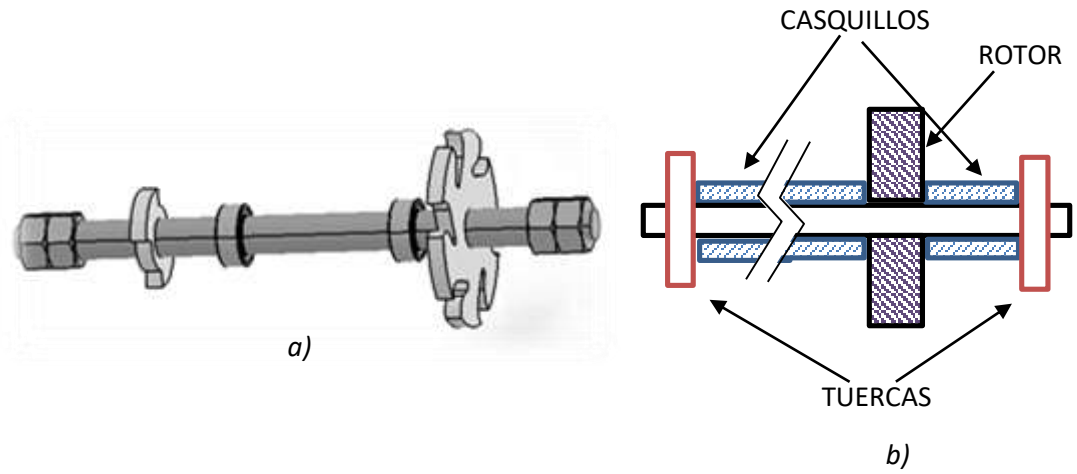


Figura 9. a) Montaje de elementos solidarios al eje. b) Sección longitudinal al eje.

No tiene por objeto este trabajo optimizar la geometría del rotor y del estátor para alcanzar unas características determinadas, por tanto, se contempla una geometría básica de los mismos para lograr que consiga una velocidad angular suficiente como para que se compruebe que el proyecto funciona correctamente, dejando a estudios del alumnado el diseño óptimo de estos elementos.

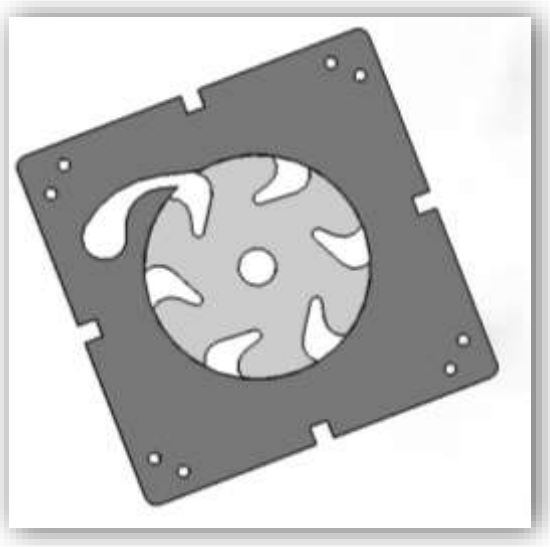


Figura 10. Estátor y rotor.

Se fijan dos premisas fundamentales a la hora de realizar el diseño: la velocidad angular límite 100.000 rpm y se trabajará con aire comprimido. Las características del fluido serán los comunes de una instalación de aire comprimido, teniendo como presión máxima de 14 bar y una temperatura del fluido de 300K.

Figura 11. Fotografía frontal del prototipo llevado a cabo.

Una vez alcanzado este esbozo de lo que será la turbomáquina, entran en juego el resto de variables del diseño condensando en el producto final.

3.2 Criterios constructivos

A lo largo de este punto se pretende destacar los factores más importantes a tener en cuenta la hora de realizar el diseño completo, teniendo como base los esbozos anteriores, analizando cada uno de los condicionantes de su puesta en servicio y vida útil. A continuación, se enumeran las premisas del diseño:

- *Facilidad de montaje y desmontaje:* se pretende que el conjunto sea manipulado frecuentemente por el usuario, intercambiando distintos diseños

de estátor y rotor, posibilitando su estudio y análisis. Esta es la motivación por la cual el conjunto debe constar de unas características mecánicas y geométricas suficientes que permitan esta interactuación.
Para cumplir este objetivo, los elementos deben ser capaces de soportar la manipulación continua, deben tener un ajuste entre elementos y un volumen adecuado para facilitar y asegurar el correcto funcionamiento del conjunto.

- *Versatilidad:* el conjunto debe ser fácilmente adaptable a distintos tipos de ensayo y modificaciones.

- *Bajo coste:* por esta parte, se pretende que la fabricación de las piezas sea moderadamente económica y permita la fabricación de distintos modelos de manera que no suponga un esfuerzo económico conseguirlo.
  Mediante elementos normalizados y la tecnología de corte por láser se conseguirá este propósito, obteniendo piezas de geométricas complejas, con tolerancias suficientes y a un coste bajo.

- *Capaz de soportar las solicitudes mecánicas:* cabe mencionar que este es un punto del proceso de diseño de cualquier elemento mecánico muy importante. En este caso pasa a un segundo plano, debido a que realizar un diseño ajustado a los esfuerzos mecánicos resultaría en una dificultad inherente de fabricación y montaje (con su repercusión en coste), trabajando con piezas con ajustes precisos, con unas medidas reducidas que dificultarían el montaje y haciendo el conjunto más sensible a errores humanos de manipulación.

  Por tanto, no se realiza un estudio mecánico exhaustivo a lo largo del proyecto, únicamente se realizará en los puntos donde las condiciones sean estrictas.

3.3 Diseño geométrico

Entendiendo el **diseño** como un proceso en el que concurren e influyen varios procesos simultáneos, a la hora de desarrollar la geometría se tiene presente cuál será su proceso de fabricación, ajuste, montaje y uso.

Atendiendo a la fabricación, como se ha remarcado anteriormente, se pretende fabricar mediante tecnología de corte láser, esto hará que la turbina este formada por piezas de geometría plana y espesor constante.

Por otro lado, la utilización de elementos de fabricación normalizada será de ayuda a la hora de conseguir al menor coste de las piezas necesarias para formar el conjunto.

La primera geometría a definir será el **rotor**. Se conoce de puntos anteriores la forma preliminar que debe obtenerse, siendo necesario concretarla. El centro de masas de este elemento debe situarse en el eje de giro, el cual debe ser eje principal de inercia para evitar vibraciones.

Figura 12. Rotor.

El radio del rotor viene dado por la velocidad angular máxima que alcanzará y la velocidad del flujo.

La velocidad que adquiere el aire depende de la geometría del estátor, situando su límite superior en el caso de bloqueo de tobera, donde el flujo alcanza la velocidad del sonido. En caso de que se produjese una onda de choque, la velocidad del flujo posterior a esta sería subsónica.

Para realizar el cálculo se supondrá que el estátor se encuentra en el límite de bloqueo, donde la velocidad de salida (c_1) es la del sonido, donde se hará uso de las expresiones que describen la situación de "tobera bloqueada":

$$T_1 = T_0 \frac{2}{\gamma + 1} \quad ; \qquad c_1 = \sqrt{\gamma \cdot R \cdot T_1}$$

Donde $T_0 = 300K$, $\gamma = 1{,}4$ y $R = 287{,}1\ {}^{J}/_{kg \cdot K}$. Sustituyendo y operando se obtiene que $c_1 = 317{,}0\ {}^{m}/_{s}$.

Si el estátor posee un ángulo de salida de $\alpha_1 = 90º$ entonces la velocidad del punto exterior del rodete (u_1) tendría una velocidad máxima igual a c_1. Conociendo el límite de velocidad angular fijado por la capacidad de los rodamientos, detallado en el

apartado de solicitaciones mecánicas, y fijado el límite de la turbina en 100.000 rpm ($10472{,}0\ {}^{rad}/_{s}$):

$$r = \frac{\omega}{c_1} = \ sustituyendo \ = \ \mathbf{3{,}0 \cdot 10^{-2} m}$$

El espesor del rodete condicionará el área de salida del flujo, por tanto, para minimizar la alimentación necesaria, se pretenderá que este sea lo menor posible. Debido a que sobre el rotor deberá roscar los tornillos que aseguren las tapas, es necesario que posea espesor suficiente para este propósito.

Las deformaciones debidas a la velocidad angular máxima que sufre el rotor son despreciables, y, por tanto, no se tienen en cuenta a la hora de especificar el espesor de este elemento.

La tornillería será de un tamaño que facilite su manejo, ya que un diámetro nominal menor cumpliría con las solicitaciones mecánicas sin problema. El tamaño por el que se opta es M4, por tanto, el espesor debe ser mayor que 4mm para que la unión sea segura, tomando el criterio de la bibliografía para elementos roscados en acero:

$$longitud\ roscada = \emptyset_n = 4mm$$

El espesor comercializado de chapa calibrada inmediatamente superior a esta medida y el cual es seleccionado como espesor del rotor y estátor es 5mm.

El siguiente elemento del conjunto a concretar su dimensión será el eje. Esta pieza es crucial a la hora del diseño del resto, por tanto, su diseño involucra múltiples factores a tener en cuenta:

- *Dimensión normalizada:* debido a que, como posteriormente se detalla, este elemento está acoplado con otros elementos normalizados, debe tener una dimensión para la cual existan dichos componentes (casquillos y rodamientos). Las medidas más comunes de varillas calibradas son: 6, 8, 10, 12 mm. Por otro lado, los diámetros más comunes de casquillos y rodamientos son 8, 10 y 12mm por tanto el eje deberá situarse en estas medidas.

- *Mecanización sencilla:* tanto a la hora de conseguir útiles de mecanizado como en el proceso de fabricación, el diámetro nominal del eje es un factor a tener en cuenta. En este sentido, cuanto mayor diámetro posea dentro de los anteriormente mencionados, más sencilla será su fijación mediante el proceso de mecanizado.

- *Debe resistir los esfuerzos mecánicos:* todos estos diámetros mencionados anteriormente superan con creces sus características a las solicitudes mecánicas requeridas durante su funcionamiento normal. Es por ello que no se realiza un análisis exhaustivo a esta pieza.

Una vez planteados estos puntos, se opta por un **diámetro de eje de 10 mm.**

Las dimensiones de la **rueda fónica** destinada a evidenciar la velocidad angular, deben ser compartibles con los sensores destinados a este propósito. El centro de masas de este elemento debe situarse en el eje de giro, el cual debe ser eje principal de inercia para evitar vibraciones.

La profundidad del diente oscila entre los 5 – 8 mm y el espesor oscila entre 3 - 8 mm en los elementos comercializados. Se fija en **5 mm** ambas medidas y el diámetro exterior en 30mm para la interferencia con el bastidor.

Figura 13. Rueda fónica.

3.4 Solicitaciones mecánicas

Los esfuerzos mecánicos de la mayor parte de elementos del conjunto no son limitantes ya que el resto de criterios constructivos son más restrictivos. No obstante, es necesario conocer las características de algunos elementos con el fin de concretar las capacidades del conjunto.

Se pretende que la velocidad angular que alcance la turbina sea alta, por tanto, los rodamientos seleccionados para el conjunto deben ser capaces de soportarlo. Se fija en el diseño preliminar una velocidad angular de 100.000 rpm.

Las características de los rodamientos, para el diámetro de eje de 10mm, son:

DIÁMETRO INTERIOR (MM)	DIÁMETRO EXTERIOR (MM)	ESPESOR (MM)	VELOCIDAD DE REFERENCIA (RPM)	DESIGNACIÓN
10	15	3	85.000	W 61700
10	15	3	85.000	W61700 R
10	19	5	80.000	61800
10	19	7	80.000	W 63800
10	22	6	70.000	61900

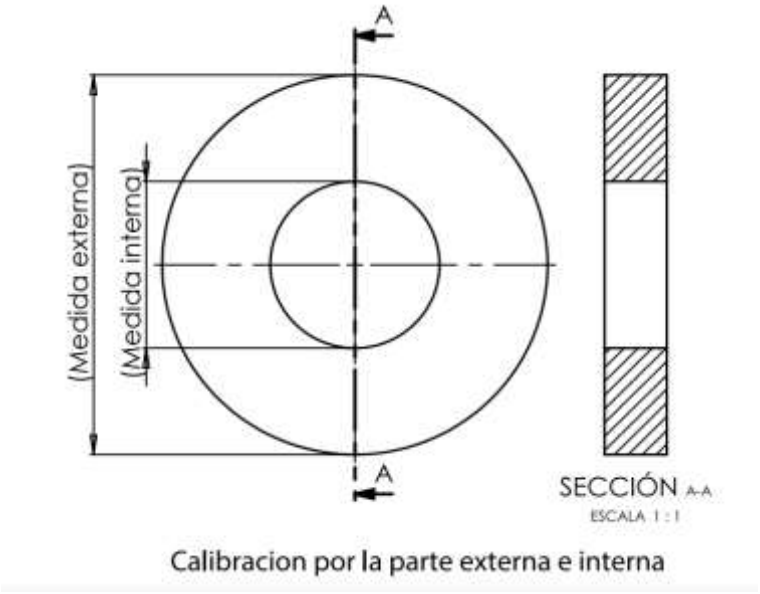
Tabla 1. Características de rodamientos SKF.

La capacidad térmica (85.000 rpm) es menor que la fijada en el concepto inicial (100.000 rpm), por lo que la vida útil del rodamiento se verá reducida si trabajase a este régimen. Debido a que los ensayos son cortos en el tiempo y las pérdidas del conjunto harán que no sea posible llegar al límite de velocidad, se selecciona el rodamiento **W 63800**, teniendo en cuenta que este elemento es posible que necesite ser sustituido con mayor frecuencia que el resto.

3.5 Ajustes

El correcto funcionamiento del conjunto vendrá dado por la definición de los ajustes entre los elementos donde esta sea crucial.

En primer lugar, se hallará el **ajuste entre rotor y estátor**. Tomando como referencia las tolerancias de fabricación de la casa *WESCO* en cuanto al corte láser de acero de espesor 5mm:



Tolerancia externa: -0,1 +0,3

Tolerancia interna: -0,3 +0,1

Figura 14. Intervalo de tolerancias de fabricación por corte láser de WESCO

Mediante estos datos, y fijando como juego mínimo entre estátor y rotor en $\boldsymbol{Jm = 0,1\ mm}$ es posible calcular las cotas de ambas piezas. Este juego permitirá un montaje sencillo y permitirá la libre rotación.

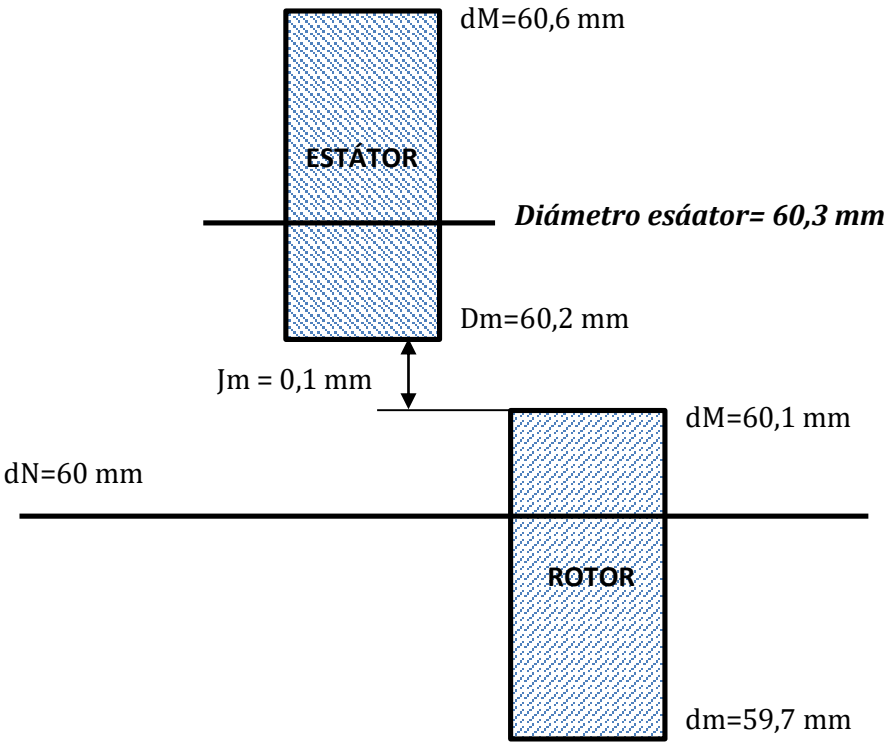


Figura 15. Representación del ajuste entre rotor y estátor.

Los cálculos, conociendo el diámetro del rotor calculado anteriormente ($dN = 60\ mm$) serán:

$$Diametro\ mayor\ rotor = dM = dN + intervalo\ de\ tolerancia\ superior = 60 + 0{,}1 = 60{,}1\ mm$$

$$Diámetro\ menor\ estator = Dm = dm + Jm = 60{,}1 + 0{,}1 = 60{,}2mm$$

$$Diámetro\ nominal\ del\ estator = DN = Dm + intervalo\ de\ tolerancia\ inferior = 60{,}2 + 0{,}1 = \boldsymbol{60{,}3\ mm}$$

La medida final del diámetro del estátor será: $\boldsymbol{DN = 60{,}3^{+0{,}3}_{-0{,}1}\ mm}$.

Por otra parte, el **ajuste entre el rotor y las tapas** vendrá dado por unas juntas que se situarán entre el estátor y las tapas.

Se realizará un cálculo preliminar de cuál será el espesor de dichas juntas, Se fijan en un espesor comercial de **0,25mm**. El cálculo se realizará mediante la acotación funcional, suponiendo el caso en el cual se fija este como su dimensión nominal con el fin de esclarecer las tolerancias entre el rotor y las tapas (J_F):

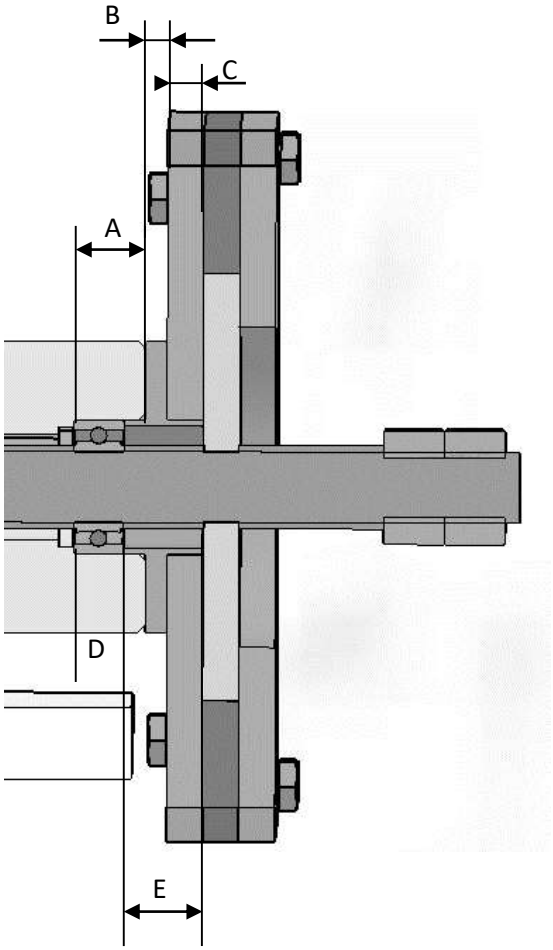


Figura 16. Corte longitudinal con acotación funcional.

