UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PROYECTO FIN DE CARRERA

INSTALACIÓN DE SENSORES Y ANÁLISIS DE DATOS DE UN AEROGENERADOR DE EJE VERTICAL PARA USO DOMÉSTICO

AUTOR: Antonio Serda Mena
DIRECTOR: Juan Romeo Granados

MADRID, Junio de 2020

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título **«Instalación de sensores y análisis de datos de un aerogenerador de eje vertical para uso doméstico»**

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2019-2020 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Antonio Serda Mena Fecha: 15/ julio/ 2020

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.:  Fecha: 16/ 07/ 2020

INSTALACIÓN DE SENSORES Y ANÁLISIS DE DATOS DE UN AEROGENERADOR DE EJE VERTICAL PARA USO DOMÉSTICO

Autor: Serda Mena, Antonio

Director: Romeo Granados, Juan

Entidad colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Introducción

Ante el auge del autoconsumo doméstico de energía eléctrica en Europa en los últimos años y los cambios de la leyes en España en relación a esta materia surgen nuevas tecnología para suplir esta demanda. Actualmente, la misma se encuentra cubierta en su mayoría gracias a los paneles fotovoltaicos, ya que en el ámbito de la energía eólica para uso doméstico todavía existe mucho margen de mejora.

Debido a la falta de una generalización de este último tipo de energía como referente de eficiencia y comodidad en el consumidor doméstico surge la idea central de este proyecto: el «Aerogenerador Inteligente» o «Smart AG». Este es un prototipo de turbina eólica de eje vertical de tipo Darrieus diseñada por el director de este proyecto: Juan Romeo Granados. La misma dispone de cinco palas que giran en torno a un eje móvil, el cual iría conectado a un motor eléctrico. A su vez, está pensado que en el futuro el aerogenerador cuente con un sistema electrónico que permita obtener la máxima eficiencia para la extracción de energía.

Para poder cumplir este último objetivo es necesario disponer de las tecnologías y las herramientas adecuadas para ello. En este proyecto se desarrollan los sistemas necesarios para:

- Tomar los valores de tiempos de conmutación de un anemómetro empleado para medir la velocidad lineal del viento.
- Tomar los valores de tiempos de conmutación de un sensor de efecto Hall o Holzer empleado para medir la velocidad angular de la turbina.
- Activar un sistema de frenado de emergencia de la turbina en caso de sobrepasar la velocidad límite especificada.
- Calcular la posición del centro de masas del perfil de ala de la turbina.
- Calcular el momento de inercia de la turbina.
- Disponer de los valores de velocidad y aceleración del viento y de la turbina para cada instante de tiempo siendo capaz de mostrarlos de forma gráfica.
- Calcular el par y la potencia mecánica en el eje de la turbina siendo capaz de mostrarlos de forma gráfica.
- Estimar los coeficientes de pérdidas de la turbina.
- Dibujar un prototipo de la curva de seguimiento óptimo de la turbina, necesaria para el uso de dispositivos de control vectorial. Para dibujar la real sería necesario sustituir la potencia mecánica en el eje por la potencia eléctrica generada en cada instante por un motor conectado a la turbina.

Metodología

Para el diseño de las tres primeras funciones se ha empleado el programa Arduino. Estas se han instalado en una placa de tipo Arduino Uno a la que se han conectado un anemómetro, un sensor Hall, un sevomotor y un módulo *bluetooth*. El anemómetro empleado dispone de tres cazoletas y conmuta dos veces por vuelta. El sensor de efecto Hall se encuentra acoplado a la carcasa de la turbina y detecta una conmutación cada vez que una pala pasa cerca del mismo. Cada pala tiene un imán adherido en su punto central. El servomotor se instala en un sistma de freno diseñado por el director del proyecto. Cuando la velocidad del viento o de la turbina superan un límite previamente especificado el servomotor se activa. El módulo *bluetooth* se emplea para enviar los datos de tiempos de conmutación del anemómetro y del sensor Hall a un ordeandor y así poder analizarlos posteriormente. Todo lo relativo al envío de datos a través de este sistema ha sido diseñado por el alumno Javier Colinas Cano en su respectivo Proyecto Fin de Carrera: «Medición, transmisión y análisis de datos de un aerogenerador de eje vertical para uso doméstico».

Para el diseño del resto de las funciones se ha empleado el programa Matlab. El cálculo de la posición del centro de masas del perfil de ala, así como el cálculo del momento de inercia de la turbina, se ha realizado discretizando el área del perfil de ala y operando en función de la posición de cada punto individual del sistema. Para el momento de inercia es necesario añadir los momentos del resto de elementos de la turbina, además de las palas. La obtención de velocidades para cada instante de tiempo se consigue gracias a los valores de tiempos de conmutación enviados al ordenador por el alumno Javier Colinas Cano. Con estos vectores se calculan los valores de velocidad para dichos tiempos de conmutación usando diferenciales de tiempo. Finalmente, se interpolan los valores de velocidad para cada instante, teniendo, en este caso, una resolución de [ms]. Mediante los valores de velocidad se puede calcular la aceleración; esta última, junto al momento de inercia de la turbina, permite obtener el momento de fuerza y la potencia mecánica para cada instante.

Resultados

Los programas presentados en este proyecto junto a los dispositivos que en el mismo se indican consiguen los objetivos propuestos. Sin embargo, inicialmente se observaban fallos repentinos en la toma de datos que producían que no se detectara alguna de las conmutaciones del sensor Hall a altas velocidades de la turbina. Esto se solucionó creando varios modos de uso en el programa de Matlab con los que se detecta el error y se corrige mediante una aproximación. A su vez, el método empleado para medir la velocidad de la turbina, empleando para ello un imán en cada pala, producía errores de medida de la aceleración, ya que los imanes no se encontraban equidistantes. Para solucionar esto se crearon dos modos de uso adicionales en el programa: uno de ellos medía las variaciones de tiempo de cada uno de los cinco imanes en relación a sí mismo y el otro suprimía todos los imanes excepto uno, ya que así se evitaba en mayor medida la aparición de errores en la detección de las conmutaciones.

Respecto al programa de estimación de los coeficientes de pérdidas, este realiza los cálculos en función de una curva de frenado de la turbina en donde no se aplican momentos externos a la misma. Según cuantos puntos de la curva se decida tomar se

calcula un número de coeficientes equivalente, siendo ascendente el orden de los mismos. Teoricamente, cuanto mayor sea el orden del coeficiente menor es su valor en este tipo de movimientos, siendo los mismos positivos y llegando un punto a partir del que son despreciables. Sin embargo, la turbina presenta vibraciones durante su movimiento, por lo que al aplicar el programa se calculan unos coeficientes que modelan dichas oscilaciones, siendo los mismos inseparables de los que modelarían exclusivamente las pérdidas.

Finalmente, el programa de obtención del prototipo de la curva de seguimiento óptimo de la turbina funciona adecuadamente. Aun así, sería necesario instalar un vatímetro a la salida de un motor eléctrico conectado al eje de la turbina para poder obtener la curva real a programar en los dispositivos de control vectorial pertinentes.

Conclusiones

Este proyecto presenta una serie de herramientas, procedimientos y recomendaciones útiles orientadas a afrontar los primeros pasos del desarrollo de una turbina eólica de estas características. A lo largo de el mismo se detallan errores que han podido ocurrir así como sus soluciones.

Tras este proyecto, está planeada la mejora del diseño de la turbina en función de los resultados obtenidos gracias a la ejecución de los porgramas aquí mostrados. También se han de elegir e instalar dispositivos como un motor eléctrico, convertidores de potencia y una reductora, en caso de ser necesaria, para la obtención de energía eléctrica y su posterior transmisión a la red.

INSTALLATION OF SENSORS AND DATA ANALYSIS OF A VERTICAL AXIS WIND TURBINE FOR DOMESTIC USE

Author: Serda Mena, Antonio

Supervisor: Romeo Granados, Juan

Colaborating institution: ICAI – Universidad Pontificia Comillas

ABSTRACT

Introduction

Given the rise of domestic self-consumption of electrical energy in Europe in recent years and the changes in the laws in Spain in relation to this matter, new technology arises to supply this demand. Currently, this demand is covered mostly by photovoltaic panels, since there is still much room for improvement in the field of wind energy for domestic use.

Due to the lack of a generalization of this last type of energy as a reference of efficiency and comfort for the domestic consumer, the central idea of this project arises: the "Smart Wind Turbine" or "Smart AG". This is a prototype of an Darrieus vertical axis wind turbine designed by the supervisor of this project: Juan Romeo Granados. It has five blades that rotate around a mobile axis, which would be connected to an electric motor. At the same time, it is thought that in the future the wind turbine will have an electronic system that allows obtaining maximum efficiency for energy extraction.

In order to fulfill this last objective, it is necessary to have the appropriate technologies and tools for it. In this project the necessary systems are developed to:

- Record the switching time values of an anemometer used to measure the linear wind speed.
- Record the switching time values of a Hall or Holzer effect sensor used to measure the angular speed of the turbine.
- Activate an emergency turbine braking system if the specified speed limit is exceeded.
- Calculate the position of the center of mass of the turbine wing profile.
- Calculate the moment of inertia of the turbine.
- Have the wind and turbine speed and acceleration values available for each moment of time, being able to display them graphically.
- Calculate the torque and mechanical power on the turbine shaft, being able to display them graphically.
- Estimate the turbine loss coefficients.
- Draw a prototype of the optimum turbine tracking curve, necessary for the use of vector control devices. To draw the real one, it would be necessary to replace the mechanical power on the shaft with the electrical power generated at each moment by a motor connected to the turbine.

Methodology

Arduino has been used to design the first three functions. These have been installed on an Arduino Uno board to which an anemometer, a Hall sensor, a sevomotor and a *bluetooth* module have been connected. The anemometer used has three cups and switches twice per lap. The Hall effect sensor is attached to the turbine casing and detects a switch every time a blade passes near it. Each blade has a magnet attached to its central point. The servomotor is installed in a brake system designed by the project supervisor. When the wind or turbine speed exceeds a previously specified limit, the servomotor is activated. The *bluetooth* module is used to send the switching time data of the anemometer and the Hall sensor to a computer for further analysis. Everything related to sending data through this system has been designed by the student Javier Colinas Cano in his respective Final Degree Project: "Measurement, transmission and data analysis of a vertical axis wind turbine for domestic use".

Matlab has been used to design the rest of the functions. The calculation of the position of the center of masses of the wing profile, as well as the calculation of the moment of inertia of the turbine, has been carried out by discretizing the wing profile area and operating according to the position of each point of the system. When it comes to the moment of inertia, it is crucial to add the moments of the other turbine elements, in addition to the ones of the blades. Obtaining speeds for each instant of time is achieved thanks to the switching time values sent to the computer by the student Javier Colinas Cano. With these vectors, the speed values for these switching times are calculated using time differentials. Finally, the speed values are interpolated for each instant, having, in this case, a resolution of [ms]. Acceleration can be calculated using speed values. Acceleration, in addition to the moment of inertia of the turbine, allows obtaining the torque and the mechanical power for each moment of time.

Results

The programs presented in this project together with the devices indicated therein achieve the proposed objectives. However, there were sudden failures in data collection initially, causing some of the Hall sensor commutations not to be detected at high turbine speeds. This was solved by creating different modes of use in the Matlab program that spot the error and correct it by an approximation. In addition, the method used to measure the speed of the turbine, using a magnet on each blade, produced errors of measurement of the acceleration, since the magnets were not equidistant. To solve this, two additional modes of use were added to the program: one of them measured the time variations of each of the five magnets in relation to itself and the other suppressed all the magnets except one, since this prevented errors in the detection of the switching to a greater extent.

Regarding the loss coefficients estimation program, it performs the calculations based on a turbine braking curve where no external torque is applied to it. Depending on how many points of the curve it is decided to take, an equivalent number of coefficients is calculated, their order being ascending. Theoretically, the higher the order of the coefficient, the lower its value in this type of movement, being all of them positive and reaching a point from which they are negligible. However, the turbine presents vibrations during its movement, so when applying the program, coefficients that model these oscillations are calculated, being them inseparable from those that would exclusively model the losses.

Finally, the program for obtaining the prototype of the optimum turbine tracking curve functions properly. Even so, it would be necessary to install a wattmeter at the output of an electric motor connected to the turbine axis in order to obtain the real curve to be programmed in the pertinent vector control devices.

Conclusions

This project presents a series of useful tools, procedures and recommendations aimed at facing the first steps of the development of a wind turbine of these characteristics. Throughout it, errors that may have occurred are detailed, as well as their solutions.

After this project, comes the improvement of the turbine design based on the results obtained thanks to the execution of the programs shown here. For obtaining electrical energy and its subsequent transmission to the grid, devices such as an electric motor, power converters or a gearbox, if necessary, must also be chosen and installed.

A mis abuelos
Juliana y José Antonio.

Agradezco a ***LaNaveMadrid.com***
por facilitar sus instalaciones y recursos
para la elaboración de este proyecto.

«Mire vuestra merced
que aquellos que allí se parecen no son gigantes,
sino molinos de viento,
y lo que en ellos parecen brazos son las aspas,
que, volteadas al viento,
hacen andar la piedra del molino.»
SANCHO PANZA

Índice

Índice de figuras

Índice de tablas

Índice de extractos de código

Acrónimos

ICAI	Insitituto Católico de Artes e Industrias
PFC	Proyecto Fin de Carrera
UE	Unión Europea
IDAE	Instituto para la Diversificación y Ahorro de la Energía
EnerAgen	Asociación de Agencias Españolas de Gestión de la Energía
PWM	*Pulse Width Modulatión* (Modulación de ancho de pulso)
ONU	Organización de las Naciones Unidas

Símbolos

$[A]$	Matriz de coeficientes de un sistema de ecuaciones lineales
$\alpha_{subindice}$	Aceleración angular del elemento indicado en el subíndice
$B_{subindice}$	Coeficiente de pérdidas de la turbina del orden indicado en el subíndice
$\vec{b}$	Vector de términos independientes de un sistema de ecuaciones lineales
ca	Factor de cazoleta del anemómetro
$D_{subindice}$	Distancia entre los dos puntos indicados en el subíndice
d	Diferencial de
E_c	Energía cinética
$F_{subindice}$	Fuerza que genera el elemento del subíndice
$I_{z\ subindice}$	Momento de inercia en el eje z del elemento del subíndice
J	Momento de inercia de la turbina
ka	Constante de conversión de tiempos de conmutación a velocidad en el anemómetro
kh	Constante de conversión de tiempos de conmutación a velocidad en el sensor Hall
L	Momento angular
$M_{subindice}$	Momento de fuerza generado por el elemento del subíndice
$m_{subindice}$	Masa del elemento del subíndice
$n_{subindice}$	Número de los elementos especificados por el subíndice
$P_{subindice}$	Potencia transmitida por o al elemento del subíndice
p	Cantidad de movimiento o momento lineal
$R_{subindice}$	Radio del elemento del subíndice
t	Tiempo
θ	Posición angular
$v_{subindice}$	Velocidad lineal del elemento del subíndice
$x_{subindice}$	Posición en el eje x del elemento del subíndice
$\vec{x}$	Vector de incógnitas de un sistema de ecuaciones lineales
$y_{subindice}$	Posición en el eje y del elemento del subíndice
$\omega_{subindice}$	Aceleración angular del elemento indicado en el subíndice

PARTE I

MEMORIA

Capítulo 1

Introducción

ESTE proyecto se centrará en los primeros pasos del desarrollo de un prototipo de aerogenerador eléctrico para uso doméstico, concretamente de la parte dedicada al muestreo de las variables relacionadas con la velocidad del viento y de las palas del generador; esto se realizará con el objetivo de obtener un modelo fiable del mismo que permita la toma de decisiones orientadas a su producción y comercialización. El diseño físico del prototipo así como la producción de este ha sido realizada por el director del proyecto: Juan Romeo Granados; la transmisión de datos desde los sensores hasta el ordenador de trabajo, así como algunos aspectos relacionados con el control del prototipo han sido realizados por el alumno Javier Colinas Cano, el cual presenta a su vez el PFC: **«Medición, transmisión y análisis de datos de un aerogenerador de eje vertical para uso doméstico»**.

En este capítulo se resumirán las condiciones actuales de la producción y el consumo de energía eléctrica en relación a la modalidad de «autoconsumo». Para ello se aportarán datos clave en relación al mercado eléctrico y se comentarán las leyes relacionadas con este ámbito. Finalmente, se presentará la idea del prototipo de aerogenerador.

Como el proyecto se encontrará enfocado inicialmente en el consumo doméstico con una distribución del producto a pequeña escala en viviendas unifamiliares se ha elegido contextualizar la motivación del mismo a la situación energética en España durante los últimos veinte años.

1.1. Contexto actual del consumo doméstico de electricidad en España

Actualmente, el sistema eléctrico español cuenta aproximadamente con $104.094\,\mathrm{MW}$ de potencia instalada, de los cuales el 46.70 % está formado por energías renovables. Sin embargo, el porcentaje empleado de las mismas a finales del año 2018 fue del 38.44 %; esto se debe a las variaciones de disponibilidad y precios de las distintas fuentes de energía. Aun así la tendencia general en los últimos diez años ha sido aumentar la potencia renovable instalada y generada, destacando la consolidación de la energía eólica como la segunda fuente de generación más empleada del total de la matriz energética. Esto ha tenido una influencia directa en el descenso de los niveles de emisiones de CO_2 derivados de la producción de electricidad, siendo actualmente entorno al 60 % de la energía eléctrica producida en España libre de emisiones. [1]

Balance de potencia eléctrica instalada a 31.12.2018. Sistema eléctrico nacional

	Sistema peninsular		Sistemas no peninsulares		Total nacional	
	MW	%18/17	MW	%18/17	MW	%18/17
Hidráulica	17.047	0,1	2	0,0	17.049	0,1
Bombeo puro	3.329	0,0	-	-	3.329	0,0
Nuclear	7.117	0,0	-	-	7.117	0,0
Carbón	9.562	0,3	468	0,0	10.030	0,3
Fuel/gas	0	-	2.490	0,0	2.490	0,0
Ciclo combinado	24.562	-1,5	1.722	0,0	26.284	-1,4
Hidroeólica	-	-	11	0,0	11	0,0
Eólica	23.091	0,7	416	97,7	23.507	1,6
Solar fotovoltaica	4.466	0,6	248	0,2	4.714	0,5
Solar térmica	2.304	0,0	-	-	2.304	0,0
Otras renovables [1]	859	0,6	6	0,0	865	0,6
Cogeneración	5.730	-1,3	10	0,0	5.741	-1,3
Residuos no renovables	452	-1,4	38	0,0	491	-1,3
Residuos renovables	123	0,0	38	0,0	162	0,0
Total	**98.643**	**-0,2**	**5.452**	**3,9**	**104.094**	**0,0**

[1] Incluye biogás, biomasa, hidráulica marina y geotérmica.

Figura 1. Balance de potencia eléctrica instalada en España en 2018 [1]

Balance de energía eléctrica nacional [1]

	Sistema peninsular		Sistemas no peninsulares		Total nacional	
	GWh	%18/17	GWh	%18/17	GWh	%18/17
Hidráulica	34.103	84,9	3	0,1	34.106	84,9
Turbinación bombeo [2]	2.009	-10,7	-	-	2.009	-10,7
Nuclear	53.198	-4,2	-	-	53.198	-4,2
Carbón	34.882	-17,8	2.392	-7,9	37.274	-17,2
Fuel/gas [3]	-	-	6.683	-4,5	6.683	-4,5
Ciclo combinado [4]	26.403	-21,5	3.642	6,5	30.044	-18,9
Hidroeólica	-	-	24	16,9	24	16,9
Eólica	48.946	3,0	625	56,6	49.570	3,5
Solar fotovoltaica	7.374	-7,8	385	-3,1	7.759	-7,6
Solar térmica	4.424	-17,3	-	-	4.424	-17,3
Otras renovables [5]	3.547	-1,5	10	-8,3	3.557	-1,5
Cogeneración	28.981	2,9	35	-3,5	29.016	2,8
Residuos no renovables	2.294	-6,7	141	-5,2	2.435	-6,6
Residuos renovables	733	0,7	141	-5,2	874	-0,3
Generación	**246.893**	**-0,5**	**14.081**	**-0,7**	**260.974**	**-0,5**
Consumos en bombeo	-3.198	-11,3	-	-	-3.198	-11,3
Enlace Península-Baleares [6]	-1.233	4,6	1.233	4,6	0	-
Saldo intercambios internacionales físicos [7]	11.102	21,1	-	-	11.102	21,1
Demanda (b.c.)	**253.563**	**0,4**	**15.314**	**-0,3**	**268.877**	**0,4**

[1] Asignación de unidades de producción según combustible principal.
[2] Turbinación de bombeo puro + estimación de turbinación de bombeo mixto.
[3] En el sistema eléctrico de Baleares se incluye la generación con grupos auxiliares.
[4] Incluye funcionamiento en ciclo abierto. En el sistema eléctrico de Canarias utiliza gasoil como combustible principal.
[5] Incluye biogás, biomasa, hidráulica marina y geotérmica.
[6] Valor positivo: entrada de energía en el sistema; valor negativo: salida de energía del sistema.
[7] Valor positivo: saldo importador; valor negativo: saldo exportador. Los valores de incrementos no se calculan cuando los saldos de intercambios tienen distinto signo.