Se despreciará el intervalo de tolerancias del rodamiento, ya que es del orden de μm, cuando el resto se sitúa en décimas de milímetro.

$$J_F(nominal) = D + E - (A + B + C) = 7 + 11{,}25 - (10 + 3 + 5) = 0{,}25mm$$

$$J_F(máximo) = D + E - (A + B + C) = 7 + 11{,}3 - (9{,}95 + 3 + 4{,}9) = 0{,}45mm$$

$$J_F(mínimo) = D + E - (A + B + C) = 7 + 11{,}2 - (10 + 3{,}05 + 5{,}1) = 0{,}05mm$$

De esta forma se asegura la no interferencia del rotor con las tapas. Teniendo en cuenta este ajuste, a la hora de realizar el montaje, es necesario concretar el **par de apriete (T)** que fijaran los elementos mediante la fricción con los casquillos.

El valor mínimo de juego entre el rotor y las tapas es de 0,05mm, por tanto, se fijará la deformación máxima del casquillo situado entre el rodamiento (cota *E*) y el rotor en la mitad de este juego (0,025mm). De esta forma se deja lugar a la dispersión propia del ajuste de par de apriete asegurando el correcto funcionamiento.

Conociendo el material, las dimensiones del casquillo y el desplazamiento máximo permitido es posible calcular la fuerza que produce esta deformación:

$$F = \frac{E \cdot A}{L} u = \frac{7 \cdot 10^{10} \left({}^{N}/_{m^2}\right) * \pi(0{,}006^2 - 0{,}005^2)(m^2)}{0{,}01125\ (m)} 0{,}025 \cdot 10^{-3} = 5375{,}61\ N$$

El par de apriete necesario para conseguir esta fuerza de compresión se calcula mediante la siguiente expresión, donde el coeficiente de torsión del perno (k) se estima en 0,15 y el diámetro nominal de la rosca es $10\ mm$:

$$\boldsymbol{T} = k \cdot F \cdot d = sustituyendo = 0{,}15 \cdot 5375{,}61 \cdot 0{,}01 = \boldsymbol{8{,}06\ Nm}$$

Como se aprecia este par de apriete queda fuera del rango normalizado ($58 - 72\ Nm$). Las contratuercas deberán tener un torque cercano a estos valores normalizados.

Esta fuerza de compresión permite el cálculo del **par máximo ($\boldsymbol{M_{max}}$)** que soportará el conjunto sin que el rotor pierda la condición de adherencia con los casquillos. Para ello se integra un diferencial de superficie:

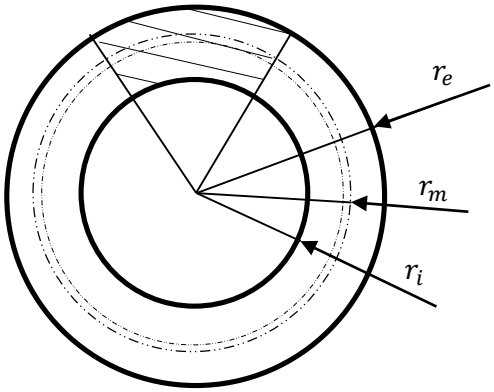


Figura 17. Superficie de apoyo de casquillo.

$$M_{max} = 2 \cdot \mu \cdot F \cdot r_m \;;\; \begin{cases} \mu = 0{,}61\ (estimación) \\ F = 5375{,}61\ N \\ r_m = \sqrt{\dfrac{{r_e}^2 - {r_i}^2}{2}} \end{cases} ;\; \boldsymbol{M_{max} = 15{,}38\ Nm}$$

3.6 Medio ambiente en servicio, reciclaje y seguridad.

El ambiente de trabajo para el cual se piensa destinar el banco de ensayo es un laboratorio técnico, por tanto, un medio poco agresivo. Las temperaturas de operación, así como las condiciones de humedad son moderadas, por lo que el proceso de corrosión que sufrirán las superficies metálicas es controlado.

Se recomienda, una vez realizados los ensayos, aplicar una fina capa de aceite con el fin de proteger las superficies frente a la posible corrosión que pueda sufrir. Es desaconsejable la aplicación de pinturas o esmaltes para evitar alterar el ajuste de las superficies.

Los elementos que forman el conjunto superan con creces las solicitaciones mecánicas, por tanto, la vida útil se prevé extensa. No obstante, en el caso de que sea necesario el remplazo de alguna de las partes, es posible su sustitución, alargando la operatividad del esto del conjunto

En cuanto al reciclaje, las piezas están formadas en su mayoría por acero al carbono con una gran capacidad de reciclaje, en cuanto a su coste y a su rendimiento.

A la hora de realizar los ensayos será necesario usar protección visual y auditiva, al igual que mantener una distancia de seguridad cuando el conjunto se encuentre en movimiento para evitar accidentes por atrapamientos al estar expuestos elementos en rotación.

3.7 Fabricación y montaje

El **proceso principal de fabricación** es el corte por láser de chapa calibrada de acero. Este proceso electro-térmico consta de un haz de luz monocromático, coherente, focalizado, no divergente, brillante y potente el cual, mediante control numérico, recorre una trayectoria definida fundiendo y evaporando el material donde incide.

Mediante esta tecnología se permite conseguir piezas de geometría compleja de forma rápida y suficientemente precisa. Sin embargo, este proceso produce una zona de afectación térmica junto a un acabado rugoso en la superficie.

La realización del resto de piezas será mediante procesos tradicionales de fabricación. El alojamiento de los rodamientos será torneado, al igual que los casquillos de aluminio extruido, parar lograr la longitud especificada. Las superficies roscadas se conseguirán mediante machos y terrajas.

A razón de los estudios realizados de esfuerzos mecánicos que sufre el conjunto, observando que son reducidos, se fija que el **material** con el que se fabricarán las piezas es **F111**, debido a que existe chapa calibrada de este material y es el más económico (con excepción de los casquillos de aluminio y los elementos normalizados).

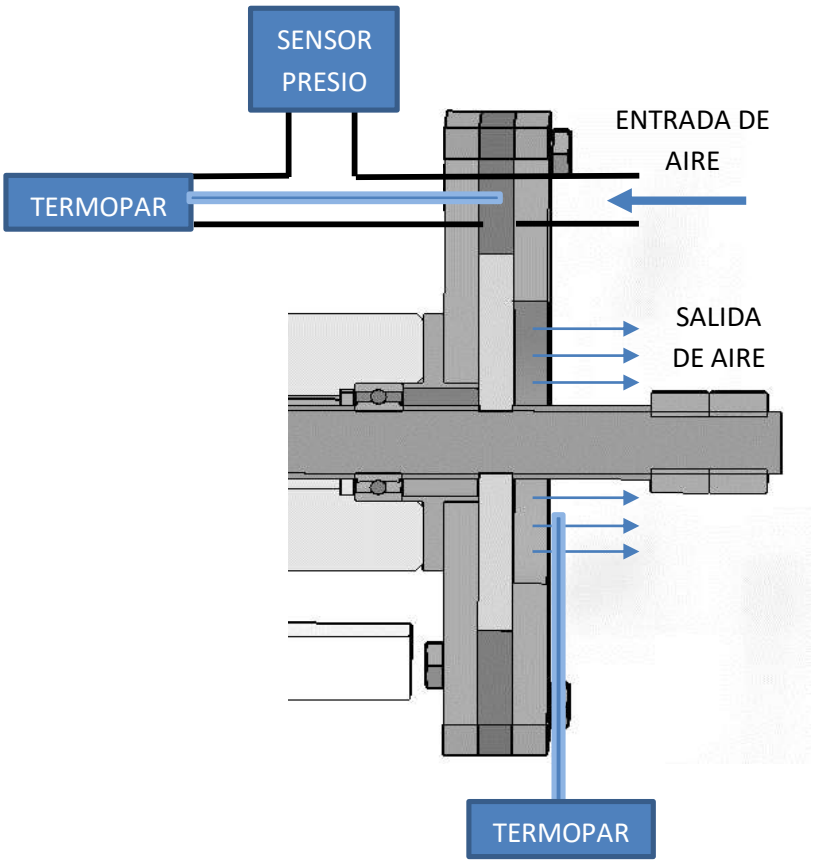


Figura 18. Esquema de situación de los sensores.

El **montaje** será intuitivo, siguiendo el plano de despiece del *Anexo I,* teniendo en cuenta las consideraciones de par de apriete de los elementos roscados. En el caso de los tornillos M4 el par de apriete será de $3{,}5\ Nm$ y en el caso de las tuercas que fijan el conjunto al eje será de $8{,}06\ Nm$, como se ha calculado previamente.

Este esquema será llevado a la práctica con los elementos de fontanería necesarios, ensamblando las distintas partes siguiendo las instrucciones de instalación del fabricante.

Es necesario que la ubicación de los sensores sea la correcta para que las hipótesis y los desarrollos teóricos sean precisos, por ello, se posicionarán los sensores de presión y temperatura de manera que la medida sea aproximadamente la de remanso.

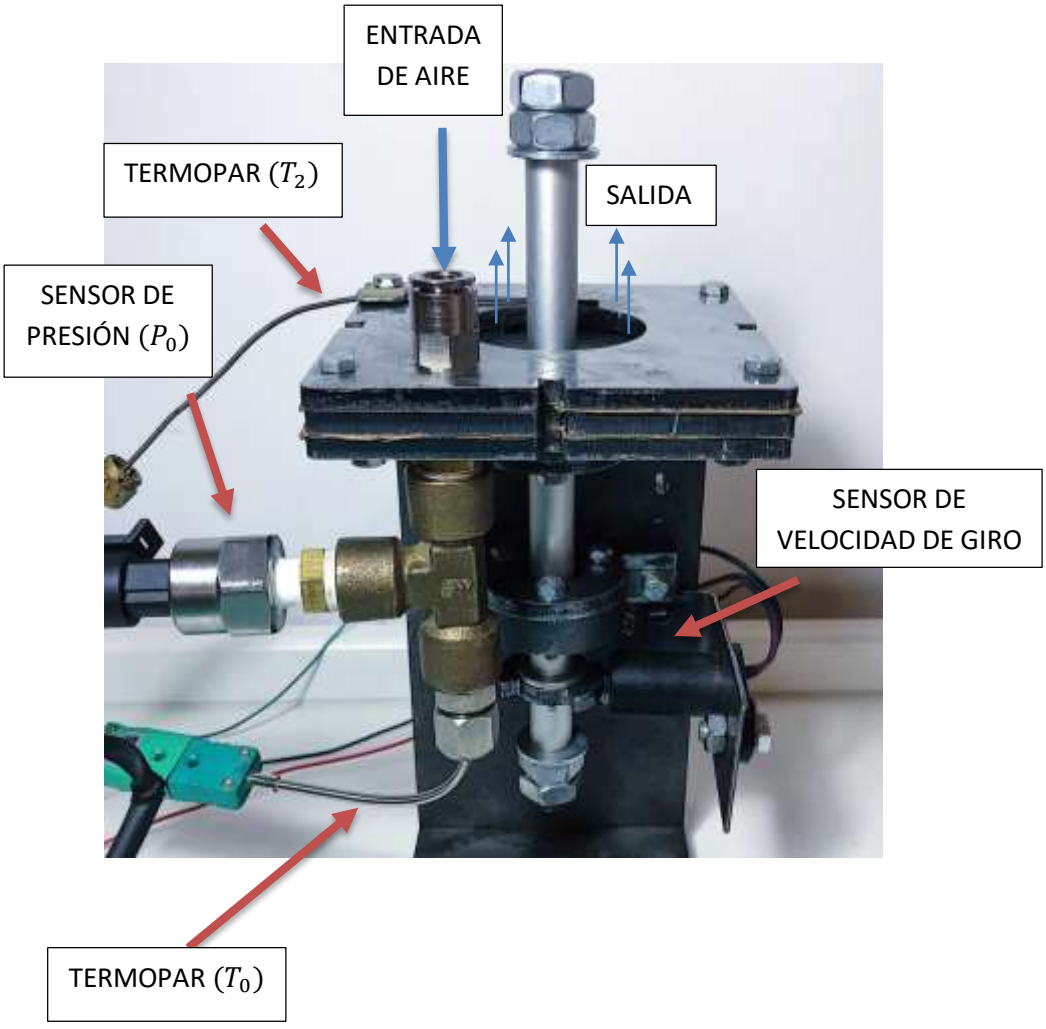


Figura 20. Montaje del prototipo.

Capítulo 4
Diseño y montaje eléctrico.

4.1 Diseño preliminar

Una vez identificadas las variables que determinan el funcionamiento de las turbomáquinas térmicas, se procede a monitorizarlas, siendo estas la presión a la entrada (P_0), temperatura a la entrada (T_0), velocidad angular (ω), temperatura a la salida (T_2,) y el tiempo (t).

En la *Figura 21* se puede observar una representación simplificada de la situación de los sensores mostrada en la *Figura 20*:

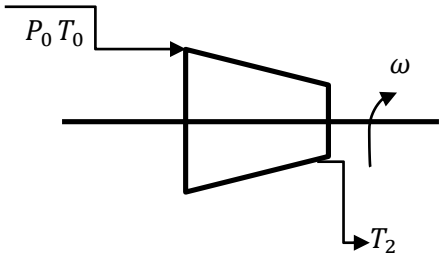


Figura 21. Representación de situación de los sensores.

Los siguientes apartados se detienen en la elección de estos sensores, su funcionamiento básico y el montaje que hace posible que el controlador sea capaz de recoger su señal para posteriormente trabajar con ella.

4.2 Elementos electrónicos utilizados.

4.2.1 Sensor de velocidad de giro.

Existen múltiples tipos de sensores cuya finalidad es determinar la velocidad de giro de un cuerpo basándose en las variaciones de una magnitud física dada (electromagnetismo, óptica, dinámicos, …), logrando una señal eléctrica de la cual se es capaz de deducir dicha variable. Los más usados industrialmente con este propósito son los que logran captar las variaciones de un campo magnético.

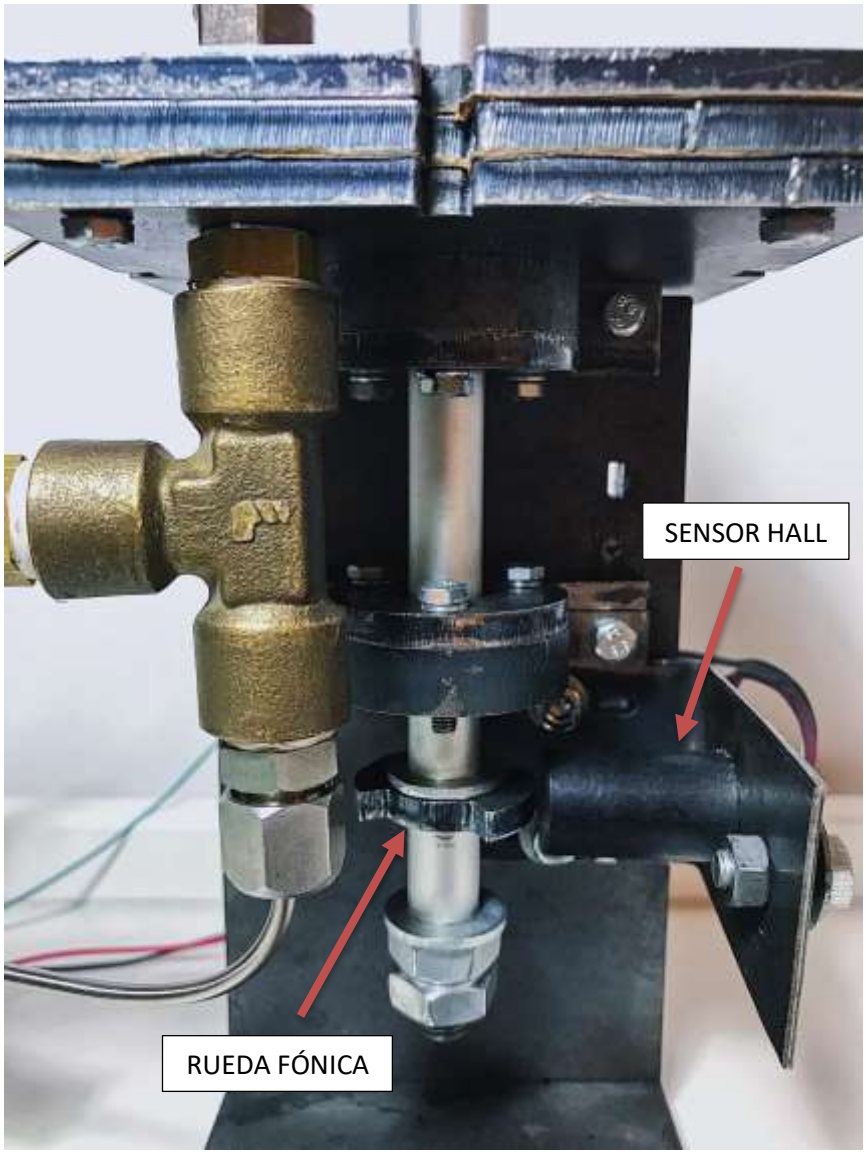


Figura 22. Vista de la rueda fónica y el sensor tipo Hall en el prototipo.

En este sentido, existen dos tipos principales de elementos que consiguen este efecto:

- *Sensor inductivo*: este tipo de sensores están compuestos por una bobina y un núcleo ferromagnético el cual, mediante la proximidad de otro material diamagnético, hace variar la inductancia de la bobina, provocando así una diferencia de potencial.
  Este tipo de sensores, cuando trabajan con una rueda fónica, producen una señal analógica sinusoidal de amplitud variable.

- *Sensor tipo Hall*: en este caso, se aplica una corriente longitudinal sobre un semiconductor, el cual, al encontrarse en las inmediaciones de un campo magnético, varía su diferencia de potencial de manera transversal a la corriente principal.

El dentado de la rueda fónica haría cambiar el campo magnético y produciría este efecto. La señal que devuelve el sensor es digital.

Se opta por usar el **sensor tipo Hall**, debido a que su salida digital facilita su tratamiento por el controlador. Se debe tener en cuenta a la hora de realizar el código que la sensibilidad del sensor dependerá de la distancia a la rueda fónica.

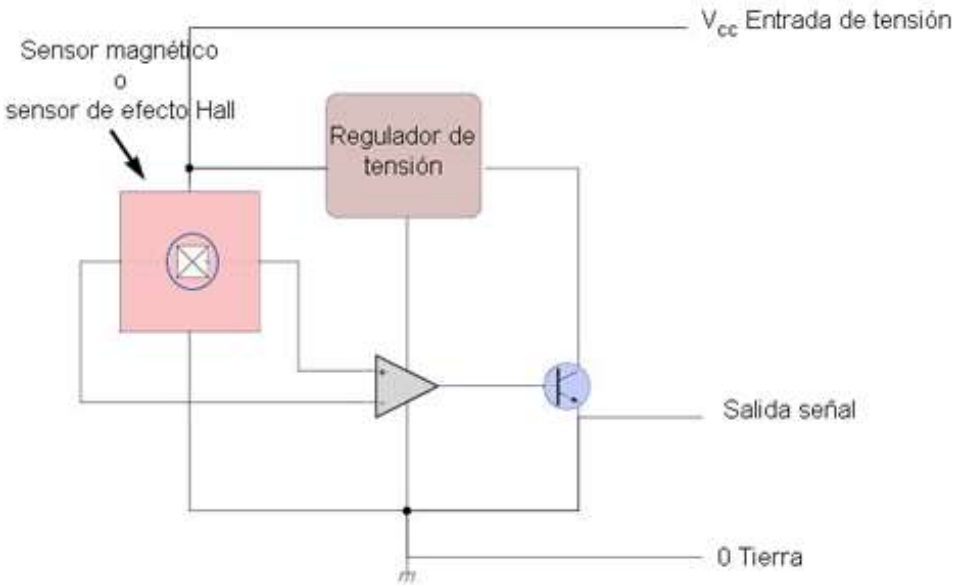


Figura 23. Esquema de funcionamiento del sensor Hall.

(tomado de: infotec.net)

Las conexiones son 3: positivo entre 5 - 12V de alimentación, negativo y señal. La implementación será detallada en el montaje. Es necesario **filtrar la señal** con un condensador de poca capacidad conectado a la señal y a tierra para eliminar errores en la señal como se muestra en los ensayos. La colocación de un condensador de elevada capacidad podría eliminar la señal en exceso a elevadas velocidades.

4.2.2 Sensor de temperatura.

Consiguen mediante variaciones de temperatura una variación eléctrica aprovechada para el cálculo de la temperatura. Debido a su coste, se baraja la implementación de dos tipos:

- *Termistor*: basa su funcionamiento en la variación de resistencia eléctrica de un determinado elemento frente a la variación de temperatura. Su coste es reducido y de aplicación sencilla. Su rango de temperaturas de aplicación es limitado. Su respuesta no es lineal.

- *Termopar*: consiste en la unión de dos metales distintos los cuales desarrollan una fuerza electromotriz debida a la diferencia de potencial electroquímico de los elementos. Este potencial varía con la temperatura. Este tipo de sensor tiene un amplio rango de operación, su tamaño es muy reducido y su precisión es de ± 1ºC. Tiene una respuesta no lineal a la variación de temperatura. Como inconveniente, necesitan un amplificador de señal.

Se opta por el **termopar**, debido a que su fabricación permite un tamaño más reducido y, por lo tanto, una mayor velocidad de lectura ante el cambio brusco de temperatura.

Existen en el mercado distintos tipos de termopares en función de la aplicación y el los metales por los cuales se componen. Debido a su extenso uso y bajo coste, se opta por un **termopar tipo K**.

A la hora de la implementación es necesario un amplificador para captar la variación de voltaje de la unión, linealice y realice la compensación de unión fría. Se escoge un módulo digital (**MAX31855**) que traduce la lectura del termopar a un número binario que ingresará al controlador y este interpretará.

Del termopar se toman dos cables (positivo y negativo) que se conectan al módulo MAX31855 el cual tiene alimentación (3,3 V), negativo, reloj, salida de datos (3V) y pin de activación.

4.2.3 Sensor de presión.

Se pueden obtener mediante diferentes fenómenos la relación entre **presión** y potencial eléctrico, no obstante, todos los métodos son adecuados para las condiciones del fluido y del conjunto, ya que las temperaturas son moderadas y el fluido es aire; a excepción de los dispositivos de membrana los cuales podrían producir un error de medida debido a las vibraciones.

La medida de presión puede ser diferencial, relativa o absoluta. Para esta aplicación será necesaria la medida de **presión relativa**, pudiendo cambiar de tipo modificando el programa.

La respuesta de este sensor será **lineal y analógica**, factores a tener en cuenta a la hora de elaborar el software y la conexión con el controlador. Posee 3 conexiones: positivo (5V), negativo y salida.

Debido a que la señal será en un rango de 0-5 V y el controlador opera con un máximo de 3,3V, se debe realizar un divisor de tensión que disminuya el voltaje de la señal.

4.3 Controlador

4.3.1 Introducción

El **controlador** seleccionado por su relación prestaciones/precio es **ESP32**. Es una plataforma electrónica que puede ser programada mediante el interfaz de Arduino, la cual está basada en hardware y software libre. Esta plataforma permite crear diferentes tipos de programas en una sola placa a los que la se le pueden dar diferentes tipos de uso.

Por otra parte, su precio es bajo, es multiplataforma (Windows, Mac, Linux…), posee un entorno de programación suficientemente sencillo y un leguaje extendido como es C o C++, siendo compatible mediante interfaces con otros lenguajes.

Ciñéndose a realizar una breve introducción del elemento, delimitándonos a las características que serán útiles para este proyecto, las partes de las que consta son las siguientes:

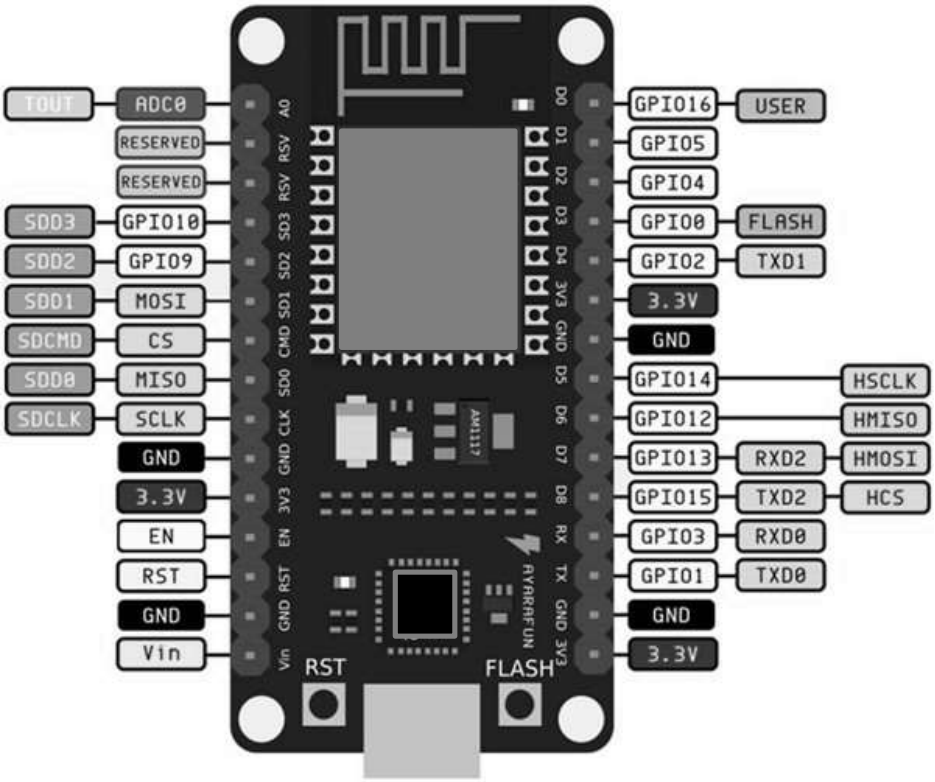


Figura 24. Esquema de los pines de ESP32.

Resumiendo las **características principales**, el controlador puede ser alimentado mediante una fuente externa o por el puerto serial conectado al USB de un equipo informático. Es capaz, mediante las salidas de 3.3V y 5V cuando está conectado al equipo, de alimentar los sensores necesarios para la aplicación que se propone.

Este dispositivo posee **entradas y salidas** de distintos tipos con distintas propiedades. Su principal diferencia reside en que unas son diseñadas para señales analógicas y otras para digitales. Como se ha expuesto anteriormente, el sensor de presión devuelve una señal analógica, mientras que el sensor hall y los termopares producen una señal digital, por lo tanto, se deberá implementar de manera correcta dichas entradas.

El procesador trabaja con las entradas mencionadas en función del programa instalado en la memoria del dispositivo. En esta aplicación en concreto, el ESP32 se comunicará con el equipo mediante el USB y transmitirá los datos de las variables.

En el siguiente punto se expondrá detalladamente la función que desarrolla este componente, como se implementa en el conjunto y la manera de proceder del programa grabado en el mismo.

A tener en cuenta, dependiendo del modelo, existen unas **limitaciones del hardware** que acotaran la capacidad de recopilación de datos:

- *Velocidad puerto serie*: Una de ellas es la velocidad con la que es capaz de comunicar información por el puerto serie USB, dependiente de la longitud del cable que comunica el puerto serie. Si es suficientemente corto es posible llegar a 2.000.000 de baudios.
- *Reloj de placa*: posee un reloj de velocidad modificable con un máximo de 240MHz.
- *Procesador*: este elemento posee un procesador de dos núcleos, por lo que pueden ejecutarse dos tareas simultáneamente.

Estas características harán posible el tratamiento de las señales con la suficiente velocidad como para no perder información del proceso. Mediante un ensayo posterior, se validarán las lecturas y la capacidad del conjunto de transmisión y gestión de datos.

4.3.2 Montaje y programa

La **finalidad del programa** cargado en el controlador será enviar los datos de los sensores pretratados a través el puerto USB al equipo para que este los gestione y calcule el resto de magnitudes que caracterizan la turbina.

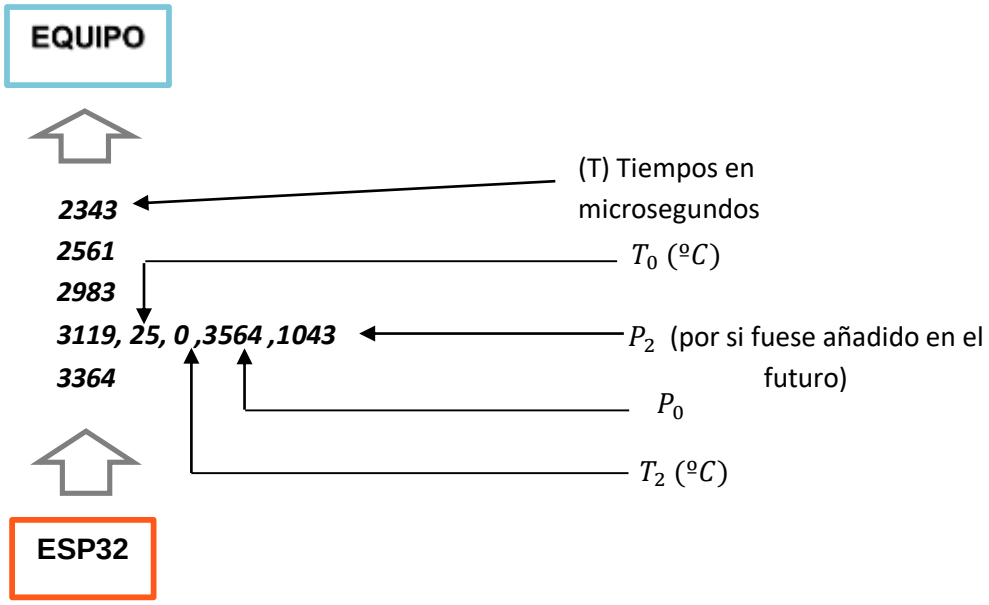


Figura 25. Diagrama de flujo explicativo del bucle principal del programa.

Para conseguir esta estructura de salida de datos es necesario que el controlador reproduzca una rutina como con un esquema similar al expuesto en la *Figura 26*.

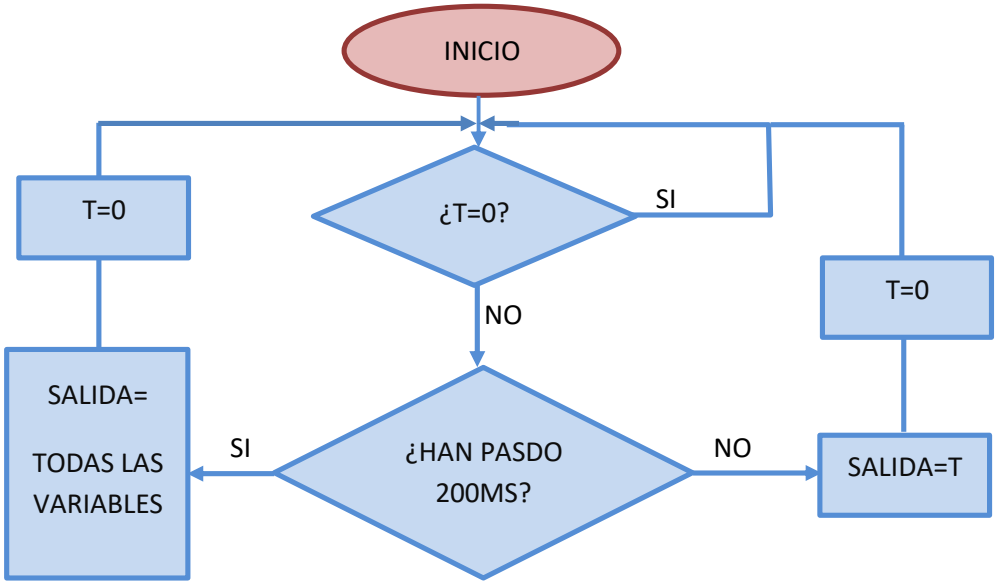


Figura 26. Diagrama de flujo explicativo del bucle principal del programa.

Mientras este esquema se ejecuta, se realizan en él las interrupciones cuando se posicione un diente de la rueda fónica frente al sensor Hall. Esto permite ejecutar una rutina a parte de la principal para fijar el tiempo en el cual se ha realizado el pulso, siguiendo el esquema que muestra la *Figura 27*.

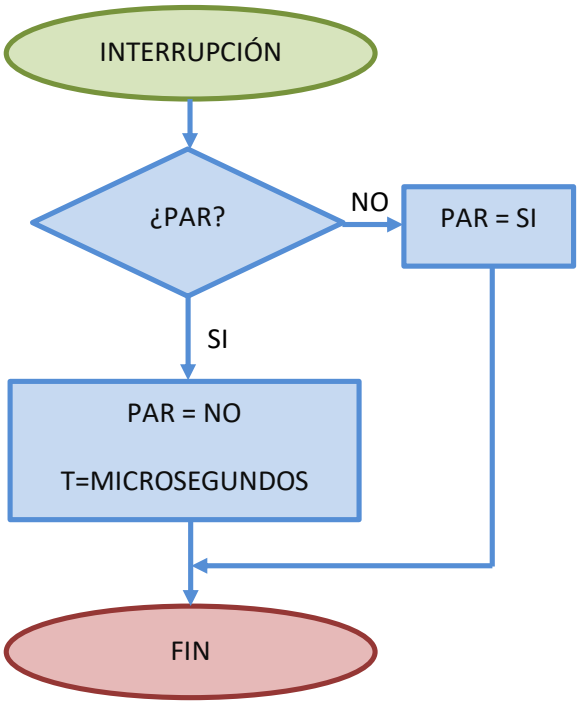


Figura 27. Diagrama de flujo de la rutina realizada durante las interrupciones.

Proveniente del sensor de **velocidad de giro**, el programa emitirá el tiempo en el cual encuentra el segundo flanco de subida de señal consecutivo, enviando así el tiempo en microsegundos en el que este sucede y pudiendo calcular la velocidad angular.

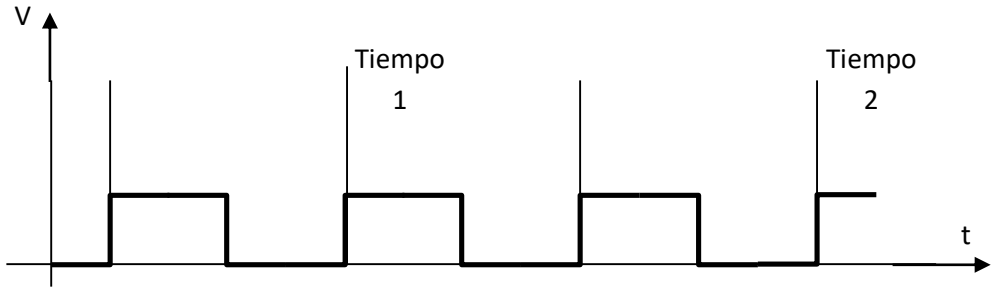


Figura 28. Señal del sensor Hall y donde efectúa toma de tiempos el controlador.

Puesto que la rueda fónica es de 2 dientes, la diferencia de tiempo entre cada medida será el periodo ($^{segundos}/_{vuelta}$)de la turbina. Se realiza de esta forma para evitar el error debido a la variación de ancho de pulso por el posicionamiento del sensor Hall.

Este procedimiento se realiza mediante interrupciones para evitar la pérdida de datos del proceso, aumentando así su precisión. Las interrupciones son herramientas que, al percibir una señal, paran el proceso y ejecutan una rutina.

El **resto de sensores**, cada cierto tiempo son muestreados para que devuelvan el valor de sus lecturas, pretratados por el controlador, mostrando el valor de la variable y no de su señal eléctrica. Estos son emitidos junto al siguiente valor de velocidad angular, enviando un vector de valores en ese caso.

En el caso de los **termopares**, se debe decodificar la señal que llega al pin, ya que llega una serie binaria generada por el módulo MAX31855.

En el caso del **sensor de presión**, el controlador traduce el voltaje que le llega a un número. Las entradas analógicas poseen una resolución de 12 bits, es decir, se asignan linealmente los valores entre 0 – 3,3V de la entrada a un número de 0 – 4095. Por tanto, en el software se deberá relacionar este digito con la presión mediante las especificaciones técnicas del fabricante.

Este procedimiento se lleva a cabo espaciado en el tiempo (200 ms) ya que la variación de estos valores no es muy acusada y podría saturar el buffer de salida.

El montaje es realizado siguiendo el esquema eléctrico. En la *Figura 29* se visualiza la realización del mismo para el prototipo.