Figura 2. Balance de energía eléctrica consumida en España en 2018 [1]

La aplicación de las nuevas tecnologías basadas en fuentes renovables y/o libres de emisiones de gases de efecto invernadero ha supuesto un impacto positivo tanto para el medioambiente como para la independencia energética del país. Pese a considerarse esto como ventajas directas para el consumidor doméstico, la aparición de estos avances energéticos no ha repercutido en una bajada sustancial de los precios, siendo España el quinto país, miembro de la UE, con el precio de la electricidad más alto en el último año y manteniéndose en una situación similar en los últimos tres; también es necesario tener en cuenta la subida generalizada de los precios en los países comunitarios durante los últimos doce años. [2]

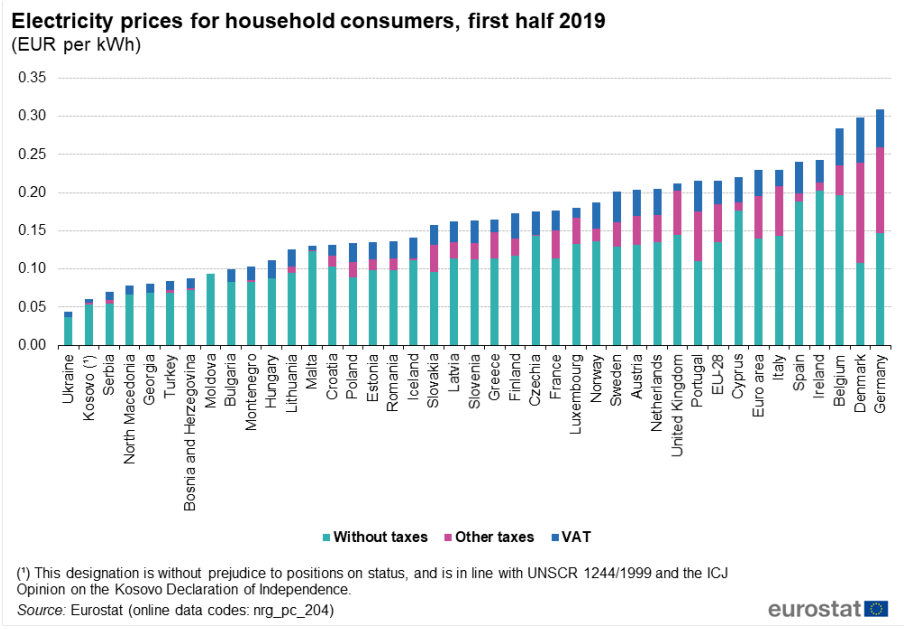


Figura 3. Precio de la electricidad para los consumidores domésticos durante la primera mitad de 2019 en los países de la UE [2]

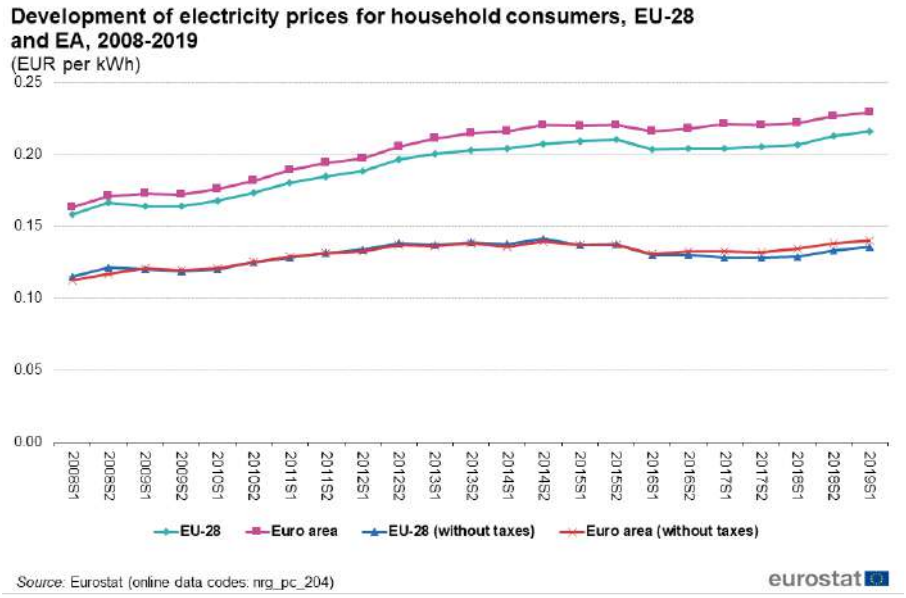


Figura 4. Evolución del precio de la electricidad para los consumidores domésticos entre 2008 y 2019 en los países de la UE [2]

Es precisamente esta subida de precios, junto con las nuevas posibilidades de generación eléctrica a nivel doméstico, lo que ha propiciado un aumento en el número de hogares europeos que recientemente cuentan con equipos capaces de producir electricidad para consumo propio. Se estima que en España durante el año 2011 un 11.2 % de los hogares disponía de algún tipo de fuente de energía renovable, desglosado como un 22 % de las viviendas unifamiliares y un 6 % de los pisos [3]; a su vez los datos más recientes muestran que el 18.9 % de la energía total consumida en el sector residencial español provenía directamente de fuentes renovables en el año 2017 [4].

Share of fuels in the final energy consumption in the residential sector, 2017
(%)

	Electricity	Derived Heat	Gas	Solid fuels	Oil & petroleum products	Renewables and Wastes
EU-28	**24.1**	**7.8**	**36.0**	**3.3**	**11.2**	**17.5**
Belgium	19.2	0.0	40.8	1.0	31.0	8.0
Bulgaria	41.3	14.7	2.9	6.8	1.1	33.2
Czechia	18.2	14.8	27.9	11.7	0.6	26.8
Denmark	18.7	36.8	13.0	0.0	4.7	26.8
Germany	19.5	7.8	38.2	0.9	20.1	13.5
Estonia	17.7	33.8	5.9	0.2	1.0	41.4
Ireland	26.5	0.0	21.5	12.4	38.1	1.5
Greece	38.2	1.2	8.2	0.1	28.2	24.1
Spain	39.0	0.0	24.3	0.5	17.3	18.9
France	33.7	3.3	27.9	0.1	13.1	21.9
Croatia	22.7	4.9	20.2	0.2	5.8	46.3
Italy	17.1	2.8	52.5	0.0	6.3	21.3
Cyprus	42.5	0.0	0.0	0.0	36.3	21.2
Latvia	11.9	31.1	9.3	0.8	4.6	42.2
Lithuania	16.8	32.6	10.6	3.9	3.8	32.3
Luxembourg	15.3	0.0	45.5	0.1	32.8	6.3
Hungary	15.4	7.9	47.2	2.2	1.2	26.1
Malta	70.0	0.0	0.0	0.0	17.4	12.6
Netherlands	20.0	2.9	71.0	0.0	0.4	5.6
Austria	22.5	11.8	22.0	0.3	15.8	27.6
Poland	12.6	19.6	18.2	32.9	3.0	13.7
Portugal	42.1	0.0	9.7	0.0	16.4	31.8
Romania	14.1	10.5	31.5	0.4	3.7	39.8
Slovenia	25.5	7.1	10.6	0.0	12.0	44.8
Slovakia	20.0	21.4	54.5	1.6	0.4	1.9
Finland	33.6	28.8	0.5	0.1	6.2	30.9
Sweden	51.7	34.8	0.5	0.0	0.3	12.7
United Kingdom	24.4	0.7	62.0	1.4	6.3	5.1
Iceland	16.7	79.6	0.0	0.0	0.9	2.8
Norway	83.3	2.7	0.2	0.0	1.6	12.2
Montenegro	42.0	0.0	0.0	0.9	0.5	56.7
North Macedonia	50.3	6.6	0.0	0.3	2.3	40.5
Albania	51.2	0.0	0.0	0.0	21.8	27.0
Serbia	41.6	13.8	6.7	8.0	1.7	28.2
Turkey	21.1	0.0	50.2	8.9	1.1	18.8
Bosnia and Herzegovina	39.1	9.1	3.6	10.0	4.3	33.9
Kosovo *	34.5	1.7	0.0	1.8	1.9	60.2
Moldova	10.7	8.8	17.2	4.0	4.4	54.9
Ukraine	18.3	16.1	53.7	1.3	0.3	10.2
Georgia	15.9	0.0	55.5	0.0	1.1	27.4

*This designation is without prejudice to positions on status, and is in line with UNSCR 1244 and the ICJ Opinion on the Kosovo declaration of independence.

Source: Eurostat (online data code: nrg_bal_c)

eurostat

Figura 5. Consumo energético doméstico en los países de la UE en 2017 según su fuente de origen [4]

Estos datos muestran cómo la generación mediante estas tecnologías, capaces de suministrar energía directamente a las viviendas, es una opción actualmente empleada en España, aunque sin llegar a los niveles de producción de otros países europeos que a su vez cuentan con unos precios para la electricidad más baratos. Esto otorga todavía un margen de mejora para la cantidad de energía generada en el ámbito doméstico mediante fuentes renovables que no se había dado hasta ahora por los motivos que se expondrán en la siguiente sección.