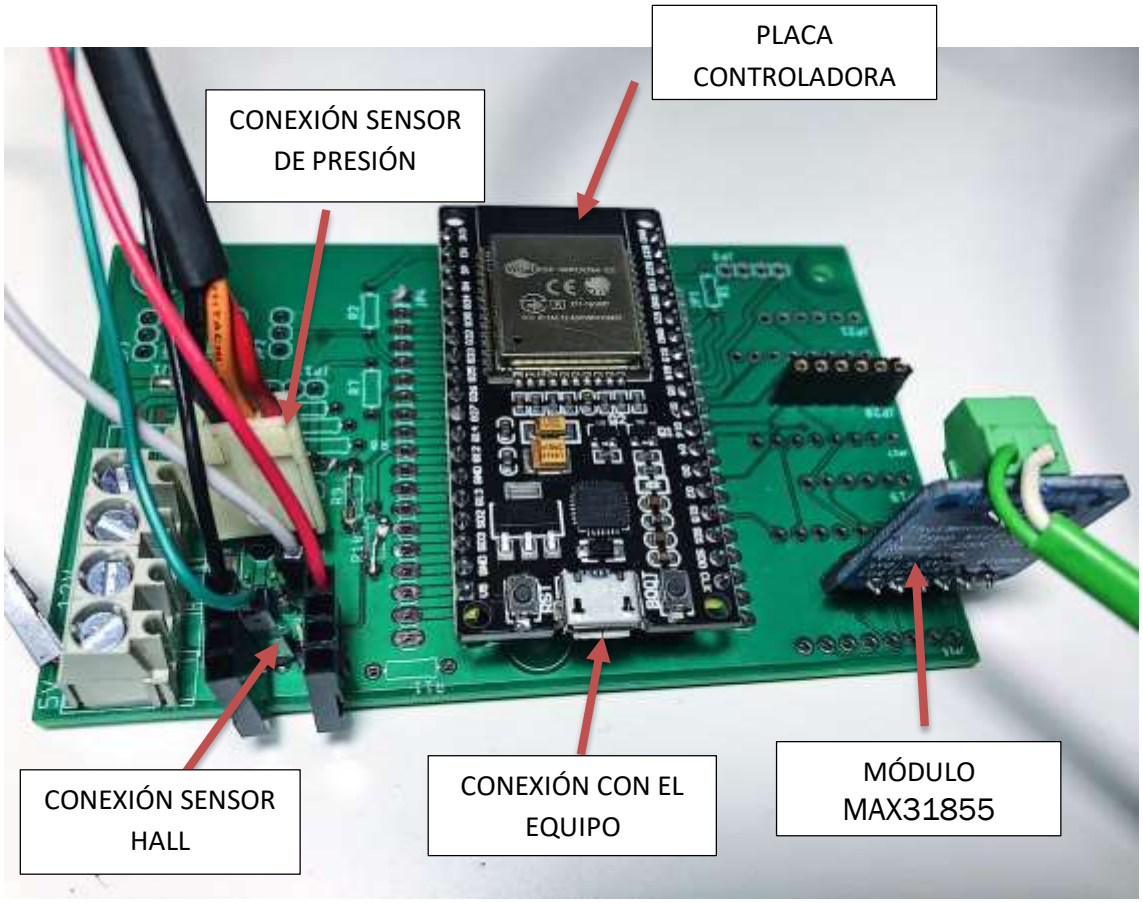


Figura 29. Conexiones del prototipo con el controlador.

Atendiendo a las consideraciones anteriores, el montaje de los sensores con los respectivos componentes auxiliares, así como la placa controladora ESP-WROOM-32, se ejecutaría siguiendo el esquema de la *Figura 30*.

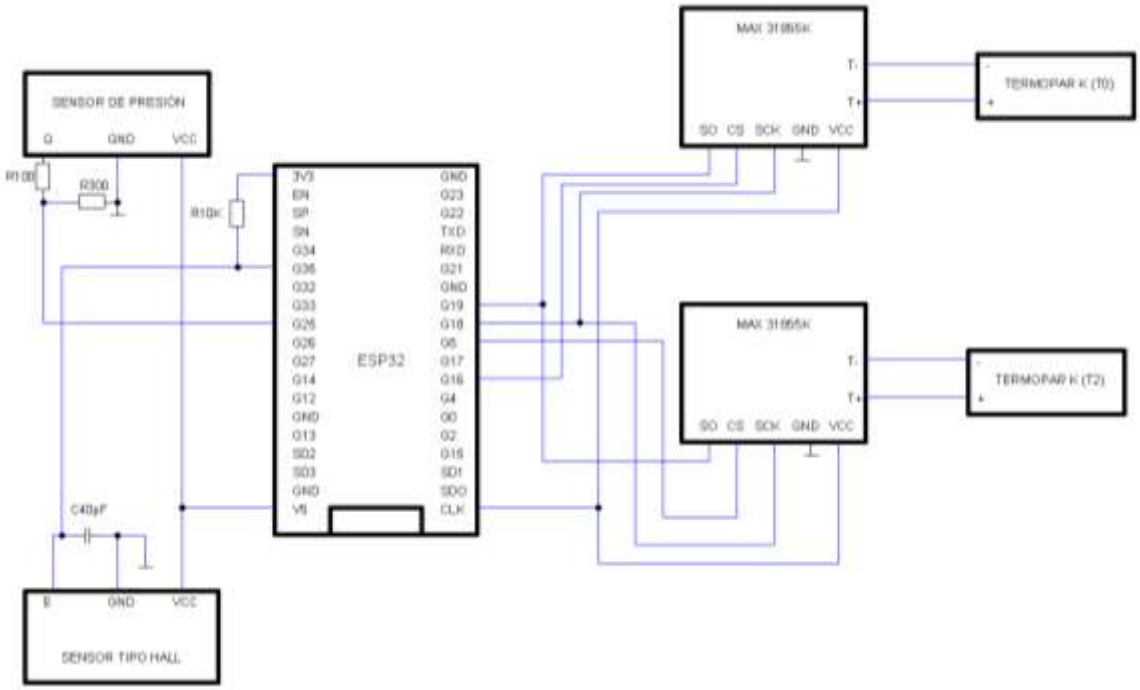


Figura 30. Esquema del montaje eléctrico.

Capítulo 5
Software, tratamiento de datos e interfaz.

5.1 Introducción

Con motivo de realizar el cálculo y posterior exposición de los datos del ensayo, se construye un **software de aplicación** que realice estas funciones de manera automática. Un software es un conjunto de herramientas lógicas para realizar tareas específicas configuradas a través de un código previamente construido.

El lenguaje de programación usado para crear la aplicación es Python, el cual está orientado a una sintaxis sencilla y legible. Sus principales características son: licencia de código abierto, lenguaje interpretado, dinámico, multiparadigma y multiplataforma.

Esta aplicación pretende comunicarse con el controlador para adquirir los datos que los sensores recojan y operar con ellos para, posteriormente, reproducirlos por el hardware en forma de interfaz gráfica a través de un monitor.

5.2 Tratamiento de datos

El tratamiento de datos es una parte esencial del proceso ya que de ella depende la correcta interpretación final de las variables que caracterizan el conjunto. Con motivo de ilustrar el proceso fundamental del software se plantea el esquema de la *Figura 31*.

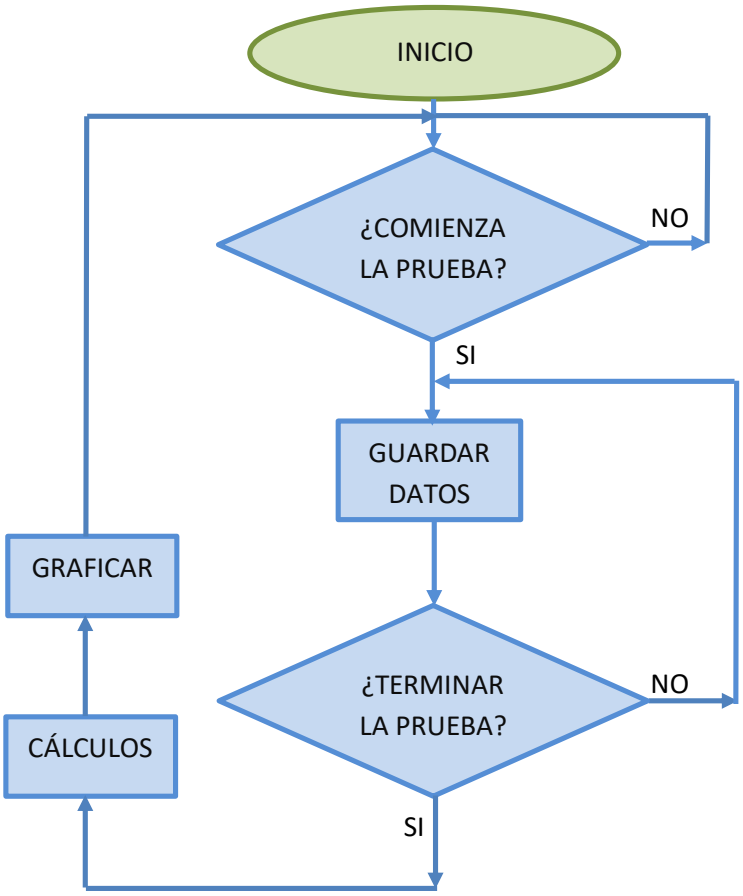


Figura 31. Esquema de funcionamiento del software.

En el momento en el cual se comienza la prueba, se recogen y guardan los datos llegados desde el puerto serie en una variable. Esto se realiza paralelamente al programa principal mediante *threading*, para evitar el bloqueo del interfaz al igual que la perdida de datos. Una vez el ensayo a concluido, se procede al tratamiento de los mismos.

En primer lugar, se calcula la **velocidad angular** (ω_j) y se fija el tiempo (t_j) en el cual esta ocurre a través del tiempo que transmite el controlador en un punto (t_i) y su inmediatamente posterior $((t_{i+1})$. El cálculo realizado será el siguiente:

$$t_j = ((t_{i+1} + t_i)/2) \cdot 10^{-6} \;\; ; \;\; \omega_j = \frac{1}{(t_{i+1} - t_i) \cdot 6 \cdot 10^{-6}} \;\; (rpm)$$

Se genera una **matriz de datos** a partir de interpolación lineal los valores de los sensores en los puntos donde no se tiene medida. En este punto, la señal del sensor de presión es convertida a presión (Pa) mediante las especificaciones técnicas del fabricante y las temperaturas se cambian sus unidades a $Kelvin$.

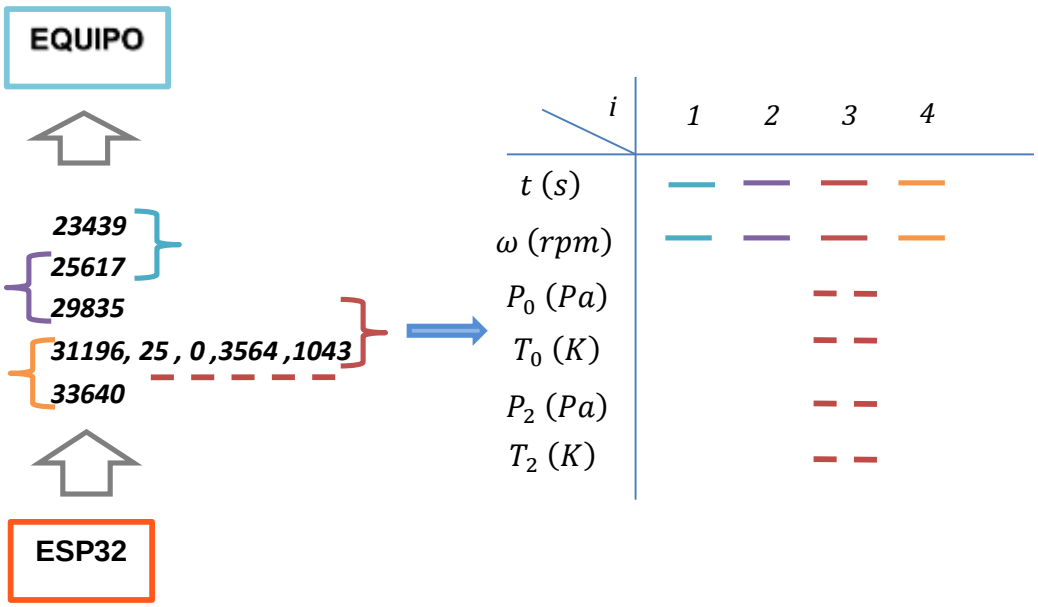


Figura 32.Generación de la matriz de datos a través de lo recibido por el puerto serie.

Una vez la matriz tiene estos datos, se procede al cálculo de la **aceleración angular**, para ello se realiza una interpolación lineal cuya pendiente será el valor de la aceleración en un punto. Para ello se escogerán 5 puntos previos y 5 posteriores.

Como se puede comprobar en la *Figura 33*, si se optase por el cálculo de la aceleración como la pendiente de la recta que une los datos anteriores y posteriores al punto *i*, el cálculo sería erróneo debido a las características de la curva.

Esta aproximación produce un error, debido a que, si se escogiesen muchos puntos antes y después de *i*, se perdería información al suavizar demasiado la curva, y no hacerlo induciría error en los cálculos de las magnitudes que dependen de esta.

$$\alpha_i = \frac{nºpuntos \cdot \sum_{j=i-nº/2}^{j=i+nº/2}(\omega_j \cdot t_j) - \sum_{j=i-nº/2}^{j=i+nº/2}(\omega_j) \cdot \sum_{j=i-nº/2}^{j=i+nº/2}(t_j)}{nºpuntos \cdot \sum_{j=i-nº/2}^{j=i+nº/2}({t_j}^2) - \left(\sum_{j=i-nº/2}^{j=i+nº/2}(t_j)\right)^2}$$

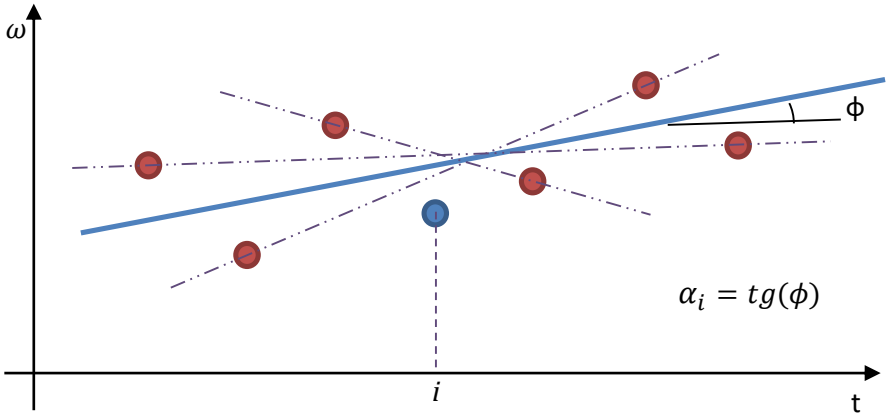


Figura 33. Ejemplo grafico de la media de pendientes para aproximar la aceleración.

A la hora de realizar los cálculos de **potencia efectiva, potencia de pérdidas y rendimiento isoentrópico** se discriminará entre aceleración positiva y negativa. Al multiplicar la velocidad angular por la aceleración angular positiva y el momento de inercia del conjunto (introducido por el usuario) se halla la potencia efectiva como ya se dedujo en el desarrollo teórico. Las pérdidas se calculan de la misma forma, siendo la aceleración angular negativa.

Se **ordenan** las filas de la matriz de menor a mayor velocidad angular y se añade en las columnas contiguas los valores de $\omega\, ordenada, P_e(\omega),\ P_p(\omega)\ y\ \eta_{iso}(\omega)$. Este paso es necesario para posibilitar el cálculo de magnitudes producto de variables calculadas en la aceleración y deceleración del conjunto.

Con los datos ordenados por orden de velocidad angular, se dispone a interpolar para hallar los datos de los cuales no se tiene conocimiento, ya que durante el proceso se aceleración y deceleración las medidas de velocidad angular son distintas y no tienen por qué coincidir. Tras este paso, se podrá operar con dichas magnitudes.

En última instancia, previa al reproducir los resultados por pantalla, se calculan las últimas magnitudes: $P_i(\omega)$, $M_i(\omega)$, $\eta_{mec}(\omega)$ y $\eta_{total}(\omega)$.

5.3 Interfaz

A través de la **interfaz** través el usuario controlará el software y se mostraran los datos. Esta parte tendrá dos botones de comienzo de prueba y paro de la misma. También poseerá varios marcos a través de los cuales es posible introducir datos o elegir entre las distintas opciones de gráfica.

Al pulsar el botón inicio, el equipo arrancará la rutina descrita de almacenamiento de datos. Cuando el botón de parar es accionado, el programa realizará los cálculos necesarios para conseguir las magnitudes requeridas.

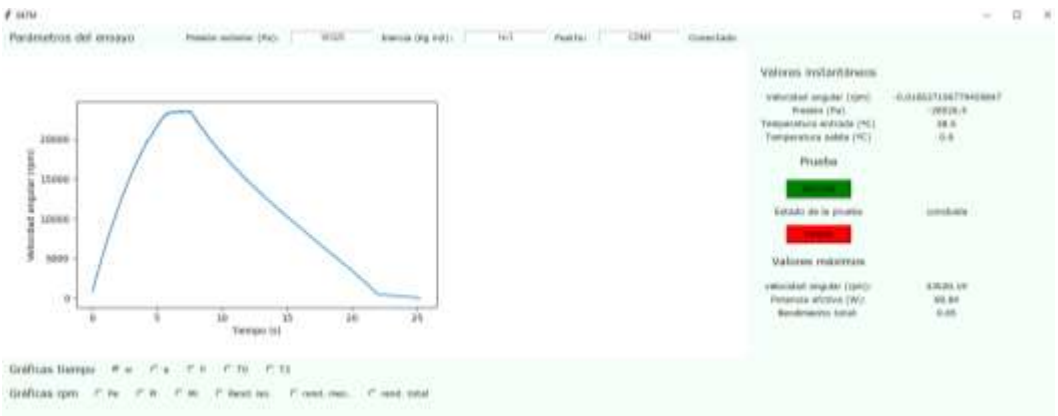


Figura 34. Interfaz gráfica.

En el marco destinado a la elección de las magnitudes a graficar, se selecciona que variables mostrar en la pantalla principal una vez el ensayo y los cálculos han concluido.

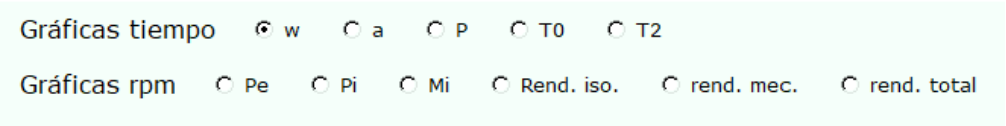


Figura 35. Marco destinado a la elección de gráfico.