1.2. El futuro del autoconsumo doméstico

Se denomina «autoconsumo eléctrico» a la capacidad de emplear energía eléctrica proveniente de instalaciones de producción cercanas y asociadas al consumidor. Esta definición se utilizó por primera vez en España de manera oficial en el artículo 9.1 de la **Ley 24/2013, de 26 de diciembre, del Sector Eléctrico** [5]:

«1. A los efectos de esta ley, se entenderá por autoconsumo el consumo de energía eléctrica proveniente de instalaciones de generación conectadas en el interior de una red de un consumidor o a través de una línea directa de energía eléctrica asociadas a un consumidor.

Se distinguen las siguientes modalidades de autoconsumo:

a) Modalidades de suministro con autoconsumo sin excedentes. Cuando los dispositivos físicos instalados impidan la inyección alguna de energía excedentaria a la red de transporte o distribución. En este caso existirá un único tipo de sujeto de los previstos en el artículo 6, que será el sujeto consumidor.

b) Modalidades de suministro con autoconsumo con excedentes. Cuando las instalaciones de generación puedan, además de suministrar energía para autoconsumo, inyectar energía excedentaria en las redes de transporte y distribución. En estos casos existirán dos tipos de sujetos de los previstos en el artículo 6, el sujeto consumidor y el productor.»

En este punto del artículo ya se puede observar como se hace una diferenciación basada en la posibilidad de exportar o no energía a la red de distribución o transporte. Esta diferenciación es la que continúa hasta ahora, a excepción de la modalidad individual única de autoconsumo, donde actualmente se añade la posibilidad de compartir entre varios consumidores el mismo dispositivo de producción.

Para poder comprender por qué en España no se ha producido un aumento significativo en el uso de estas tecnologías es necesario mencionar tanto la **Ley 24/2013, de 26 de diciembre** [5], donde se recogía la modalidad de autoconsumo, como el **Real Decreto 900/2015, de 9 de octubre** [6], donde se regulaban las distintas condiciones del mismo. El objetivo de esta sección no es realizar un informe detallado sobre las consecuencias de los aquí nombrados si no argumentar, empleando diversos estudios, por qué es muy probable que en los próximos años aumenten significativamente las inversiones en tecnologías para el autoconsumo basadas en energías renovables, a diferencia de lo que ocurría en el pasado, gracias al nuevo **Real Decreto-ley 15/2018, de 5 de octubre** [7], que derogó numerosos artículos relacionados con el autoconsumo del anterior Real Decreto mencionado, y el **Real Decreto 244/2019, de 5 de abril** [8], que completa a este último. Es decir, que se produzca un cambio de paradigma que haga muy rentable, tanto para el consumidor como para el productor de los dispositivos, la aplicación generalizada de estas nuevas tecnologías que todavía no han llegado a su máxima eficiencia, aunque ya cuenten con una implantación significativa en otros países de la UE.

El principal problema que se atribuía a la normativa empleada con anterioridad era la falta de coherencia con los objetivos medioambientales de desarrollo sostenible en materia energética de la UE puesto que, sin entrar de forma exhaustiva en la materia debido a la gran complejidad jurídica que presenta, esta agravaba a los productores de energía conectados a una red de distribuión o transporte con numerosos tributos de tipo fijo, como por ejemplo:

- Los **«Peajes de acceso a las redes de aplicación a las modalidades de autoconsumo»** (Artículo 16, [6]), los cuales son los costes por acceder a la red, aplicados tanto por consumir como por verter energía en ella, en caso de comercializarla. Están asociados tanto a la potencia contratada como a la energía intercambiada.

- Los **«Cargos asociados a los costes del sistema eléctrico»** (Artículo 17, [6]), destinados a cubrir los costes adicionales de producción en territorios no peninsulares y la deuda por el déficit de tarifa; así como a cubrir el régimen retributivo especial de la generación mediante cogeneración de alta eficiencia, residuos y energías renovables, independientemente de si la instalación de autoconsumo se basa en esta última tecnología.

- El denominado como **«Cargo por otros servicios del sistema»** (Artículo 18, [6]), también conocido como «costes de respaldo del sistema» o «impuesto al Sol», de forma coloquial e incorrecta. Destinado a equilibrar las posibles diferencias entre la oferta y demanda de energía eléctrica en tiempo real que pudiera ocasionar la implantación del autoconsumo generalizado en España.

Todos ellos, exceptuando a los dos últimos en las islas, eran de obligatorio pago en caso de encontrarse la instalación conectada a la red eléctrica, independientemente de si se empleaba o no. Es en el último donde se generaba la mayor controversia puesto que mediante un aumento de los costes fijos del autoconsumo se compensaba la bajada de costes variables que conllevaba el mismo, además de generarse una gran situación de incertidumbre en cuanto a su cuantía y la forma de cobrarlo, lo que ocasionó una reducción de incentivos para producir y consumir energía de esta manera frente a obtenerla directamente de la propia red, repercutiendo por tanto en las inversiones sobre la misma. [9]

Las nuevas leyes derogaron, entre otros, los artículos 17 y 18, además de realizar cambios sustanciales en la **Ley 24/2013, de 26 de diciembre, del Sector Eléctrico** [5]. Las medidas a destacar de la nueva normativa son entre otras [10]:

- Habilitar la figura del autoconsumo colectivo, como bien se indicó con anterioridad. Con esto se pretende impulsar el autoconsumo en comunidades de vecinos para conseguir una mejor eficiencia.

- Establecer los mecanismos de compensación pertinentes para los autoconsumidores tanto «sin excedentes» como «con excedentes».

- Reducir los trámites de forma general. Además existe una **«Guía profesional de tramitación del autoconsumo»** [11], desarrollada por el IDAE junto con EnerAgen, donde se indican los pasos a seguir para las distintas modalidades de autoconsumidor.

- Aumentar la potencia permitida de las instalaciones monofásicas de autoconsumo de $5\,\mathrm{kW}$ a $15\,\mathrm{kW}$, permitiendo con esto un mayor grado de libertad en los sistemas según las distintas necesidades del autoconsumidor.

- Simplificar requerimientos técnicos de las instalaciones, ya sea permitiendo el cambio de ubicación de contadores o reduciendo el número necesitado de los mismos en ciertos casos, reduciendo con esto los costes fijos de la inversión.

El conjunto de estas medidas otorga una mayor flexibilidad a la hora de instalar este tipo de tecnología, mejoras fiscales para el autoconsumo basado en fuentes de energía renovable y lo más importante: un mayor grado de certidumbre en cuanto a costes y beneficios asociados a

la Administración, permitiendo con ello un marco legal seguro, concreto y predecible que, si todo continúa acorde con estos nuevos parámetros, fomentará una mayor disposición por parte del consumidor a adoptar estos avances tecnológicos en su hogar y por parte del fabricante, a mejorarlos. [12]

Figura 6. Cuadro resumen de las modalidades y las diferentes posibilidades de autoconsumo [11]

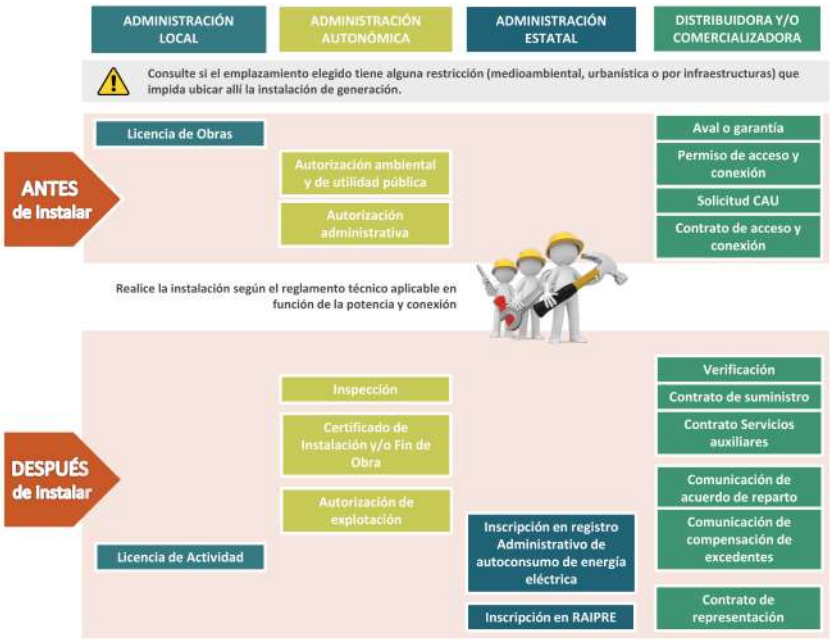


Figura 7. Resumen de las etapas de tramitación y organismos/entidades implicados [11]

1.3. Aerogenerador inteligente

Ante las nuevas perspectivas favorables a la instalación de equipos de autoconsumo de energía eléctrica en los hogares españoles surge el elemento central de este proyecto: el **«Aerogenerador Inteligente»** o **«Smart AG»**. Este dispositivo es un aerogenerador Darrieus de eje vertical y pequeño tamaño diseñado por el director del proyecto: Juan Romeo Granados. El mismo se encuentra actualmente en fase de prototipado y consta de:

- Una turbina de cinco palas, cada una de ellas formada por un perfil de ala desarrollado por el propio director. El vector normal a cada perfil de ala se encuentra en posición vertical y gira en torno al eje unido al mismo mediante dos barras paralelas entre sí.
- Una base circular sobre la que gira el eje unido a la misma a través de un rodamiento.
- Varios alerones fijos, aunque capaces de variar su ángulo de forma manual, colocados en vertical rodeando a la turbina. Los mismos permiten dirigir los caudales entrantes y salientes al generador de una manera más favorable a la rotación de las palas.
- Una tapa similar a la de la base que reposa sobre los alerones, así como sobre unas barras de aluminio de sección cuadrada cuya función es aportar rigidez a la estructura. Por el centro de la misma sobresale, a través de otro rodamiento, el eje de la turbina.
- Un compartimento cilíndrico, situado sobre el centro de la tapa superior, capaz de alojar un motor eléctrico de pequeño tamaño conectado al eje de la turbina.

La forma de la turbina otorga una mayor facilidad para apilar varias entre sí, permitiendo con ello incrementar la potencia generada sin aumentar el espacio ocupado por las mismas ni reducir el rendimiento de estas como consecuencia de situarlas con proximidad, disminuyendo con ello el área de incidencia del viento.