En el marco de parámetros el usuario modificará las variables que son constantes en el ensayo, como se concretó en el estudio teórico. También se puede modificar el puerto en el cual se conecta el controlador, mostrando si este se encuentra conectado o no.

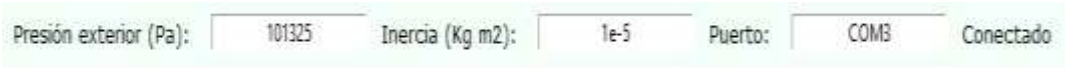


Figura 36. Marco destinado a fijar los parámetros del ensayo

Por último, en el marco de la información de la prueba se puede iniciar y parar la prueba, al igual que visualizar datos instantáneos cuando no se está ejecutando la prueba. También posee un apartado en el cual se muestran los valores máximos de velocidad angular, potencia efectiva y rendimiento total.

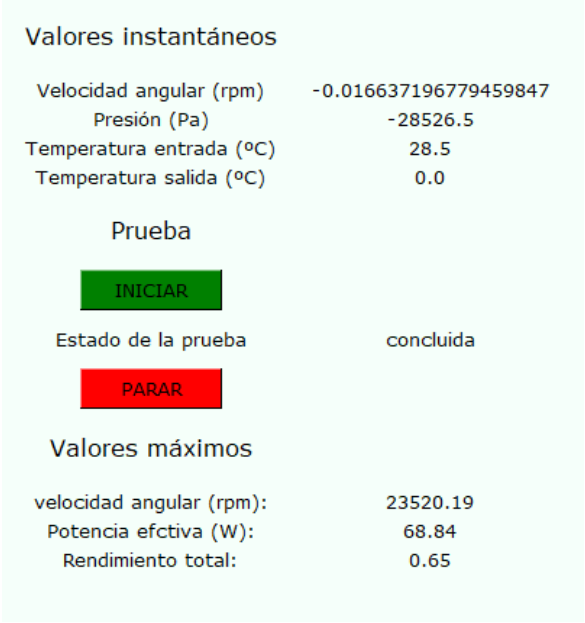


Figura 37. Marco de información de la prueba.

Capítulo 6 Estudio económico

6.1 Costes de ingeniería

Equipo informático					
Software	Coste unitario (€)	Unidades	Tiempo de amortización	Tiempo de uso	Coste final (€)
Microsoft Office	412,8	1	12 meses	3 meses	103,2
Catia	6.000	1	48 meses	3 meses	375
Hardware	Coste unitario	unidades	Tiempo de amortización	Tiempo de uso	Coste final (€)
PC	700	1	36 meses	3 meses	58,4
TOTAL					536,6

Recursos humanos			
Tarea	Horas	€/hora	Coste final (€)
Estudios previos	25	30	750
Diseño mecánico	80	50	4000
Diseño eléctrico	35	30	1050
Diseño de software	80	50	4000
Montaje	2	30	60
Verificación	16	30	480
TOTAL			10340

Coste total ingeniería	€
Equipo informático	536,6
Recursos humanos	10340
Coste bruto	10876,6

El coste de ingeniería asciende a DIEZ MIL OCHOCIENTOS SETENTA Y SEIS EUROS con SESENTA CÉNTIMOS.

6.2 Costes materiales mecánicos

Componentes mecánicos					
Componente	Referencia	Distribuidor	Cantidad	Precio unitario (€)	Precio total (€)
Rueda fónica	01-E5 U1	CMO	1	1,2	1,2
Rotor	05-E5 U1	CMO	1	1,2	1,2
Tapa superior	06-E5 U1	CMO	1	4,5	4,5
Estátor	07-E5 U1	CMO	1	4,5	4,5
Tapa inferior	08-E5 U1	CMO	1	4,5	4,5
Bastidor	09-E10 U1	CMO	1	3,6	3,6
Tornillo cabeza avellanada M4x16	ISO 10642	BRICOMART	3	0,03	0,09
Tornillo cabeza hexagonal M4x8	ISO 4017	BRICOMART	8	0,03	0,24
Tornillo cabeza hexagonal M4x12	ISO 4017	BRICOMART	2	0,03	0,06
Tornillo cabeza hexagonal M4x20	ISO 4017	BRICOMART	1	0,03	0,03
Tuerca M4	ISO 4032	BRICOMART	1	0,03	0,03
Tuerca M10	ISO 4032	BRICOMART	4	0,03	0,12
Rodamiento	W_63800	123RODAMIENTO	2	3,8	7,6
Tubo extruido aluminio	-	BRICOMART	1	1,5	1,5
Papel de junta	-	MMG	1	4,32	4,32
Racor neumático 1/8”	-	RS	3	1,18	3,44
Racor neumático T 1/8”	-	RS	1	3,2	3,2
Varilla calibrada	-	HTA3D	1	2,3	2,3
Cilindro 50mm F111	-	RANDRADE	1	29,12	29,12
TOTAL					71,55

El coste de materiales asciende a SETENTA Y UN EUROS con CINCUENTA Y CINCO CÉNTIMOS.

6.3 Costes de fabricación

Costes de fabricación					
Fase	Material	Operación	Cantidad	Precio unitario	Total
Torneado alojamiento rodamientos	F111	torneado	2 hora de trabajo	55€/hora	110€
Roscado	-	Roscado de piezas	1 hora de trabajo	25€/hora	25€
TOTAL					135

El coste de fabricación asciende a CIENTO TREINTA Y CINCO EUROS.

6.4 Costes eléctricos

Componentes electrónicos					
Componente	Referencia	Distribuidor	Unidades	Precio unitario (€)	Coste (€)
Sensor de presión	RS 3987039	RS	1	35,85	35,85
Termopar tipo K	-	RS	2	8,6	17,2
ESP32	-	BRICOGEEK	1	6,5	6,5
Sensor Hall	BOSCH - 0 281 002 474	OSCARO	1	32,31	32,31
MAX 31855	134-6476	RS	2	16,88	33,76
Cables y resistencias	-	RS	-	1	1
TOTAL					126,62

Recursos humanos			
Tarea	Horas	€/hora	Coste final
Montaje	2	25	50
TOTAL			50

6.5 Coste de ejecución

Coste de ingeniería	10876,6
Coste materiales mecánicos	71,55
Coste de fabricación	135
Costes eléctricos	176,62
TOTAL BRUTO	11259,77

Capítulo 8 Ensayos y resultados

Para llegar a realizar los ensayos, se generará un prototipo que sirva de modelo para estos efectos. Será fabricado y ensamblado según los criterios que se desarrollan a lo largo del trabajo, sin embargo, con las modificaciones necesarias para hacerlo posible en el contexto de falta de recursos en el que es fabricado, fuera de las instalaciones de la Escuela de Ingenierías Industriales.

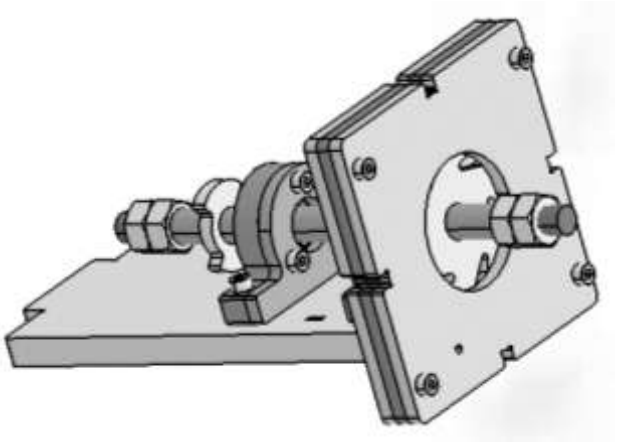


Figura 38. Prototipo.

Los asientos de los rodamientos se realizarán mediante corte láser y no torneados. Los casquillos serán ajustados sin el uso de torno, lo cual reducirá su precisión. Por todo esto, se ve necesario implementar unas juntas más gruesas que permitan que gire libremente el rotor sin interferir con las tapas, aumentando presumiblemente las pérdidas de presión del fluido.

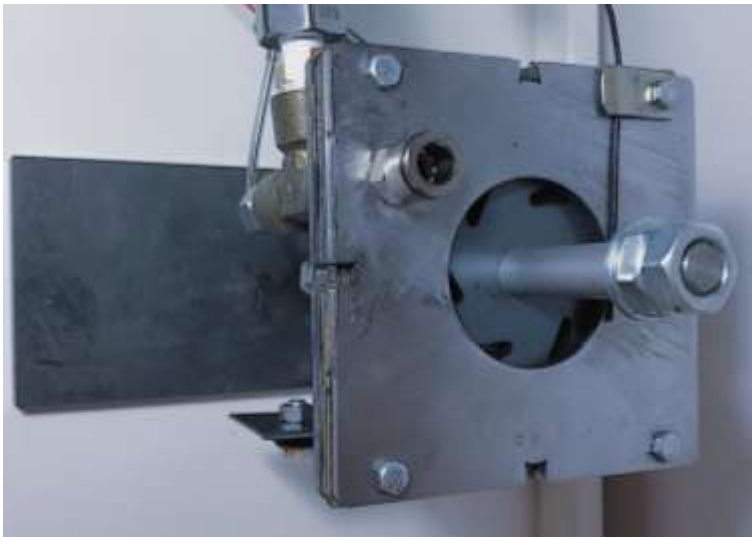


Figura 39. Vista superior del prototipo.

Asumiendo las limitaciones del prototipo y los factores diferenciales con el modelo original, se realizan dos ensayos: uno mecánico y otro del conjunto.

8.1 Ensayo mecánico.

Se predispone que, ante la incertidumbre de la correcta realización del diseño geométrico en todos sus aspectos, se fija este ensayo con el objetivo de evaluar su correcta definición y la adecuación de los procesos de fabricación.

Se analiza el conjunto en todo su proceso de ejecución, intentando identificar posibles incorreciones:

- *Errores de ajustes*: detección de posibles errores de cálculo o factores desestimados durante el proceso de diseño.
- *Dificultad de fabricación o montaje*: aproximación a la realización del modelo definitivo bajo la seguridad de cumplir los criterios constructivos.
- *Capacidad de giro mediante alimentación de aire*: con el fin de confirmar o refutar la correcta dimensión entre rotor y tapas; así como la geometría básica del escalonamiento.

Los **resultados** que se obtienen del **ensayo mecánico** del prototipo son:

- Los rodamientos quedan desalineados debido a la fabricación de los asientos, esto hace que las pérdidas mecánicas incrementen y la velocidad angular máxima se vea reducida.
- El ajuste mediante el casquillo situado entre los rodamientos no asegura el desplazamiento axial.
- El juego entre estátor y rotor es suficiente para permitir su funcionamiento.
- El montaje es factible.

De este ensayo se extrae que es necesario tornear el asiento de rodamientos para garantizar su alineación y concentricidad. A su vez, el movimiento axial debe ser reducido o impedido para evitar interferencias entre el rotor y las tapas.

Estos resultados entran dentro de los esperados y quedan recogidos en el diseño principal. Como se corroborará en el próximo ensayo, el conjunto es capaz de llegar bajo estas condiciones a **25.000 rpm,** esperando mayores velocidades con el diseño original al verse reducidas las pérdidas.

8.2 Prueba del conjunto.

Ante la imposibilidad de concretar la precisión de medida de los sensores y la correcta generación las magnitudes a partir de estas, se diseña un ensayo donde se esclarezca la precisión de medida y funcionamiento del conjunto.

Debido a la experiencia sobre la lectura de termopares y sensores de presión, así como las mayores limitaciones e incertidumbres se encuentran en la correcta

definición de la velocidad angular, esencial para el cálculo del resto de magnitudes; se decide **comprobar la respuesta** del conjunto sobre la **captación de la velocidad angular** únicamente.

Para ello se monitorizará la respuesta del sensor hall con un osciloscopio mientras se acciona el conjunto, al realizar la prueba con ambos equipos trabajando simultáneamente (el osciloscopio y el propio) se compararán las respuestas directamente.

Este ensayo será repetido 5 veces para comprobar la repetibilidad de los resultados y dar una cantidad de datos suficiente para que los datos obtenidos se consideren seguros.

Tras realizar los primeros ensayos se observan errores en la señal como se refleja en la *Figura 40* estos son debidos al ruido de la señal que el controlador malinterpreta como pulsos. Con el fin de eliminar dichos errores se conecta un filtro, realizándolo con un condensador de baja capacidad o una pequeña bobina para eliminar los posibles ruidos, rebotes y picos en la señal.

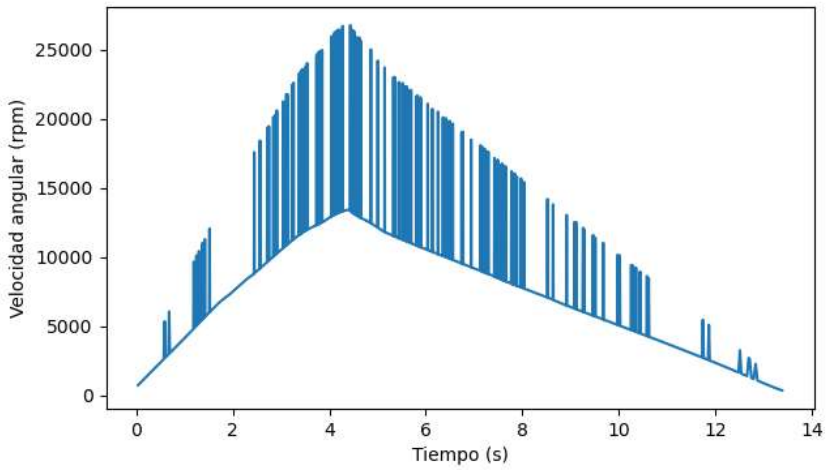


Figura 40. Gráfica obtenida mediante el software donde se aprecian errores debidos a la señal.

Tras aplicar el filtro se vuelven a realizar las pruebas, obteniendo la salida como la que se muestra en la *Figura 41*. En este caso se puede observar que, en ocasiones, siguen apareciendo errores, aunque menos numerosos.

Estos serán solventados mediante el software, que eliminará estos puntos realizando una interpolación entre el punto del error y el siguiente, estimando que se trata de un error cuando el valor de la velocidad angular supere en un 50% al valor anterior.

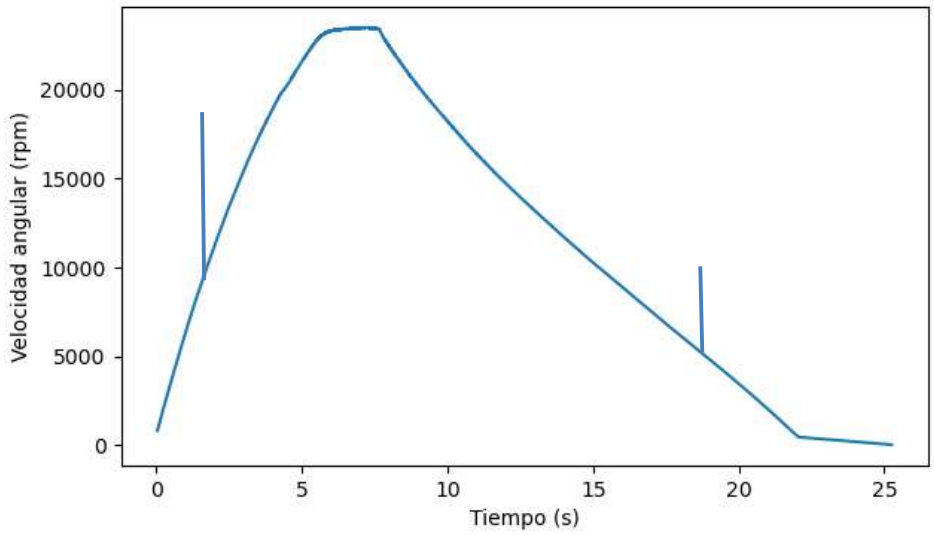


Figura 41. Gráfica realizada por el software donde se evidencia la existencia de errores.

Tras estas correcciones se vuelven a realizar los ensayos con los siguientes **resultados**:

- Para el régimen de giro ensayado las mediciones son correctas y precisas.
- Se evidencia la relación de ajustes del conjunto con la capacidad de realizar un ensayo correcto debido a la variación de perdidas mecánicas a lo largo de la prueba.

A continuación, se exponen las gráficas obtenidas por el osciloscopio y el programa con su comparación.

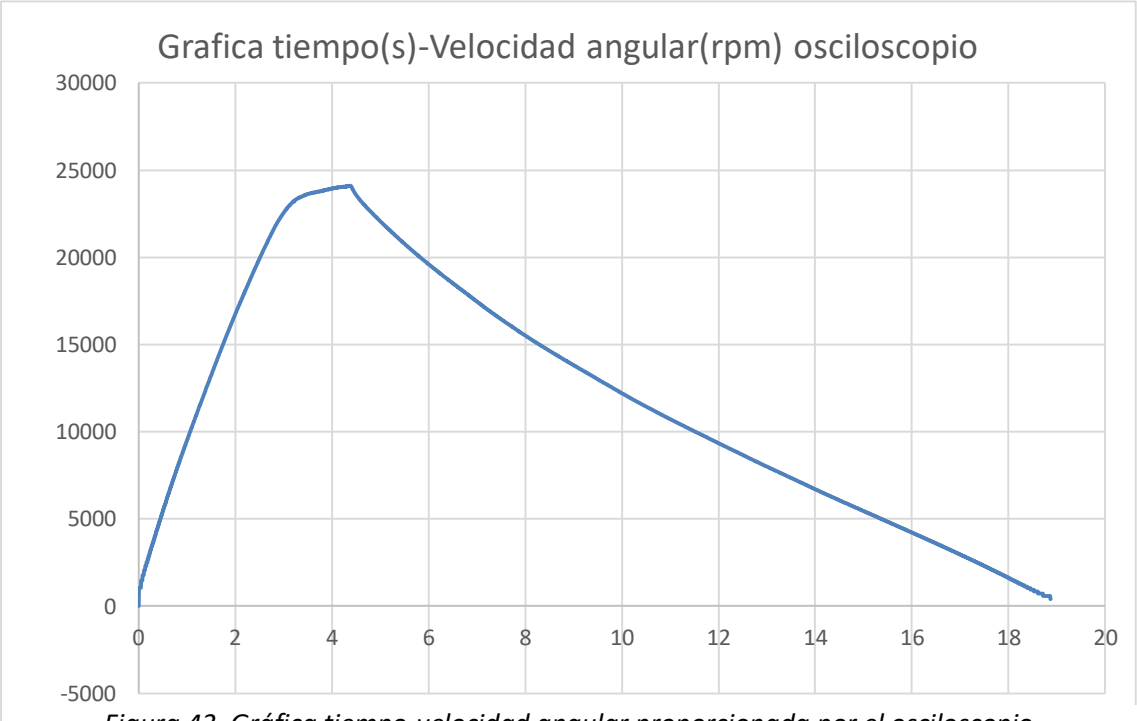


Figura 43. Gráfica tiempo-velocidad angular proporcionada por el osciloscopio.