Las dimensiones exteriores del prototipo son $850\,\mathrm{mm}$ de altura y $780\,\mathrm{mm}$ de diámetro, encontrándose el mismo elevado $150\,\mathrm{mm}$ sobre el suelo gracias a unas ruedas con freno de bloqueo. Respecto a la turbina, esta cuenta con cinco palas de $710\,\mathrm{mm}$ de altura cuyo punto mas alejado del eje se encuentra a un radio de $265\,\mathrm{mm}$, las mismas cuentan con un área de $156\,200\,\mathrm{mm}^2$ en su base. El compartimento superior para el motor es de $170\,\mathrm{mm}$ de altura y $170\,\mathrm{mm}$ de diámetro, aunque es fácilmente reemplazable en caso de ser necesario.

Estas características hacen que el emplazamiento ideal para este sistema sea en zonas altas de viviendas unifamiliares, aunque en el futuro no se cierra la posibilidad a implementarse de manera conjunta en edificios de mayor tamaño.

El aerogenerador tendrá, acoplado al eje, un motor eléctrico cuyo propósito será transformar la energía mecánica del viento en electricidad; dependiendo de los resultados de este proyecto se podrá elegir un tipo de motor u otro. A su vez, de manera conjunta, se optará por uno de los distintos modelos de convertidores de potencia disponibles, además de las baterías para el almacenamiento de energía en caso de considerarse útiles; también será necesario diseñar la forma de conectar todo entre sí. Finalmente, será un microprocesador el encargado de controlar los parámetros del convertidor de potencia necesarios para una obtención óptima de energía en función de la velocidad de la turbina así como de la del viento.

Por otro lado, para el estudio de las variables relevantes que permitan modelar la turbina se han empleado un anemómetro y un sensor hall conectados a un microprocesador tipo Arduino,

los tres elegidos por el director del proyecto. Para detener la turbina en caso de sufrir una aceleración excesiva durante los ensayos pertinentes se ha instalado un sistema de frenado mecánico, diseñado por el director del proyecto, el cual es activado por un servomotor que también se encuentra controlado por el microprocesador. El funcionamiento de estos sistemas se explicará en los capítulos posteriores de esta memoria.

Otro punto importante para el correcto funcionamiento de todos los sistemas es la transmisión de datos desde el microprocesador, instalado en la carcasa de la turbina, hasta un ordenador capaz de gestionar los mismos; esta tarea ha sido realizada por el alumno Javier Colinas Cano en su respectivo proyecto ya mencionado con anterioridad.

Figura 8. Turbina del aerogenerador

Capítulo 2

Estado tecnológico actual

En este capitulo se describirán brevemente las tecnologías mediante las cuales el desarrollo de este proyecto puede llegar a ser viable actualmente.

2.1. Vectores energéticos para autoconsumo doméstico

Son denominados como vectores energéticos aquellos dispositivos o sustancias capaces de almacenar energía para su posterior uso controlado. Si se analiza el consumo energético en una vivienda media española en el año 2011, del cual se pueden extrapolar los datos aproximados a los últimos diez años, se puede observar como el 35.1 % del mismo procede de la energía eléctrica de la red; además, del consumo eléctrico total, un 41.3 % del gasto se corresponde con el de electrodomésticos que se encuentran permanentemente conectados y con un consumo constante [3]; es decir: más de la mitad del consumo eléctrico doméstico responde a variaciones a lo largo del día. Es este uno de los motivos, sumado a las variaciones del precio de la electricidad en el tiempo, por los que si se desea disponer de energía en cualquier momento, con independencia de la red, son necesarios los sistemas de almacenamiento de la misma.

Para el almacenamiento de energía eléctrica dentro del domicilio el sistema más empleado son las baterías. Su desarrollo se ha visto fuertemente impulsado en los últimos años gracias a la irrupción de la energía solar fotovoltaica así como el coche eléctrico. Los modelos que existen tienen diversas características y precios, ajustándose cada una a las distintas necesidades del usuario. Entre estos están: las baterías de plomo ácido, abiertas y cerradas, en donde las primeras requieren un mayor mantenimiento que las segundas; las de gel de sílice selladas, que no requieren mantenimiento pero son más delicadas, aunque funcionan mejor y su huella ecológica es menor; y las AGM, de fibra de vidrio, las cuales presentan una calidad superior pero son más costosas [13].

2.2. Aerogeneradores de uso doméstico

La mayoría de la energía obtenida mediante tecnologías de autoconsumo proviene actualmente de instalaciones solares fotovoltaicas, como consecuencia de su sencilla instalación, ya que solamente hacen falta las placas, un regulador, una batería y un inversor para transformar la corriente continua en alterna. Sin embargo, puede comprobarse con una simple búsqueda en internet como cada vez aparecen a la venta más sistemas para el autoconsumo basados en la energía eólica.

Los precios por kW de estos sistemas suelen rondar entre los 1000€ y los 2000€, a esto hay que añadirle la necesidad de rentabilizar la inversión y el inconveniente de que no todas las zonas geográficas disponen de caudales de viento suficientes para hacer esto posible [14]. A su vez, la mayoría de las turbinas del mercado son de eje horizontal y por ello suelen generar vibraciones elevadas así como ruidos. Son estos los motivos por los que todavía queda margen de mejora dentro de este mercado tanto a un nivel técnico como económico.

2.3. Motores eléctricos de baja tensión

Afortunadamente, tanto la calidad, como la eficiencia y el precio de los motores eléctricos se han visto mejorados durante los últimos años. Empresas como Siemens [15], ABB [16] o WEG [17] ofrecen numerosas alternativas para motores de baja tensión tanto asíncronos de jaula de ardilla o rotor bobinado, hasta síncronos con excitación independiente o de imanes permanentes. Son este tipo de dispositivos los que han de conectarse en el eje de la turbina para transformar la energía cinética del viento en energía eléctrica, mediante la velocidad que establece la frecuencia de la red eléctrica en conjunto al par mecánico que genera el viento en las palas de la turbina.

2.4. Electrónica de potencia

Los dispositivos cuyo funcionamiento depende de la electrónica de potencia son los encargados de convertir la frecuencia y la tensión de la máquina a las de la red. Son los llamados convertidores, los cuales, de manera adicional, pueden controlar desde la frecuencia de giro del rotor junto a la tensión entre los bornes del motor, en el caso de aplicar control de tensión y frecuencia, hasta el par mecánico que ofrecen en cada instante, en el caso de aplicar control vectorial.

Si se quiere obtener el máximo rendimiento de la turbina eólica el control a utilizar será el control vectorial. Será necesario cargar previamente en el convertidor la llamada «curva de seguimiento óptimo», de la que se explicará sus fundamentos en la subsección 3.2.2, la cual consigue que el convertidor ajuste la velocidad del motor que mueve a la turbina mediante la variación del par del mismo, en función del par al que la esté sometiendo el viento en cada instante, para obtener la máxima potencia disponible.

Ejemplos de este tipo de tecnología serían la gama de convertidores «SINAMICS» de Siemens [18] o los convertidores de la marca ABB [19].

2.5. Prototipado mediante fabricación aditiva

Gracias a la fabricación aditiva se han podido desarrollar numerosas partes del prototipo de la turbina, entre estas se encuentran los perfiles de ala y las juntas de unión entre varillas. Esta tecnología permite crear, con unos costes muy bajos de producción, piezas con una sola tirada. A su vez, la resistencia que ofrecen estas piezas es suficiente para los usos que se les ha dado en el prototipo.

2.6. Microprocesadores

Por último, este proyecto se ha realizado mediante el uso de un microprocesador para la medida de todas las magnitudes relevantes. Los mismos son dispositivos de pequeño tamaño y bajo precio que se pueden comprar en cualquier tienda de electrónica. En este caso el programa controlador empleado ha sido Arduino, el cual es un *software* de código libre que permite programar acciones en función de las entradas que reciban los sensores conectados a la placa. Actualmente existen numerosas marcas con diversos modelos, adaptados a distintos objetivos, que permiten el uso de este programa.

Capítulo 3

Modelo desarrollado

En este capítulo se explicarán todos los aspectos técnicos relevantes del proyecto así como todos los pasos seguidos en las diferentes fases del mismo. Las secciones de las que consta son un desarrollo orientado a la obtención del **momento de inercia de la turbina**, la **potencia mecánica instantánea en el eje**, los **coeficientes de pérdidas de la turbina asociados al par** y la **curva característica de potencia mecánica frente a velocidad angular**, empleando para ello los medios disponibles.

3.1. Objetivos y especificación

En esta sección se describen los cuatro objetivos principales a cumplir previamente para poder conseguir valores fiables que permitan la consecución de lo anteriormente comentado.

3.1.1. Toma de datos

El primer paso es conseguir medidas fiables tanto de la velocidad del viento cómo de la velocidad angular de la turbina en cada instante de tiempo. Para ello es sabido que los valores en módulo[1] de velocidades lineales y angulares instantáneas son:

$$v = \frac{dx}{dt} \tag{1}$$

$$\omega = \frac{d\theta}{dt} \tag{2}$$

Los cuales se pueden relacionar utilizando la expresión:

$$\omega = \frac{v}{R} \tag{3}$$

Es decir, para un instante dado de tiempo dt la velocidad angular de la turbina se mide mediante las variaciones en su ángulo de posición θ; a su vez la velocidad lineal del viento se puede relacionar con una velocidad angular a través de un radio de valor R que rote alrededor de un centro fijo. Sin embargo es necesario tener en cuenta que el aire, al igual que cualquier elemento en movimiento que posea masa dentro de un sistema, cuenta con una energía cinética,

[1]Independientemente de si se trata de magnitudes vectoriales o escalares, en esta memoria solamente se hará referencia al módulo de las mismas. El objetivo de esto es simplificar conceptos, puesto que los algoritmos empleados no almacenan las direcciones de las mismas.

siendo es esta la que se transmite, sujeta a pérdidas, al resto de elementos del sistema con los que interacciona, dependiendo esta transmisión del momento lineal, el cual, en ausencia de la acción de fuerzas externas, es constante.

$$E_c = \frac{1}{2} \cdot m \cdot v^2 \tag{4}$$

$$p = m \cdot v \tag{5}$$

$$\sum F_{ext} = 0 \Leftrightarrow p = cte = m_1 \cdot v_1 = m_2 \cdot v_2 \Rightarrow v_2 = \frac{m_1}{m_2} \cdot v_1 \tag{6}$$

Es por esto por lo que la velocidad del viento no se podrá calcular con una precisión aceptable a partir de la velocidad angular de la turbina, si no que será necesario disponer de un dispositivo con las menores pérdidas de energía posibles debidas a fricción y una masa lo suficientemente similar a la del caudal de viento que interactúa con él, es decir: un anemómetro.