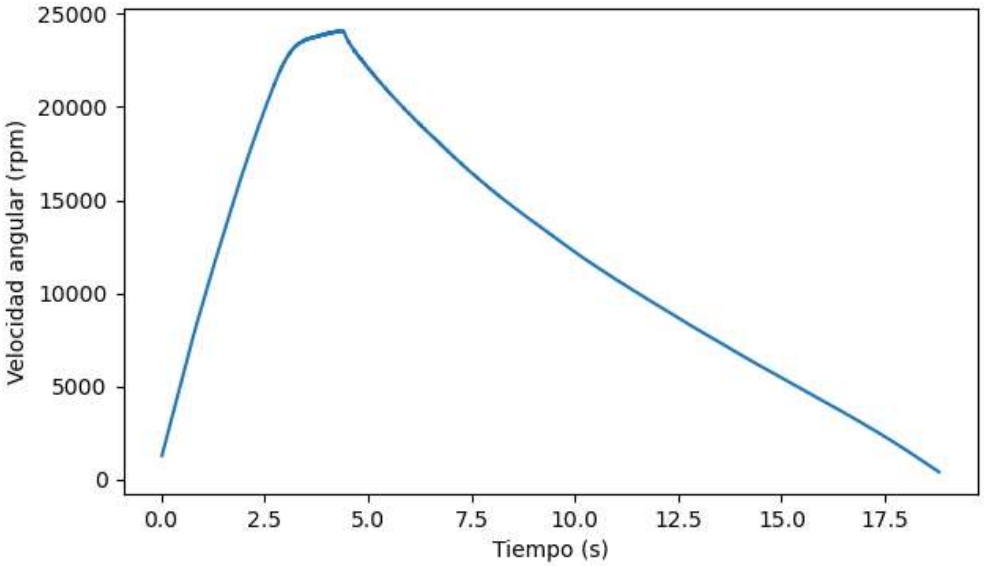


Figura 44. Gráfica obtenida a través del software.

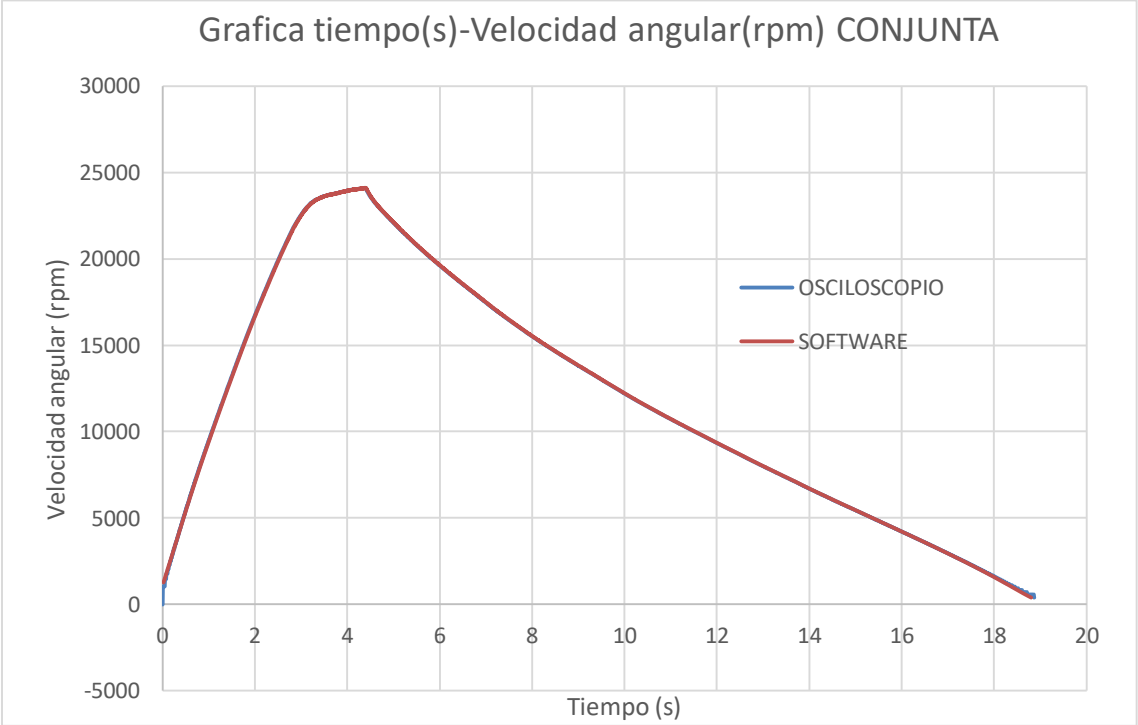


Figura 45. Grafica comparativa de los resultados obtenidos por el osciloscopio (azul) y el software (naranja).

Como se puede comprobar, ambas gráficas con semejantes por lo cual se puede afirmar que las lecturas que realiza el programa son suficientemente precisas. Con el fin de poder visualizar mejor las diferencias entre ambas salidas, se efectúa una gráfica comparativa en detalle del primer segundo de prueba reflejada en la *Figura 46*.

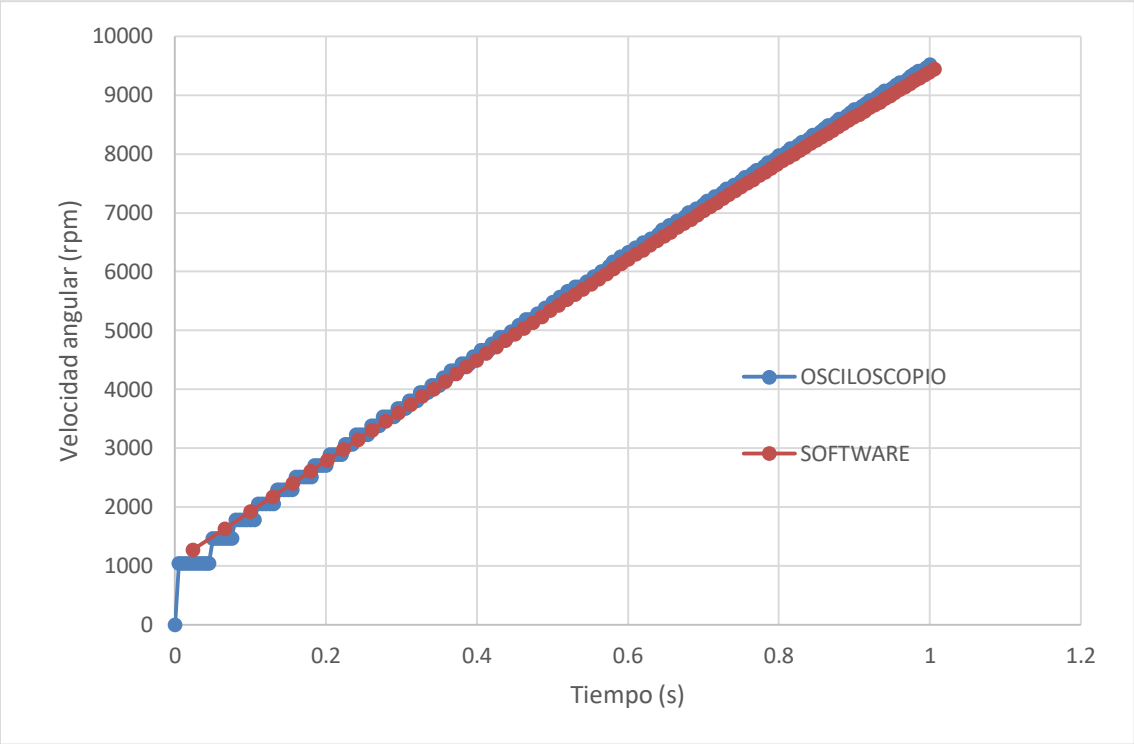
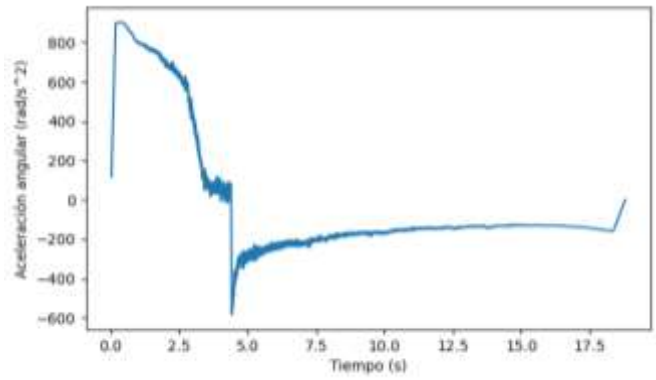
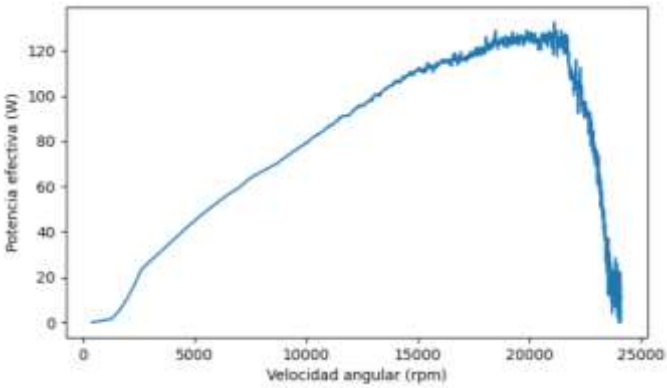


Figura 46. Grafica de detalle del primer segundo de prueba comparando la salida del osciloscopio (azul) y la del software (naranja).

En la *Figura 47* se expone de manera ilustrativa, algunas de las variables que devuelve el programa.



a)



b)

Figura 47. a) grafica aceleración-tiempo. b) gráfica potencia efectiva-velocidad angular.

Capítulo 9 Conclusiones

Las **conclusiones finales** del trabajo, evaluando los objetivos fijados inicialmente, son:

- *Definición mecánica, eléctrica y construcción*: Se ha **conseguido** realizar el diseño de un conjunto eléctrico y mecánico funcional, **materializándose** en forma de prototipo debido a las circunstancias del contexto en el cual ha sido ejecutado.
- *Implementación electrónica y de software:* Se ha **logrado** establecer una aplicación básica que realiza la impresión de las magnitudes de funcionamiento.
- *Realización de pruebas de verificación*: ha sido posible realizar dichos ensayos para dilucidar las **faltas y aciertos del proceso**, quedando plasmados en esta memoria.
- *Construcción de una herramienta útil:* siendo esta meta abstracta, se concibe el trabajo en su conjunto como una herramienta útil, completa y sencilla de ensayo de turbomáquinas.

Las posibles **líneas de futuro** que se plantean son las siguientes:

- Mejorar la apariencia del software, sugiriendo una interfaz más estética y funcional que la presente.
- Permitir comparaciones entre distintos ensayos y magnitudes por parte del programa.
- Controlar el caudal, presión y/o la temperatura de alimentación desde el equipo.
- Conectar el controlador con el equipo sin cables, evitando así la necesidad de situarlos próximos.

Capítulo 10 Bibliografía

- MUÑOZ, M. y PAYRI, F. *Turbomáquinas térmicas*. Madrid: ETSSIM,1978
- MUÑOZ, M. y ROVIRA, A. J. *Máquinas térmicas*. U.N.E.D., 2011
- DIXON, S.L. *Termodinámica de turbomáquinas*. Dossat, 1978.
- GUITIERREZ, J. L. *Turbomáquinas térmicas, teoría y problemas*. Universidad del País Vasco, 2005 I.S.B.N. 84-8373-768-X
- SHIGLEY J.E., MISCHKE, C.R. *Diseño en ingeniería mecánica*. Mc Graw Hill, 6ª Ed.,2002 ISBN: 9684227787.
- FAIRES V.M. *Diseño de elementos de máquinas.* Tema 4. Limusa, 1999. ISBN: 9681842073.
- JUVINALL R.C. *Diseño de elementos de máquinas*. Limusa, 2ª Ed., 2013. Tema 10. ISBN:9786070504365.
- BENEDICT, G.F. *Nontraditional Manufacturing Processes*. Marcel Dekker Incorporated, 1987. ISBN: 0-8247-7352-7.
- MCGEOUGH, J.A. *Advanced Methods of Machining*. Chapman and Hall, 1988 ISBN: 0-412-31970-5.
- KNOWLTON, J. *Python*. (1 edición), 2009. Anaya. ISBN 978-84-415-2513-9.
- SERWAY, A. y JEWETT, J. *Physics for Scientists and Engineers*. Brooks/Cole, 6th ed, 2004. ISBN 0-534-40842-7.
- TIPLER, P. *Física para la ciencia y la tecnología*. Barcelona: Ed. Reverté, 2000. ISBN 84-291-4382-3.
- *Apuntes de Máquinas Térmicas*, departamento de ingeniería energética y fluidomecánica
- *Python*. Copyright ©2001-2020. [consulta: 26 de junio de 2020]. Disponible en: https://www.python.org/
- *ESPRESSIF SYSTEMS CO.* @2019, LTD. All rights reserved. [consulta: 13 de junio de 2020]. Disponible en: https://www.espressif.com/en/products/socs/esp32/overview
- *Doc. Python*. © Copyright 1990-2020. [consulta: 27 de junio de 2020]. Disponible en: https://docs.python.org/2/library/tkinter.html
- *Matplotlib* © Copyright 2002 – 2012. [consulta: 2 de julio de 2020]. Disponible en: https://matplotlib.org/

Anexo I planos

Anexo II código

Código del controlador

Principal

```
#include <SPI.h>

#include "pin_out.h"

#include "EEPROM.h"

#include "Espera.h"


//**********************Media Movil

#define Muestras_MediaMovil 7

#include "Media_Movil.h"


// ******************Termopares

#include "MAX31855K.h"

MAX31855k  Temp1(CS1,0),Temp2(CS3,0);


bool leer=1; //bandera para proceso de lectura

Media_Movil TM1(Muestras_MediaMovil),TM2(Muestras_MediaMovil);  //valores medios de cada medida

TEspera F_Muestreo,F_Display;  //variable para cuenta de tiempo de muestreo

char version[]="Turbina V1.0.";

const byte interruptPin = rpm;

unsigned long oldt=0;

int i=0,n=0;


boolean par=false;

void leer_sondas()

{float T;
```

```
   T=Temp1.readTempC();
   if(T>0) TM1.Media(T);
   T=Temp2.readTempC();
   if(T>0) TM2.Media(T);
}
void IRAM_ATTR isr()
{
if(par){
  oldt=micros();
}
  par^=1;
 // digitalWrite(ledPin,par);
}
void setup() {
  Serial.begin(1000000);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin),isr, RISING);
    oldt=0;
   F_Muestreo.set(100);  //muestreo cada 200ms
   F_Display.set(500); //frecuencia de display 500ms
}
void loop() {
//Actualización sondas
  if(F_Muestreo.done()){
          F_Muestreo.reset();
          leer_sondas();}
   if(oldt!=0){
      //Serial.print(i);
      //Serial.print(",");
      Serial.print(oldt);
```

```
    if(F_Display.done()){F_Display.reset();
                    Serial.print(",");
                    Serial.print(String(TM1.get()));
                    Serial.print(",");
                    Serial.print(String(TM2.get()));
                    Serial.print(",");
                    Serial.print(String(1.1134*analogRead(A2) + 171.89));
                    Serial.print(",");
                    Serial.println(String(1.1134*analogRead(A1) + 171.89));

    Serial.println();
    oldt=0;}
}
```

Biblioteca Espera.cpp

```
/*   Espera.c                                        */
/*   (c) 2019 Jose Angel Moneo                              */
/*   v1.3                                             */
/*   Clase Espera                                     */
/* ========================================================================
*/

#include "Espera.h"

//asigna el valor tiempo de espera a lavariable tsec
void TEspera::set(int espera)
{ Tsec=millis()+espera;
Tespera=espera; //alamacena el tiempo fijado
  fdone=false;
}
```

```
//espera a que se cumpla el tiempo de espera asignado
//if op es true realiza un reset automático al cumplirse el tiempo
boolean TEspera::done(bool op)
{  if (!fdone) if(Tsec<millis()) {fdone=true; if (op) reset();}
  return fdone;
}
void TEspera::reset(void)
{   Tsec=millis()+Tespera;
  fdone=false;
}
```

Biblioteca Espera.h

```
//  Clase Espera.h
// (c) Jose Angel Moneo (2016)
//  Sistema de espera sin usar timer
//
 // (c)  José Angel Moneo (2016)
// V1.2



#ifndef _Espera_class_H
#define _Espera_class_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif
class TEspera
{
```

```
 private:
      unsigned long Tsec;
      int Tespera; //tiempo almacenado
      boolean fdone; //marca tiempo finalizado, para no tener que comprobar el tiempo
despues de que se cumpla
public:
   TEspera(){}; //constructor para servo. indicamos el las salidas del servo
   boolean done(bool op=false);
   void set(int espera);
   void reset(); //Reinicia el tiempo preficado anteriormente
            };
 #endif
```