Por lo tanto, serán necesarios dos dispositivos distintos: uno para medir la velocidad angular de la turbina y otro para medir la velocidad lineal del viento a través de una velocidad angular y un radio dados.

3.1.1.1. Tecnología empleada

Para medir la velocidad angular de la turbina el director del proyecto optó por emplear un **sensor magnético de efecto Hall o Holzer VMA313**. Este sensor cuenta con tres pines de conexión: uno a tierra, otro a una fuente de tensión continua de $5\,\mathrm{V}$ y otro empleado para emitir una señal continua. Cuando el sensor se encuentra alimentado a su tensión nominal este emite dicha señal hasta que detecta un campo magnético mayor o igual a $3\,\mathrm{mT}$, en ese momento la señal se vuelve nula hasta que la magnitud del campo se vuelve inferior a $1\,\mathrm{mT}$.

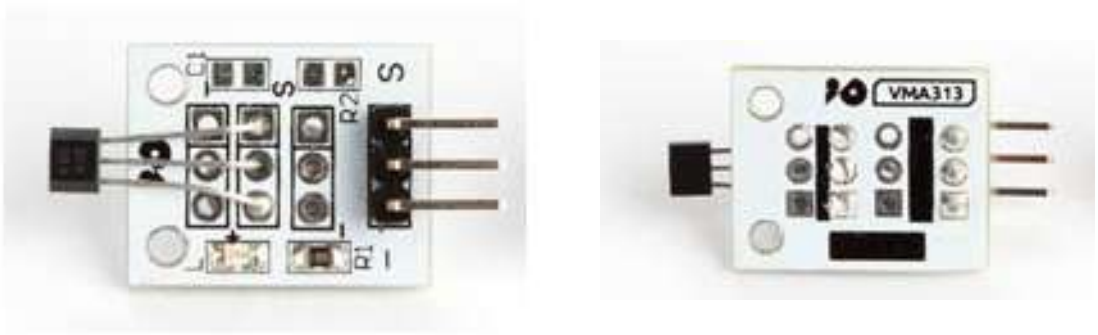


Figura 9. Sensor magnético de efecto Hall o Holzer VMA313

Para medir la velocidad del viento el director del proyecto escogió un **anemómetro para estación meteorológica N25FR WH1080**. Este es un dispositivo formado por tres palas semiesféricas huecas que giran simultáneamente en torno a un eje central unidas, cada una de ellas, mediante una varilla. El radio desde el punto central del eje hasta el centro de cada pala es es de $70\,\mathrm{mm}$. En el interior del anemómetro se aloja un imán que, durante su rotación simultánea al eje, abre y cierra un circuito eléctrico, el cual sale desde la base a través de dos cables que van a parar a un conector **RJ11** macho.

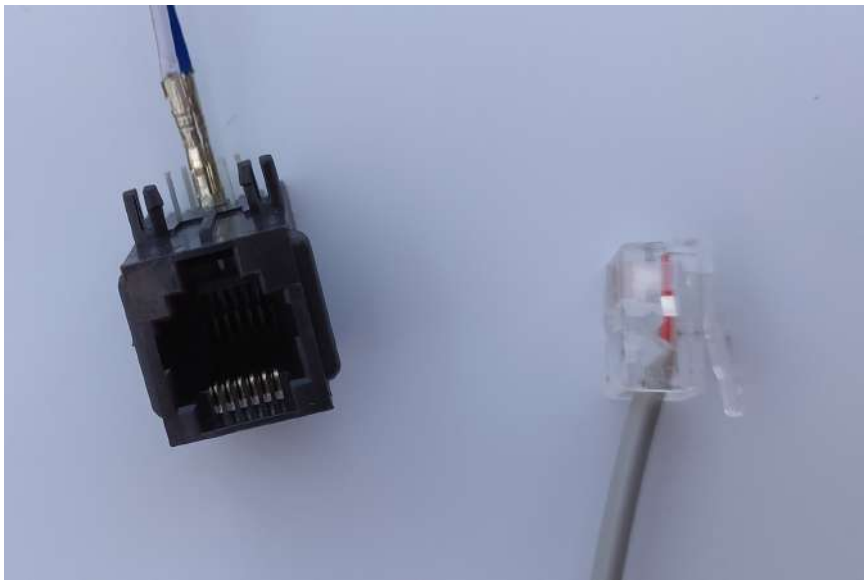


Figura 10. Conectores RJ11 hembra y macho

La hoja de características del mismo indica que para una velocidad del viento de $2.4\,\mathrm{km\,h^{-1}}$ se produce en el circuito una conmutación cada segundo, sin embargo se consideró oportuno comprobar esto de manera empírica; a la conclusión a la que se llegó fue que pasado un cuarto de vuelta el circuito se abría y permanecía así otro cuarto de vuelta hasta volverse a cerrar, la siguiente media vuelta se repetía el mismo proceso. Es decir, cada media vuelta se produce una conmutación. Como la conversión de la velocidad angular del anemómetro a la velocidad lineal en el centro de las semiesferas de las palas no se traduce en la velocidad lineal del viento según la hoja de características, es necesario definir una constante de conversión, a la que se ha denominado como **(ca)**.

$$\frac{2{,}4[\mathrm{km}]}{[\mathrm{h}]} \equiv \frac{1[conmutacion]}{[\mathrm{s}]} \tag{7}$$

$$w_{anemometro} = \frac{1[conmutacion]}{[\mathrm{s}]} = \frac{1[conmutacion]\cdot 1[vuelta]\cdot 2\pi[\mathrm{rad}]}{[\mathrm{s}]\cdot 2[conmutacion]\cdot[vuelta]} = \frac{\pi[\mathrm{rad}]}{[\mathrm{s}]} \tag{8}$$

$$v_{viento} = \frac{2{,}4[\mathrm{km}]}{[\mathrm{h}]} = \frac{2{,}4[\mathrm{km}]\cdot 1000[\mathrm{m}]\cdot[\mathrm{h}]}{[\mathrm{h}]\cdot[\mathrm{km}]\cdot 3600[\mathrm{s}]} = \frac{2[\mathrm{m}]}{3[\mathrm{s}]} \tag{9}$$

Combinando las tres expresiones anteriores se obtiene que:

$$\frac{2{,}4[\mathrm{km}]}{[\mathrm{hour}]} \equiv \frac{1[conmutacion]}{[\mathrm{s}]} \Rightarrow ca = \frac{v_{viento}}{R_{pala\ anemometro}\cdot w_{anemometro}} = \frac{\frac{2}{3}}{0{,}07\cdot\pi} = \frac{200}{21\pi} \tag{10}$$

El significado físico de esta constante implica que a la velocidad lineal del viento que mediría el anemómetro, teniendo en cuenta su radio y el número de conmutaciones por vuelta, habría que multiplicarla por **(ca)** para obtener la velocidad real. En otros trabajos se denomina a este valor como «factor de la cazoleta del anemómetro», indicándose el mismo mediante la letra **(k)**. El mismo se puede obtener mediante cálculo numérico [20] o de manera empírica para conseguir una mayor precisión, sin embargo, en este proyecto se ha decidido emplear el que el

fabricante indica de forma indirecta, puesto que se considera suficientemente preciso, habiéndose comentado lo anterior con una intención explicativa.

Figura 11. Anemómetro para estación meteorológica N25FR WH1080 con conector RJ11

El dispositivo elegido por el director del proyecto para la recogida de los datos fue una **placa Arduino Uno Rev3**. La misma se conecta a un ordenador mediante un cable USB a través del que se carga un programa de control creado mediante el software libre **Arduino**. Es este programa el que identifica los tiempos de conmutación del sensor hall y del anemómetro, los cuales han de encontrarse previamente conectados a la placa de la manera que se indicará en la siguiente subsubsección. Una vez cargado el programa la placa puede ser alimentada, por el propio cable USB, por una batería externa de $9\,\mathrm{V}$ o por un cable de alimentación a tensiones entre $6\,\mathrm{V}$ y $20\,\mathrm{V}$.

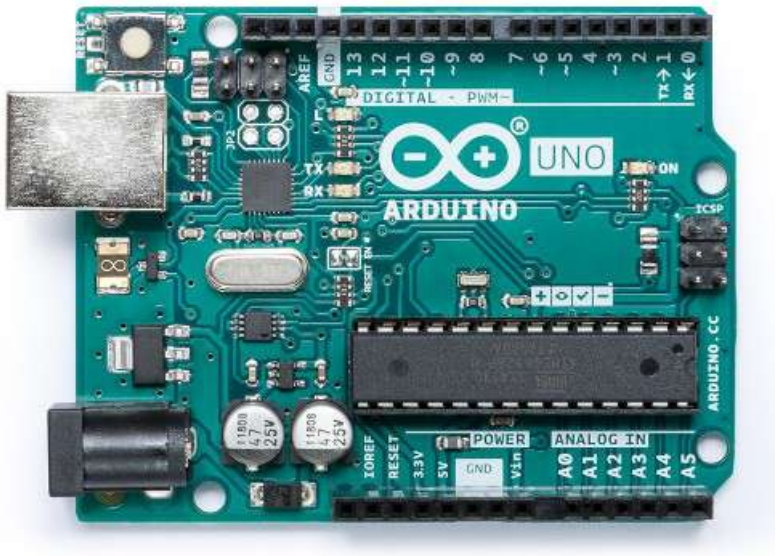


Figura 12. Placa Arduino Uno Rev3

3.1.1.2. Instalación de los sensores y esquemas de conexión

Una vez dispuesta la tecnología mencionada con anterioridad, se procede a conectar entre sí los distintos elementos tal y como se indica en el siguiente esquema:

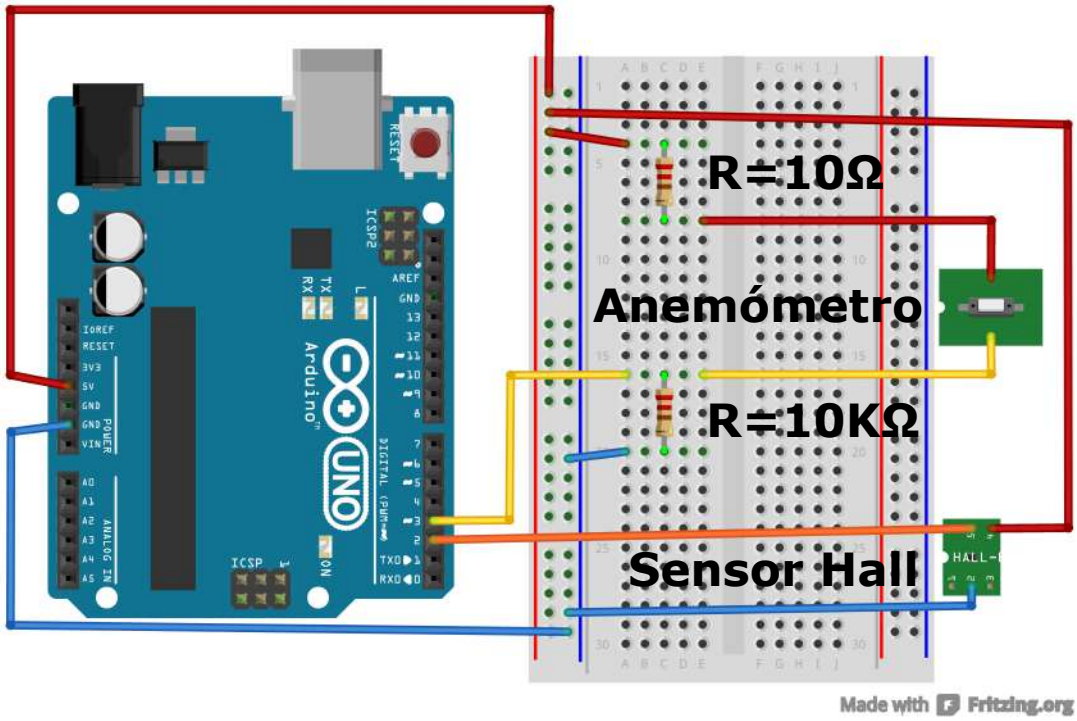


Figura 13. Esquema de conexión del anemómetro y el sensor magnético de efecto Hall o Holzer sobre la placa Arduino

Como bien se puede observar, para el correcto funcionamiento del dispositivo, es necesario emplear un divisor de tensión formado por una resistencia de $10\,\Omega$ y otra de $10\,\mathrm{k}\Omega$. El valor de estas resistencias no tiene por que ser exactamente este, aun así es necesario que exista, por lo menos, una diferencia de dos órdenes de magnitud entre las dos para que no se produzcan errores en la toma de la señal del anemómetro; en este caso la diferencia es de tres órdenes de magnitud. Se han escogido estas resistencias puesto que eran las que se encontraban disponibles.

El funcionamiento del anemómetro se basa en el divisor de tensión ya mencionado. Sobre este se genera, mediante la placa Arduino, una diferencia de tensión de $5\,\mathrm{V}$ de tal manera que la caída de tensión en la resistencia más grande sea lo más parecida a este valor. Esta resistencia se conecta entre el pin a tierra y el pin número 3 de la placa. La otra resistencia se conecta entre la salida de tensión del Arduino y el anemómetro. Finalmente, el pin restante del anemómetro se conecta al mismo nudo de la resistencia y el pin 3. De este modo cada vez que el anemómetro da un cuarto de vuelta se cierra o se abre el circuito, lo cual produce que el pin número 3 detecte una variación de tensión de aproximadamente $5\,\mathrm{V}$.

Para acoplar el anemómetro al sistema descrito anteriormente es necesario utilizar un conector hembra de tipo RJ11 al que se conectarán, en los dos pines centrales, dos de los cables usados para el prototipo previamente crimpados. Una vez conectado el anemómetro se ubica el mismo

sobre el compartimento superior del aerogenerador de tal manera que el viento incida sobre el dispositivo con independencia de su dirección, tal y cómo se muestra en la imagen.

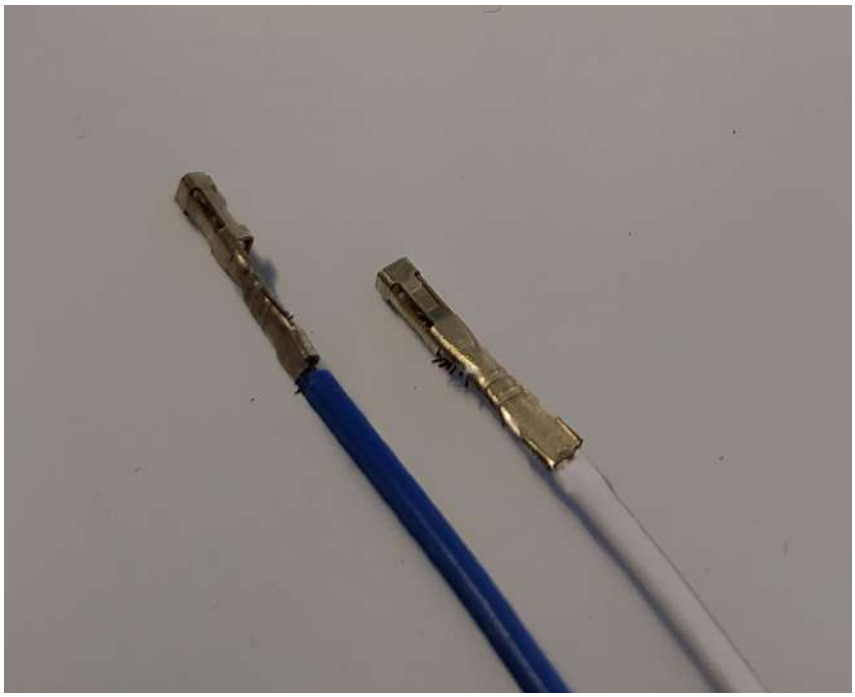


Figura 14. Cables crimpados

Figura 15. Anemómetro instalado de manera provisional

El sensor Hall se encuentra alimentado con los $5\,\mathrm{V}$ que es capaz de suministrar la placa. El pin de señal del mismo se conecta con el pin número 2 del Arduino. A su vez, el sensor se engancha mediante bridas a cualquiera de las barras de sección cuadrada de la estructura de la turbina. Estas barras se encuentran lo más cerca posible de las palas, de tal manera que, cuando las mismas giran, activan el sensor mediante una variación en el flujo magnético que atraviesa al mismo, empleando para ello unos imanes circulares de ferrita pegados a las palas. Cada pala

cuenta con un imán situado en la parte superior de la misma orientado de manera colineal al radio que une la pala con el eje. El funcionamiento de este sistema sería similar al de un «codificador rotatorio» o «encoder».

Figura 16. Sensor de efecto Hall o Holzer instalado en la turbina junto al imán de una pala

3.1.1.3. Cálculos del microprocesador

Tras acoplar los dispositivos a la turbina, se instala el programa de toma de datos en la placa Arduino. El mismo consta de un bucle principal, el cual se ejecuta permanentemente una vez se encuentra conectada la alimentación a la placa, tomando los valores de tiempo de conmutación del anemómetro y del sensor hall. Para esto, al principio del programa[2], se definen diversas variables:

```
//Variables

//Sensor hall
int ah; // Comprobación de inicio de conmutación en el sensor hall
unsigned long toh=0, tfh, dth, kh=12000, vh, vhmax=200;
// Variables no empleadas
// unsigned long vh2, dvh, aah;

//Anemómetro
int aan; // Comprobación de inicio de conmutación en el anemómetro
unsigned long toa=0, tfa, dta, ka=666667, va, vamax=10000;
// Variables no empleadas
// unsigned long va2, dva, aa;
```

Código 1. Líneas 7-19 del «programa principal» de la placa Arduino (PARTE III, sección 1.1)

[2]Todos los código completos se encuentra en la «PARTE III: CÓDIGO FUENTE».

Las variables de tipo entero o «int» sirven para detectar si en cada iteración del bucle principal se acaba de producir una conmutación tanto en el sensor hall **(ah)** como en el anemómetro **(aan)** de manera independiente; las mismas adoptarán solamente valores de 0 o 1. Las variables de tipo «unsigned long» sirven para almacenar los valores de tiempo de conmutación de cada dispositivo y realizar los cálculos de velocidad pertinentes para poder usar el sitema de freno que se explicará en la siguiente subsección. Estas variables han de ser de este tipo ya que la función de la que dispone Arduino para almacenar valores de tiempo solamente es capaz de funcionar correctamente de esta manera; estos valores se guardan en $[\mathrm{ms}]$. Las constantes que se muestran **(kh)** y **(ka)** sirven para convertir los valores de los diferenciales de tiempo de cada dispositivo tras cada conmutación en valores de velocidad. En el caso de la turbina esta velocidad se encuentra referida en $[\mathrm{rpm}]$ y en el caso del anemómetro, en $[\mathrm{mm\,s^{-1}}]$[3].

El cálculo de la constante del sensor hall **(kh)** se ha realizado teniendo en cuenta los parámetros de la turbina, donde:

$$kh = \frac{60000[\mathrm{ms}]}{n_{palas\ turbina} \cdot [min]} = \frac{60000}{5} = 12000 \tag{11}$$

El cálculo de la constante del anemómetro **(ka)** se ha realizado teniendo en cuenta los parámetros del mismo, donde:

$$ka = \frac{ca \cdot 2\pi[\mathrm{rad}] \cdot R_{pala\ anemometro}[\mathrm{mm}] \cdot 1000[\mathrm{ms}]}{n_{conmutaciones\ anemometro} \cdot [\mathrm{s}]} = \frac{\frac{200}{21\pi} \cdot 2 \cdot \pi \cdot 70 \cdot 1000}{2} = 666667 \tag{12}$$

Los valores de velocidades máximas tanto de la turbina **(vhmax)** como del viento **(vamax)** se encuentran elegidos arbitrariamente puesto que a partir de esos valores la turbina se embala, pudiendo producirse fallos mecánicos en la misma. Finalmente, dentro de las variables no empleadas se encuentran las necesarias para calcular aceleraciones empleando el Arduino; esto no es necesario puesto que las mismas se calcularán de forma más precisa mediante Matlab.