Biblioteca MAX31855K.cpp

```
#include "MAX31855k.h"
//////////////////////////////////////////////////////////////////////////////
// Description  : This constructor does the required setup
// Input        : uint8_t _cs: The Arduino pin number of the chip select line
//              for this instance
//              uint8_t _sck: The Arduino pin number of the SPI clock
//              uint8_t _so:  The Arduino pin number of the SPI MISO
//              uint8_t _vcc: The Arduino pin number to source the power
//              uint8_t _gnd: The Arduino pin number to sink the power
// Output       : Instance of this class with pins configured
// Return       : None
// Usage        : SparkFunMAX31855k <name>(<pinNumber>);
//////////////////////////////////////////////////////////////////////////////
MAX31855k::MAX31855k(const uint8_t _cs, const bool _debug) : cs(_cs), debug(_debug)
{
  // Redundant with SPI library if using default SS
```

```
  pinMode(cs, OUTPUT);
  // SCK & MOSI set in SPI library, MISO autoconfigures
  SPI.begin();
  digitalWrite(cs, HIGH);
}
////////////////////////////////////////////////////////////////////////////
// Change the cs pin
////////////////////////////////////////////////////////////////////////////
void MAX31855k::setCS(int pin)
{
  digitalWrite(cs, HIGH); // Make sure to set old cs high to disable the chip
  cs = pin;             //change to new cs pin
  pinMode(cs, OUTPUT);
  digitalWrite(cs, HIGH);
}
////////////////////////////////////////////////////////////////////////////
// Deconstructor does nothing.  It's up to the user to re-assign
// chip select pin if they want to use it for something else.  We don't call
// SPI.end() in case there is another SPI device we don't want to kill.
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
// Description  : This function reads the cold junction temperature
// Input        : None
// Output       : Loads member variables with data from the IC
// Return:      : None
// Usage        : <objectName>.readBytes();
// MAX31855K Memory Map:
//  D[31:18] Signed 14-bit thermocouple temperature data
//  D17      Reserved: Always reads 0
//  D16      Fault: 1 when any of the SCV, SCG, or OC faults are active, else 0
```

```
// D[15:4]  Signed 12-bit internal temperature
// D3       Reserved: Always reads 0
// D2       SCV fault: Reads 1 when thermocouple is shorted to V_CC, else 0
// D1       SCG fault: Reads 1 when thermocouple is shorted to gnd, else 0
// D0       OC  fault: Reads 1 when thermocouple is open-circuit, else 0
//////////////////////////////////////////////////////////////////////////////
void MAX31855k::readBytes(void)
{
  digitalWrite(cs, LOW);

  SPI.beginTransaction(SPISettings(4000000, MSBFIRST, SPI_MODE0));  // Defaults
  data.bytes[3] = SPI.transfer(0x00);
  data.bytes[2] = SPI.transfer(0x00);
  data.bytes[1] = SPI.transfer(0x00);
  data.bytes[0] = SPI.transfer(0x00);
  SPI.endTransaction();
  digitalWrite(cs, HIGH);
  return;
}

//////////////////////////////////////////////////////////////////////////////
// Description  : This function reads the current temperature
// Input        : SparkFunMAX31855k::units _u: The units of temperature to
//                return. C (default), K, R, or F
// Output       : Error messages before freezing
// Return:      : float: The temperature in requested units.
// Usage        : float tempC = <objectName>.read_temp();
//              float tempC = <objectName>.read_temp(SparkFunMAX31855k::C);
//              float tempF = <objectName>.read_temp(SparkFunMAX31855k::F);
//              float tempK = <objectName>.read_temp(SparkFunMAX31855k::K);
```

```
//              float tempR = <objectName>.read_temp(SparkFunMAX31855k::R);
////////////////////////////////////////////////////////////////////////////
float MAX31855k::readTemp(MAX31855k::units _u)
{
 int16_t value;
 float temp;

 readBytes();
 if (checkHasFault()) {
  return NAN;
 } else {
  // Bits D[31:18] are the signed 14-bit thermocouple temperature value
  if (data.uint32 & ((uint32_t)1 << 31)) { // Sign extend negative numbers
   value = 0xC000 | ((data.uint32 >> 18) & 0x3FFF);
  } else {
   value = data.uint32 >> 18; // Shift off all but the temperature data
  }
 }
 temp = value/4.0;
 switch (_u) {
 case F:
  temp = (temp * 9.0 / 5.0) + 32.0;
  break;
 case K:
  temp += 273.15;
  break;
 case R:
  temp = (temp + 273.15) * 9.0 / 5.0;
 case C:
 default:
```

```
   break;
  }
  return temp;
}
////////////////////////////////////////////////////////////////////////////
// Description  : This function reads the cold junction temperature
// Input        : None
// Output       : None
// Return:      : float: The temperature in ˚C
// Usage        : float tempC = <objectName>.readCJT();
////////////////////////////////////////////////////////////////////////////
float MAX31855k::readCJT(void)
{
  float ret;
  readBytes();
  if (checkHasFault())
  {
    return NAN;
  }
  if (data.uint32 & ((int32_t)1 << 15))
  { // Sign extend negative numbers
    ret = 0xF000 | ((data.uint32 >> 4) & 0xFFF);
  }
  else
  {
    ret = (data.uint32 >> 4) & 0xFFF;
  }
  return ret/16;
}
////////////////////////////////////////////////////////////////////////////
```

```
// Description  : This function checks the fault bits from the MAX31855K IC
// Input        : None
// Output       : Serial prints debug mesages if debug == true
// Return:      : Fault bits that were high, or 8 for unknow & 0 for no faults
// Usage        : checkHasFault();
////////////////////////////////////////////////////////////////////////////////
uint8_t MAX31855k::checkHasFault(void)
{
  if (!data.uint32) {
    // If all bits are low, either it's not wired right, or we actually measured
    // 0˚.  There is no way to tell. With debug turned on this will warn.
    if (debug)
      Serial.println(F("\nMAX31855K::All bits were zero.  Fishy..."));
  }
  if (data.uint32 & ((uint32_t)1<<16)) { // Bit D16 is high => fault
    if (data.uint32 & 1) {
      if (debug)
        Serial.println(F("\nMAX31855K::OC Fault: No Probe"));
      return 0b1;
    } else if (data.uint32 & (1<<1)) {
      if (debug)
        Serial.println(F("\nMAX31855K::SCG Fault: Thermocouple is shorted to GND"));
      return 0b10;
    } else if (data.uint32 & (1<<2)) {
      if (debug)
        Serial.println(F("\nMAX31855K::SCV Fault: Thermocouple is shorted to VCC"));
      return 0b100;
    } else {
      if (debug)
        Serial.println(F("\nMAX31855K::Unknown Fault"));
```

```
    return 0b1000;
   }
  } else {
   return 0; // No fault
  }
}
```

Biblioteca MAX31855K.h

```
/**************************************************************************
* Copyright (c) Jose Angel moneo Fernandez
*
**************************************************************************
/
/**************************************************************************
*
* @file                    MAX31855K.h
*
            Dependencies: SPI library for ESP32
* @version                 0.1.0
* @date                    19/10/03
 *                                                *
 * Library to read temperature from a MAX6675 type K thermocouple digitizer    *
 *                                                *
 * SO output format:                                        *
 * 16 bits                                                *
 * The first bit, D15, is a dummy sign bit and is always zero.              *
 * Bits D14:D3 contain the converted temperature in the order of MSB to LSB.   *
 * Bit D2 is normally low and goes high when the thermocouple input is open.   *
 * D1 is low to provide a device ID for the MAX6675 and bit D0 is tri-state.   *
 *                                                *
 * The method read_temp(max31855k::unit) takes a parameter which determines    *
```

```
 * the unit of temperature returned.                                *
 * max31855k::C will return degrees Celsius                         *
 * max31855k::K will return degrees Kelvin                          *
 * max31855k::F will return degrees Fahrenheit                      *
 * max31855k::R will return degrees Rankine                         *

*************************************************************************
**/

#ifndef _MAX31855k_h_
#define _MAX31855k_h_
#include <Arduino.h>
#include <SPI.h> // Have to include this in the main sketch too... (Using SPI)
const uint8_t NONE = 255; // This is used to indicate VCC or GND pin isn't used
class MAX31855k
{ public:
  // Simple Arduino API style guide functions
  inline float readTempC() { return readTemp(MAX31855k::C); }
  inline float readTempF() { return readTemp(MAX31855k::F); }
  inline float readTempR() { return readTemp(MAX31855k::R); }
  inline float readTempK() { return readTemp(MAX31855k::K); }
  // More advanced code concepts used below
  enum units {
    F, C, K, R
  };
  // If non-zero will turn on serial debugging messages
  uint8_t debug;
  // Returns the temperature in degrees F, K, R, or C (default if unspecified)
  float readTemp(MAX31855k::units _u=C);
  // Returns the cold junction temperature in ˚C
  float readCJT(void);
```

```
  // Pass a pin number to set as CS
  void setCS(int pin);
  MAX31855k(const uint8_t, const bool _debug=0);
  ~MAX31855k() {} // User responsible 4 reassigning pins & stopping SPI
protected:
  union { // Union makes conversion from 4 bytes to an unsigned 32-bit int easy
    uint8_t bytes[4];
    uint32_t uint32;
  } data;
  uint8_t cs;
  void readBytes(void);
  uint8_t checkHasFault(void);
};
#endif /* __max6675_H_ */
```

Biblioteca Media_Movil.cpp

```
/* ============================================================================
*/
/*                                                                            */
/*   Media_Movil.c                                                            */
/*   (c) 2019 Jose Angel Moneo                                                */
/*                                                                            */
/*   Clase Media Movil                                                        */
/* ============================================================================
*/
#include "Media_Movil.h"
Media_Movil::Media_Movil(int muestras){
   Muestras=muestras;
   for(int h=0;h<20;h++)     Medidas[h]=0; //borra tabla de medias
    S_Medidas=0; //Borra la suma de valores
```

```
}
float Media_Movil::Media(float valor)
{ S_Medidas-=Medidas[0]; //Resta el valor que sale de la tabla
  //Hace hueco moviendo los datos hacia el fondo y suma los datos para la media
  for(int i=0;i<Muestras;i++)
      Medidas[i]=Medidas[i+1];
    S_Medidas+=Medidas[Muestras-1]=valor;  //añade el valor al fina de la tabla y lo suma a la suma
  return get();
}
//toma la media movil actual
float Media_Movil::get()
{  return S_Medidas/Muestras;}
```

Biblioteca Media_Movil.h

```
//  Clase Media_Movil.h
// (c) Jose Angel Moneo (2019)
//
// devuelve su media movil
//  v1.1
 // (c)  José Angel Moneo (2019)
#ifndef _Media_Movil_class_H
#define _Media_Movil_class_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif
class Media_Movil
{ private:
```

```
  float  Medidas[20];  //Tabla de valores de las últimas medidas para media para 20 variables con una media de 20 maxima
  float S_Medidas;  //Suma de las medidas
  int Muestras;
public:
  Media_Movil(int muestras); //constructor para servo. indicamos el las salidas del servo
  float  Media(float valor);  //almacena un nuevo valor y devuelve la media movil actual
  float get();  //devuelve la media movil actual
          };
 #endif
```

Biblioteca pin_out.h

```
//Analogicas
#define A1 26
#define A2 25
#define A3 33
#define A4 32
//Reles
#define R1 2
#define R2 15
#define R3 13
#define R4 27
//cs SPI
#define CS1 5
#define CS2 17
#define CS3 16
#define CS4 4
#define DRDY 0
//#define Wire18b20 34
#define rpm 35
```

//SCLK = 18, MISO = 19, MOSI = 23,

Código software

```
from tkinter import *
import numpy as np
import serial
import threading
import matplotlib.pylab as plt
import math
from matplotlib.backends.backend_tkagg import (
    FigureCanvasTkAgg, NavigationToolbar2Tk)
# Implement the default Matplotlib key bindings.
from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
import serial
import time
import sys


global a,b, esp32
a=0
b=0

def graficar():
    global selx_fila,sely_fila,etx,ety
    v_datfich2 = []
    v_dataarray2 = []
    datos2 = []
    fichero2 = open("datos2.txt", 'r')
    v_datfich2 = (fichero2.readlines())
    fichero2.close()
    n = 0
    mat_graficar=[]
    while n < (len(v_datfich2)):
        temp2 = (v_datfich2[n])
        temp2 = str(temp2.strip("b \ n \ r '"))
        v_temp2 = []
        v_temp2 = (np.array(temp2.split()))
        v_temp2 = v_temp2.astype(str)
        mat_graficar.append(v_temp2)
        n += 1

    fig = Figure(figsize=(7, 4), dpi=100)
    v_x=[]
    v_y=[]
    n=0
    while n<len(mat_graficar):
        v_x.append(float(mat_graficar[n][selx_fila]))
        v_y.append(float(mat_graficar[n][sely_fila]))
        n+=1
    grafica = fig.add_subplot(111)
    grafica.plot(v_x, v_y)
    grafica.set_xlabel(etx)
    grafica.set_ylabel(ety)
    canvas = FigureCanvasTkAgg(fig, master=marco_graf)
    canvas.draw()
```

```
    canvas.get_tk_widget().grid(row=1, column=0, padx=25, pady=25)

def calculos():
    global term_leer
    v_datfich=[]
    v_datarray=[]
    datos=[]
    fichero = open("datos1.txt", 'r')
    v_datfich=(fichero.readlines())
    fichero.close()
    n=0
    while n<(len(v_datfich)):
        temp=(v_datfich[n])
        temp=str(temp.strip("b \ n \ r '"))
        v_temp=[]
        v_temp=(np.array(temp.split()))
        v_temp=v_temp.astype(np.float)
        if len(v_temp)>1:
            datos.append(list(v_temp))
        if len(v_temp)==1:
            datos.append(list([v_temp[0],"a","a","a","a"]))
        n+=1
    #hasta aqui lee el fichero y prepara la matriz
    #ahora vamos a rellenar los datos que sean necesarios antes de
transformar las magnitudes
    n=0
    while n<len(datos):
        datos[n][4]=float(ent_Pext.get())
        n+=1
    #se interpola los valores de presión y temperaatura
    n=0
    i=0
    j=0
    k=0
    while (datos[n][1] or datos[n][2] or datos[n][3] or
datos[n][4])=="a":
        n+=1
    j=n
    n=0
    while n<j:
        datos[n][1] = datos[j][1]
        datos[n][2] = datos[j][2]
        datos[n][3] = datos[j][3]
        datos[n][4] = datos[j][4]
        n += 1
    n=len(datos)-1
    while (datos[n][1] or datos[n][2] or datos[n][3] or
datos[n][4])=="a":
        n-=1
    i=n
    n=len(datos)-1
    while i<n:
        datos[n][1] = datos[i][1]
        datos[n][2] = datos[i][2]
        datos[n][3] = datos[i][3]
        datos[n][4] = datos[i][4]
        n -= 1
    n=0
    while n<len(datos):
        if (datos[n][1] or datos[n][2] or datos[n][3] or
datos[n][4])=="a":
```

```
            k=n
            while (k<len(datos)) and ((datos[k][1] or datos[k][2] or
datos[k][3] or datos[k][4])=="a") :
                    k+=1
            pendientet0=(datos[k][1]-datos[j][1])/(datos[k][0]-
datos[j][0])
            indt0=datos[j][1]-(pendientet0*datos[j][0])
            datos[n][1]=pendientet0*datos[n][0]+indt0

            pendientet2 = (datos[k][2] - datos[j][2]) / (datos[k][0] -
datos[j][0])
            indt2 = datos[j][2] - (pendientet2 * datos[j][0])
            datos[n][2] = pendientet2 * datos[n][0] + indt2

            pendientep0 = (datos[k][3] - datos[j][3]) / (datos[k][0] -
datos[j][0])
            indp0 = datos[j][3] - (pendientep0 * datos[j][0])
            datos[n][3] = pendientep0 * datos[n][0] + indp0

            pendientep2 = (datos[k][4] - datos[j][4]) / (datos[k][0] -
datos[j][0])
            indp2 = datos[j][4] - (pendientep2 * datos[j][0])
            datos[n][4] = pendientep2 * datos[n][0] + indp2
        else:
            j=k
        n+=1
    n=0

    #una vez preparados los datos se extraen las magnitudes.
    print("calculo de magnitudes")
    magnitudes=[]
    n=0
    while n<(len(datos)-1):
        magnitudes.append([])

magnitudes[n].append((((float(datos[n][0])+float(datos[n+1][0]))/2)-
float(datos[0][0]))/1000000)

        magnitudes[n].append(60000000/(float(datos[n+1][0])-
float(datos[n][0])))

        magnitudes[n].append(float((datos[n][3]-500)*0.0017225)*1e5)

        magnitudes[n].append(float(datos[n][1])+273.15)

        magnitudes[n].append(float(datos[n][4])) #si se añade el
sensor de presion a la salida es necesario modificar esta linea

        magnitudes[n].append(float(datos[n][2]) + 273.15)
        n+=1
    #Limpiamos las revoluciones
    n=0
    while n<(len(magnitudes)-1):
        if
(float(magnitudes[n][1])>float(1.5*float(magnitudes[n+1][1])))and
magnitudes[n][1]>1000:
            magnitudes[n][1]=((magnitudes[n-
1][1]+magnitudes[n+1][1])/2)
        n+=1
    #calculamos la aceleración
    print("calculo aceleracion")
```

```
    num_puntos = 11
    if(len(magnitudes)>num_puntos):
        n=int((num_puntos-1)/2)
        i=n-(int((num_puntos-1)/2))
        print("i vale:",i)
        print("n vale:",n)
        while n < (len(magnitudes) - ((num_puntos-1)/2)):
            sum_x=0
            sum_y=0
            sum_xy=0
            sum_x2=0
            aceleracion=0
            while i<((n+((int(num_puntos)-1)/2))+1):
                sum_x=sum_x+float(magnitudes[i][0])
                sum_y=sum_y+float(magnitudes[i][1])

sum_xy=sum_xy+float(magnitudes[i][0])*float(magnitudes[i][1])
                sum_x2=sum_x2+float(magnitudes[i][0]**2)
                i+=1
            aceleracion=(2*np.pi/60)*(((num_puntos)*sum_xy)-
(sum_y*sum_x))/(((num_puntos)*sum_x2)-(sum_x**2))
            magnitudes[n].append(aceleracion)
            i=n-(int((num_puntos-1)/2)-1)
            n += 1
        n=0
        while n<len(magnitudes):
            if (n<(int((num_puntos-1)/2))):
                acel=(magnitudes[int((num_puntos-
1)/2)][6]/magnitudes[int((num_puntos-1)/2)][0])*magnitudes[n][0]
                magnitudes[n].append(acel)
            n+=1
        n = 0
        pend=(magnitudes[len(magnitudes) - (int((num_puntos - 1) /
2))-1][6])/(magnitudes[len(magnitudes) - (int((num_puntos - 1) / 2))-
1][0]-magnitudes[len(magnitudes)-1][0])
        indpen=-magnitudes[len(magnitudes)-1][0]*pend
        while n < len(magnitudes):
            if (n >= (len(magnitudes) - (int((num_puntos - 1) / 2)))):
                acel=pend*magnitudes[n][0]+indpen
                magnitudes[n].append(acel)
            n += 1
    else:
        n=0
        while n<len(magnitudes):
            magnitudes[n].append(0)


    #calculamos potencia efectiva, potencia de perdidas y rendimiento
isoentrópico
    print("calculo potencia")
    n=0
    while n<len(magnitudes):

        if magnitudes[n][6]>=0:

magnitudes[n].append(magnitudes[n][6]*magnitudes[n][1]*float(ent_I.get
()))
            magnitudes[n].append("a")

T2_teorica=magnitudes[n][3]*((magnitudes[n][4]/(magnitudes[n][2]+magni
tudes[n][4]))**((1.4-1)/1.4))
```

```
            if(T2_teorica-magnitudes[n][3]!=0):
                rend_iso=((magnitudes[n][5]-
magnitudes[n][3])/(T2_teorica-float(magnitudes[n][3])))

            else:
                rend_iso=(-1)
            try:
                if rend_iso > 0 and rend_iso<1:
                    magnitudes[n].append(rend_iso)
                else:
                    magnitudes[n].append("a")
            except:
                magnitudes[n].append("a")
        else:
            magnitudes[n].append("a")
            magnitudes[n].append(abs(magnitudes[n][6] *
magnitudes[n][1] * float(ent_I.get())))
            magnitudes[n].append("a")
        n+=1
    #ordenamos las magnitudes por velocidad angular en una matriz
auxiliar
    print("ordenar")
    n=0
    matriz_aux=[]
    while n<len(magnitudes):
        matriz_aux.append(magnitudes[n])
        n+=1

    n=0
    i=0
    k=0
    while n<len(magnitudes):
        num_menor = matriz_aux[0][1]
        while i<len(matriz_aux):
            if matriz_aux[i][1]<=num_menor:
                num_menor=matriz_aux[i][1]
                k=i
            i+=1
        i=0
        magnitudes[n].append(matriz_aux[k][1])
        magnitudes[n].append(matriz_aux[k][7])
        magnitudes[n].append(matriz_aux[k][8])
        magnitudes[n].append(matriz_aux[k][9])
        matriz_aux.pop(k)
        n+=1
    #reordenada la matriz con w,Pe,pp, rend iso, se pasa al cálculo de
las falta de datos de Pe, Pp y rend iso
    print("reordenando potencia")

    n=0
    i=0
    magnitudes[0][11] = 0
    magnitudes[len(magnitudes)-1][11] = 0
    while n<len(magnitudes):
        if magnitudes[n][11]=="a":
            i=n
            while magnitudes[i][11]=="a":
                i+=1
            if ((magnitudes[i][10]-magnitudes[n-1][10])!= 0):
                magnitudes[n][11]=(((magnitudes[i][11]-magnitudes[n-
```

```
1][11])/(magnitudes[i][10]-magnitudes[n-
1][10]))*magnitudes[n][10])+(magnitudes[n-1][11]-(magnitudes[n-
1][10]*(magnitudes[i][11]-magnitudes[n-1][11])/(magnitudes[i][10]-
magnitudes[n-1][10])))
            else:
                magnitudes[n][11] =magnitudes[n-1][11]
        n+=1
    n=0
    i=0
    print("reordenando potencia perdidas")
    magnitudes[n][12] = 0
    magnitudes[len(magnitudes) - 1][12] = 0
    while n < len(magnitudes):
        if magnitudes[n][12] == "a":
            i = n
            while magnitudes[i][12] == "a":
                i += 1
            if (magnitudes[i][10] - magnitudes[n - 1][10] != 0):
                magnitudes[n][12] = (((magnitudes[i][12] -
magnitudes[n - 1][12]) / (
                                     magnitudes[i][10] - magnitudes[n -
1][10])) * magnitudes[n][10]) + (
                                                          magnitudes[n -
1][12] - (magnitudes[n-1][10]*(magnitudes[i][12] - magnitudes[n -
1][12]) / (

magnitudes[i][10] - magnitudes[n - 1][10])))
            else:
                try:
                    magnitudes[n][12] =magnitudes[n-1][12]
                except:
                    magnitudes[n][12]=0
        n += 1
    print("reordenando rendimiento")
    n=0
    i=0
    magnitudes[n][13] = 0
    magnitudes[len(magnitudes) - 1][13] = 0
    while n < len(magnitudes):
        if magnitudes[n][13] == "a":
            i = n
            while magnitudes[i][13] == "a":
                i += 1
            if (magnitudes[i][10] - magnitudes[n - 1][10] != 0):
                magnitudes[n][13] = (((magnitudes[i][13] -
magnitudes[n - 1][13]) / (
                                magnitudes[i][10] - magnitudes[n - 1][10])) *
magnitudes[n][10]) + (
                                                    magnitudes[n - 1][13] -
(magnitudes[n-1][10]*(magnitudes[i][13] - magnitudes[n - 1][13]) / (
                                                    magnitudes[i][10] -
magnitudes[n - 1][10])))
            else:
                magnitudes[n][13] =magnitudes[n-1][13]

        n += 1
    #con estas magnitudes calculamos el resto, Pi, Mi, rend mec y rend
total
    n=0
    while n<len(magnitudes):
        magnitudes[n].append(magnitudes[n][11]+magnitudes[n][12])
```

```
            if magnitudes[n][10]!=0:
                magnitudes[n].append(magnitudes[n][14]/magnitudes[n][10])
            else:
                magnitudes[n].append(0)
            if magnitudes[n][14]!=0:
                magnitudes[n].append(magnitudes[n][11]/magnitudes[n][14])
            else:
                magnitudes[n].append(0)
            magnitudes[n].append(magnitudes[n][13]*magnitudes[n][16])

            n+=1

        fichero2 = open("datos2.txt", 'w')
        n=0
        i=0
        while n<len(magnitudes):
            while i<len(magnitudes[n]):
                fichero2.write(str(magnitudes[n][i]))
                fichero2.write(" ")
                i+=1
            n+=1
            i=0
            fichero2.write("\n")
        fichero2.close()
        #calculo de máximos
        n=0
        var_maxrpm = 0
        while n<len(magnitudes):
            if magnitudes[n][10]>var_maxrpm:
                var_maxrpm=magnitudes[n][10]
            n+=1

        n=0
        var_maxpot = 0
        while n<len(magnitudes):
            if magnitudes[n][11]>var_maxpot:
                var_maxpot=magnitudes[n][11]
            n+=1

        n=0
        var_maxrend = 0
        while n<len(magnitudes):
            if magnitudes[n][17]>var_maxrend:
                var_maxrend=magnitudes[n][17]
            n+=1
        sa_rpmmax['text']=round(var_maxrpm,2)
        sa_Pemax['text'] =round(var_maxpot,2)
        sa_renmax['text'] =round(var_maxrend,2)
        graficar()


    def lecturas():
        global a,b, esp32, term_leer
        t0 = 0
        while b==0:
            try:
                esp32 = serial.Serial(ent_Puer.get(), 1000000,timeout=1)
                et_estadoPuer['text'] = "Conectado"
                d_inst = str(esp32.readline())     #elimino la primera
    linea para que no de problemas
                while a==0:
```

```
                try:
                    d_inst =str(esp32.readline())
                    d_inst = str(d_inst.strip("b \ n \ r '"))
                    d_inst=str([d_inst.replace(',',' ')])
                    d_inst =str(d_inst.strip("]['"))
                    v_inst=[]
                    v_inst = np.array(d_inst.split())
                    v_inst = v_inst.astype(np.float)


                    t1=v_inst[0]
                    if t0!=0:

                        sa_rpminst['text']=60000000/(t1-t0)
                    t0=t1
                    if len(v_inst)>1:
                        Pinst=v_inst[3]*222.24 -103421.38
                        sa_Pinst['text']=str(Pinst)
                        sa_T0inst['text'] =str(v_inst[1])
                        sa_T1inst['text'] =str(v_inst[2])
                except:
                    print("No puedo imprimir porque no tengo datos")



            d_inst = str(esp32.readline())  # elimino la primera linea
para que no de problemas

            fichero=open("datos1.txt", 'w')
            term_leer=False
            while a==1:
                d_inst = str(esp32.readline())
                d_inst = str(d_inst.strip("b \ n \ r '"))
                d_inst = str([d_inst.replace(',', ' ')])
                d_inst = str(d_inst.strip("]['"))
                fichero.write((d_inst))
                fichero.write("\n")
            fichero.close()
            esp32.close()
            term_leer = True
        except:
            et_estadoPuer['text']="Sin conexión"

def seleccion():
    global sel_visual, selx_fila,sely_fila,etx,ety
    sel_visual.get()

    if sel_visual.get()==1:
        selx_fila=0
        sely_fila=1
        etx="Tiempo (s)"
        ety="Velocidad angular (rpm)"
    if sel_visual.get()==2:
        selx_fila=0
        sely_fila=6
        etx="Tiempo (s)"
        ety="Aceleración angular (rad/s^2)"
    if sel_visual.get()==3:
        selx_fila=0
        sely_fila=2
```

```
            etx="Tiempo (s)"
            ety="Presión (Pa)"
        if sel_visual.get()==4:
            selx_fila=0
            sely_fila=3
            etx="Tiempo (s)"
            ety="Temperatura 0 (k)"
        if sel_visual.get()==5:
            selx_fila=0
            sely_fila=5
            etx="Tiempo (s)"
            ety="Temperatura 2 (K)"
        if sel_visual.get()==6:
            selx_fila=10
            sely_fila=11
            etx="Velocidad angular (rpm)"
            ety="Potencia efectiva (W)"
        if sel_visual.get()==7:
            selx_fila=10
            sely_fila=14
            etx="Velocidad angular (rpm)"
            ety="Potencia indicada (W)"
        if sel_visual.get()==8:
            selx_fila=10
            sely_fila=15
            etx="Velocidad angular (rpm)"
            ety="Par indicado (Nm)"
        if sel_visual.get()==9:
            selx_fila=10
            sely_fila=13
            etx="Velocidad angular (rpm)"
            ety="Rendimiento isoentrópico"
        if sel_visual.get()==10:
            selx_fila=10
            sely_fila=16
            etx="Velocidad angular (rpm)"
            ety="Rendimiento mecánico"
        if sel_visual.get()==11:
            selx_fila=10
            sely_fila=17
            etx="Velocidad angular (rpm)"
            ety="Rendimiento total"
        graficar()
    def var_usuario():
        print(ent_Pext.get())
        print(ent_I.get())
    def ejecutar():
        global a
        a=1
        sa_est['text']="en curso"
    def parar():
        global a,term_leer
        a=0
        sa_est['text']="concluida"
        while term_leer != True:
            print("Espero a true=",term_leer)
        calculos()
```

```
mutex = threading.Lock()

ventana=Tk()
ventana.title("BETM")
ventana.geometry("1700x600")
ventana.configure(background='mint cream')

global sel_visual,selx_fila,sely_fila,etx,ety,term_leer,esp32
term_leer = True
sel_visual=IntVar()
selx_fila=0
sely_fila=1
etx="Tiempo (s)"
ety="Velocidad angular (rpm)"
ent_Puer="'COM3'"

c_fondo="mint cream"
c_tex="mint cream"
c_entrada="mint cream"
f_titulo="ms reference sans serif",12,
f_texto="ms reference sans serif",10,
f_entradas='arial'
tam_tex=1
tam_marpadx=5
tam_marpady=5
tam_texpadx=10
tam_texpady=0

########################################################################
########################
marco_var=Frame(ventana)
marco_var.configure(background=c_fondo,
padx=tam_marpadx,pady=tam_marpady,bd=1,relief='flat')
marco_var.grid(row=0,column=0, sticky="WENS")

marco_pru=Frame(ventana)
marco_pru.configure(background=c_fondo,
padx=tam_marpadx,pady=tam_marpady,bd=2,relief='flat')
marco_pru.grid(row=1,column=1, sticky="WENS")

marco_graf=Frame(ventana)
marco_graf.configure(background='white',
padx=tam_marpadx,pady=tam_marpady,bd=1,relief='flat')
marco_graf.grid(row=1,column=0, sticky="WE")

marco_t=Frame(ventana)
marco_t.configure(background=c_fondo,
padx=tam_marpadx,pady=tam_marpady,bd=1,relief='flat')
marco_t.grid(row=2,column=0, sticky="WE")

marco_rpm=Frame(ventana)
marco_rpm.configure(background=c_fondo,
padx=tam_marpadx,pady=tam_marpady,bd=0.5,relief='flat')
marco_rpm.grid(row=3,column=0, sticky="WE")
########################################################################
#################################



et_param=Label(marco_var, text="Parámetros del ensayo          ",
```

```
font=f_titulo)
et_param.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_param.grid(row=0,column=0)

et_paramP=Label(marco_var, text="Presión exterior (Pa): ",
font=f_texto)
et_paramP.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_paramP.grid(row=0,column=1)

ent_Pext=Entry(marco_var,justify="center")
ent_Pext.grid(row=0 ,column=2)
ent_Pext.insert(0, "101325")

et_paramI=Label(marco_var, text="Inercia (Kg m2): ", font=f_texto)
et_paramI.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_paramI.grid(row=0,column=3)

ent_I=Entry(marco_var,justify="center")
ent_I.grid(row=0 ,column=4)
ent_I.insert(0, "1e-5")

et_paramPuer=Label(marco_var, text="Puerto: ", font=f_texto)
et_paramPuer.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_paramPuer.grid(row=0,column=5)

ent_Puer=Entry(marco_var,justify="center")
ent_Puer.grid(row=0 ,column=6)
ent_Puer.insert(0, 'COM3')

et_estadoPuer=Label(marco_var, font=f_texto)
et_estadoPuer.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_estadoPuer.grid(row=0,column=7)

########################################################################
#########################


et_t=Label(marco_t, text="Gráficas tiempo", font=f_titulo)
et_t.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_t.grid(row=0,column=0)

bt_rpm=Radiobutton(marco_t, text="w", font=f_texto)
bt_rpm.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=1)
bt_rpm.grid(row=0,column=1)
bt_rpm.select()

bt_ac=Radiobutton(marco_t, text="a", font=f_texto)
bt_ac.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=2)
bt_ac.grid(row=0,column=2)

bt_P=Radiobutton(marco_t, text="P", font=f_texto)
bt_P.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=3)
```

```
bt_P.grid(row=0,column=3)

bt_T0=Radiobutton(marco_t, text="T0", font=f_texto)
bt_T0.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=4)
bt_T0.grid(row=0,column=4)

bt_T2=Radiobutton(marco_t, text="T2", font=f_texto)
bt_T2.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=5)
bt_T2.grid(row=0,column=5)

#########################################################################
#########################

et_rpm=Label(marco_rpm, text="Gráficas rpm", font=f_titulo)
et_rpm.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_rpm.grid(row=0,column=0)

bt_Pe=Radiobutton(marco_rpm, text="Pe", font=f_texto)
bt_Pe.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=6)
bt_Pe.grid(row=0,column=1)

bt_Me=Radiobutton(marco_rpm, text="Pi", font=f_texto)
bt_Me.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=7)
bt_Me.grid(row=0,column=2)

bt_Pi=Radiobutton(marco_rpm, text="Mi", font=f_texto)
bt_Pi.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=8)
bt_Pi.grid(row=0,column=3)

bt_iso=Radiobutton(marco_rpm, text="Rend. iso.", font=f_texto)
bt_iso.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=9)
bt_iso.grid(row=0,column=4)

bt_mec=Radiobutton(marco_rpm, text="rend. mec.", font=f_texto)
bt_mec.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=10)
bt_mec.grid(row=0,column=5)

bt_tot=Radiobutton(marco_rpm, text="rend. total", font=f_texto)
bt_tot.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex, variable=sel_visual, command=seleccion, value=11)
bt_tot.grid(row=0,column=6)

#########################################################################
#########################

et_pru=Label(marco_pru, text="Valores instantáneos", font=f_titulo)
et_pru.configure(padx=tam_texpadx, pady=tam_texpady+15, bg=c_tex,
height=tam_tex)
et_pru.grid(row=0,column=0)

et_rpminst=Label(marco_pru, text="Velocidad angular (rpm)",
font=f_texto)
et_rpminst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
```

```
height=tam_tex)
et_rpminst.grid(row=1,column=0)

sa_rpminst=Label(marco_pru, font=f_texto)
sa_rpminst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_rpminst.grid(row=1,column=1)

et_Pinst=Label(marco_pru, text="Presión (Pa)", font=f_texto)
et_Pinst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_Pinst.grid(row=2,column=0)

sa_Pinst=Label(marco_pru, font=f_texto)
sa_Pinst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_Pinst.grid(row=2,column=1)

et_T0inst=Label(marco_pru, text="Temperatura entrada (°C)",
font=f_texto)
et_T0inst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_T0inst.grid(row=3,column=0)

sa_T0inst=Label(marco_pru, font=f_texto)
sa_T0inst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_T0inst.grid(row=3,column=1)

et_T1inst=Label(marco_pru, text="Temperatura salida (°C)",
font=f_texto)
et_T1inst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_T1inst.grid(row=4,column=0)

sa_T1inst=Label(marco_pru, font=f_texto)
sa_T1inst.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_T1inst.grid(row=4,column=1)

et_p=Label(marco_pru, text="Prueba", font=f_titulo)
et_p.configure(padx=tam_texpadx, pady=tam_texpady+15, bg=c_tex,
height=tam_tex)
et_p.grid(row=5,column=0)

bot_ini=Button(marco_pru, text=("INICIAR"), font=f_texto,
command=ejecutar)
bot_ini.configure(padx=tam_texpadx+10, pady=tam_texpady+2, bg="green",
height=tam_tex, bd=1)
bot_ini.grid(row=6, column=0)

et_est=Label(marco_pru, text="Estado de la prueba", font=f_texto)
et_est.configure(padx=tam_texpadx, pady=tam_texpady+10, bg=c_tex,
height=tam_tex)
et_est.grid(row=7,column=0)

sa_est=Label(marco_pru, font=f_texto)
sa_est.configure(padx=tam_texpadx, pady=tam_texpady+10, bg=c_tex,
height=tam_tex)
sa_est.grid(row=7,column=1)
```

```
bot_stop=Button(marco_pru, text=("PARAR"), font=f_texto,
command=parar)
bot_stop.configure(padx=tam_texpadx+14, pady=tam_texpady+2, bg="red",
height=tam_tex, bd=1)
bot_stop.grid(row=8, column=0)

et_max=Label(marco_pru, text="Valores máximos", font=f_titulo)
et_max.configure(padx=tam_texpadx, pady=tam_texpady+15, bg=c_tex,
height=tam_tex)
et_max.grid(row=9,column=0)

et_rpmmax=Label(marco_pru, text="velocidad angular (rpm):",
font=f_texto)
et_rpmmax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_rpmmax.grid(row=10,column=0)

sa_rpmmax=Label(marco_pru, font=f_texto)
sa_rpmmax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_rpmmax.grid(row=10,column=1)

et_Pemax=Label(marco_pru, text="Potencia efctiva (W):", font=f_texto)
et_Pemax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_Pemax.grid(row=11,column=0)

sa_Pemax=Label(marco_pru, font=f_texto)
sa_Pemax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_Pemax.grid(row=11,column=1)

et_renmax=Label(marco_pru, text="Rendimiento total:", font=f_texto)
et_renmax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
et_renmax.grid(row=12,column=0)

sa_renmax=Label(marco_pru, font=f_texto)
sa_renmax.configure(padx=tam_texpadx, pady=tam_texpady, bg=c_tex,
height=tam_tex)
sa_renmax.grid(row=12,column=1)



t = threading.Thread(target=lecturas)
t.setDaemon(True)
t.start()



ventana.mainloop()
b=1
a=1
try:
    esp32.close()
except:
    print("no hay puerto")
sys.exit()
```