Una vez definidas todas las variables necesarias se procede a asignar los pines de entrada del sensor Hall y del anemómetro en los números 2 y el 3 respectivamente. Para esto se introducen los siguientes comandos dentro de la sección de inicialización del código o **«void setup()»**:

```
pinMode(2, INPUT); // Asigna el pin 2 al sensor Hall
pinMode(3, INPUT); // Asigna el pin 3 al anemómetro
```

Código 2. Líneas 33-34 del «programa principal» de la placa Arduino (PARTE III, sección 1.1)

A continuación se ilustra el funcionamiento del sensor Hall dentro del bucle principal o **«void loop()»**. No se muestra el del anemómetro puesto que sus fundamentos son los mismos. Este bucle ejecuta los siguientes comandos si se cumplen las condiciones que en ellos se indican; para ello la placa Arduino se encuentra permanentemente realizando estas comprobaciones entre instantes de tiempo casi despreciables.

[3]Todos los valores, tanto los de las constantes como los de las magnitudes, han de encontrarse referidos en números enteros puesto que Arduino trunca los decimales a 0 en este caso

```
        if(digitalRead(2)==HIGH) // Si el sensor hall no detecta conmutación
        {
            ah=1;
            servo(); // Ejecuta la función de movimiento del freno
        }
```

Código 3. Líneas 51-55 del «programa principal» de la placa Arduino (PARTE III, sección 1.1)

En el extracto de código superior se observa como, si el pin referido al sensor Hall detecta que el mismo no se ha activado[4], pone la variable auxiliar entera **(ah)** en un valor de 1. A su vez, se ejecuta la función **«servo()»** empleada para activar el freno si fuera necesario.

```
        if(digitalRead(2)==LOW) // Si el sensor hall detecta conmutación
        {
            if(ah==1) // Si la conmutación es detectada justo después de la ausencia de
                conmutación del sensor hall
            {
                    tfh = millis(); // Guarda el tiempo de la conmutación actual del
                        sensor hall
                    dth = tfh-toh; // Calcula el diferencial de tiempo respecto a la
                        conmutación anterior del sensor hall
                    toh = millis(); // Guarda el tiempo de la conmutación actual para ser
                        empleado en la siguiente conmutación del sensor hall

                if(dth!=0) // Si el diferencial de tiempo del sensor hall no es 0
                {
                    vh = kh/dth; // Calcula la velocidad angular de la turbina
                    //dvh = vh-vh2;
                    //vh2 = vh;
                    //aah = dvh/dth;
                }
                else // Si el diferencial de tiempo del sensor hall es 0
                {
                    vh = 199; // Fija una velocidad de la turbina muy alta
                    Serial.print("error vh");  // Indica mediante un mensaje en el
                        "Serial" que se ha producido un error en el sensor hall
                    //dvh = vh-vh2;
                    //vh2 = vh;
                    //aah = 10e6;
                }

                // Indica en el "Serial" la velocidad angular de la turbina
                Serial.print("vh:");
                Serial.print(vh);
                Serial.println('\t');

                enviar(2); //Función de envío de datos (desarrollada por Javier Colinas
                    Cano)
            }

            ah=0; // Asegura que no se vuelva a calcular el diferencial de tiempo hasta la
                siguiente conmutación real del sensor hall

            servo(); // Ejecuta la función de movimiento del freno
        }
```

Código 4. Líneas 94-129 del «programa principal» de la placa Arduino (PARTE III, sección 1.1)

En el extracto de código superior, el cual se ejecuta si el sensor detecta el paso de un imán, se observa cómo aparece una condición de funcionamiento que solamente se activará si en la

[4]Es necesario recordar que el sensor Hall mantiene una señal contínua de 5 V hasta que es activado por el cambio de flujo magnético producido por el paso de un imán, en ese momento el valor de la señal se vuelve nulo.

iteración previa del bucle principal no se detectó una variación de flujo magnético. Para asegurar esto, al final del extracto se pone la variable **(ah)** en un valor de 0. Esto es así puesto que no se sabe el tiempo que la señal del sensor Hall puede permanecer inactiva y no se quiere que se produzcan repeticiones indeseadas al contabilizar cada conmutación.

Dentro de dicha condición se encuentra el algoritmo que calcula la velocidad de rotación de la turbina (en el caso del anemómetro, la velocidad lineal del viento). Para esto almacena en dos variables, **(tfh)** y **(toh)**, el tiempo de conmutación del dispositivo en [ms], empleando para ello la función **«millis()»**. Después calcula el valor de un diferencial de tiempo **(dth)** restando el valor de tiempo de la conmutación anterior **(toh)** al de la actual **(tfh)**. Una vez hecho esto se divide el valor de la constante del sensor hall **(kh)** entre dicho diferencial y se obtiene el valor de velocidad instantánea. De manera adicional, si el diferencial de tiempo resultante es 0 se fijará un valor de velocidad arbitrario muy alto para que no se produzcan errores de cálculo, dando un aviso de que esto se ha producido. Una vez calculado todo esto se procede a enviar los datos a un ordenador mediante la función **«enviar()»**, así como a activar el freno, si fuera necesario.

3.1.2. Frenado de la turbina

Una vez se dispone de los valores calculados de las velocidades del viento y la turbina, se puede comenzar a configurar el sistema de frenado de la misma. La motivación de este sistema reside en la necesidad de poder detenenr la rotación de la turbina de forma automática en caso de que una racha de viento muy fuerte la embale. Este es un sistema provisional aplicado en el prototipo para evitar que se produzcan daños a personas o en el aeorgenerador cuando se encuentra a la intemperie, bien sea debido a los fallos producidos por una velocidad muy elevada del mismo o por los intentos de detenerlo.

Como en las fases iniciales la turbina no se encuentra conectada a ninguna máquina eléctrica que pueda regular la velocidad de la misma, para hacer efectivo el frenado es necesario aplicar en el eje de la turbina algún tipo de resistencia mecánica capaz de disipar la energía cinética de esta mediante calor generado por fricción.

3.1.2.1. Tecnología empleada

Para realizar dicha resistencia en el eje, el director del proyecto diseñó y fabricó un un freno mecánico que, mediante palancas, transforma el par generado por un servomotor en una fuerza lineal de un valor elevado en el extremo contrario, a cambio se reduce el valor de los desplazamientos en dicho extremo. Los fundamentos de esto se encuentran en el **«método del trabajo virtual»**.

El servomotor escogido para aportar este par al sistema de frenado fue un **servomotor de alta potencia HD-1501MG**. Los motivos de la elección son el alto par que es capaz de ofrecer cuando su posición se encuentra fijada, su pequeño tamaño y el bajo precio frente a servomotores de más potencia. El mismo cuenta con un cable de alimentación continua, la cual puede estar comprendida entre $4.8\,\mathrm{V}$ y $6\,\mathrm{V}$, un cable de conexión a tierra y un cable a través del que se transmite la señal de control. Para un funcionamiento óptimo se recomienda conectar el mismo a una fuente de alimentación externa de corriente continua, sin embargo es factible conectarlo a la placa Arduino para comprobar su correcto funcionamiento, ya que para los ensayos que se realizan en este proyecto no es necesario disponer del freno, puesto que los mismos se realizan empleando una fuente de viento controlada. Finalemente, el servo es capaz de girar un ángulo comprendido entre 0° y 180°.

Figura 17. Servomotor de alta potencia HD-1501MG

3.1.2.2. Instalación de los dispositivos y esquemas de conexión

El servomotor se conecta con la placa Arduino según el siguiente esquema[5]:

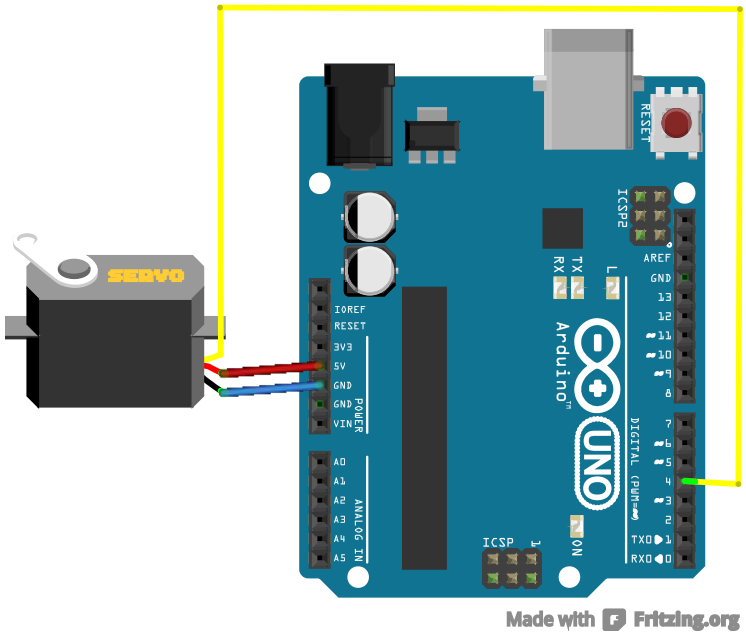


Figura 18. Esquema de conexión del servomotor sobre la placa Arduino

[5]Los elementos referidos los sensores han de enontrarse a su vez conectados según el esquema indicado con anterioridad. Este esquema de conexión ha de superponerse con el anterior.

Se puede observar como el pin de señal al que se conecta el servomotor es el número 4, no indicado como pin PWM. Aunque el control del servomotor funciona mediante este sistema de «modulación de ancho de pulso» la librería de Arduino empleada para su control consigue que el mismo pueda controlarse desde cualquiera de los pines digitales. Aun así, de cara al futuro es necesario tener en cuenta que dicha librería inhabilita el uso de PWM en los pines 9 y 10 en aquellas placas que no sean de tipo **Arduino Mega**.

Tras realizar las conexiones pertinentes, se procede a instalar el servomotor dentro del sistema de palancas del freno. Para ello se retira una de las tapas del mecanismo, se sitúa el servomotor en el hueco designado para ello y se atornilla el extremo del brazo del servo al extremo del brazo del freno, una vez hecho esto se vuelve a colocar la tapa. Finalmente, el sistema completo de frenado se acopla debajo de la tapa superior de la turbina a la distancia necesaria del eje para maximizar su efectividad. Cuando el servo se active este rotará desplazando las palancas hasta realizar una presión en el eje que, por efecto de la fricción, frene la rotación de la turbina.

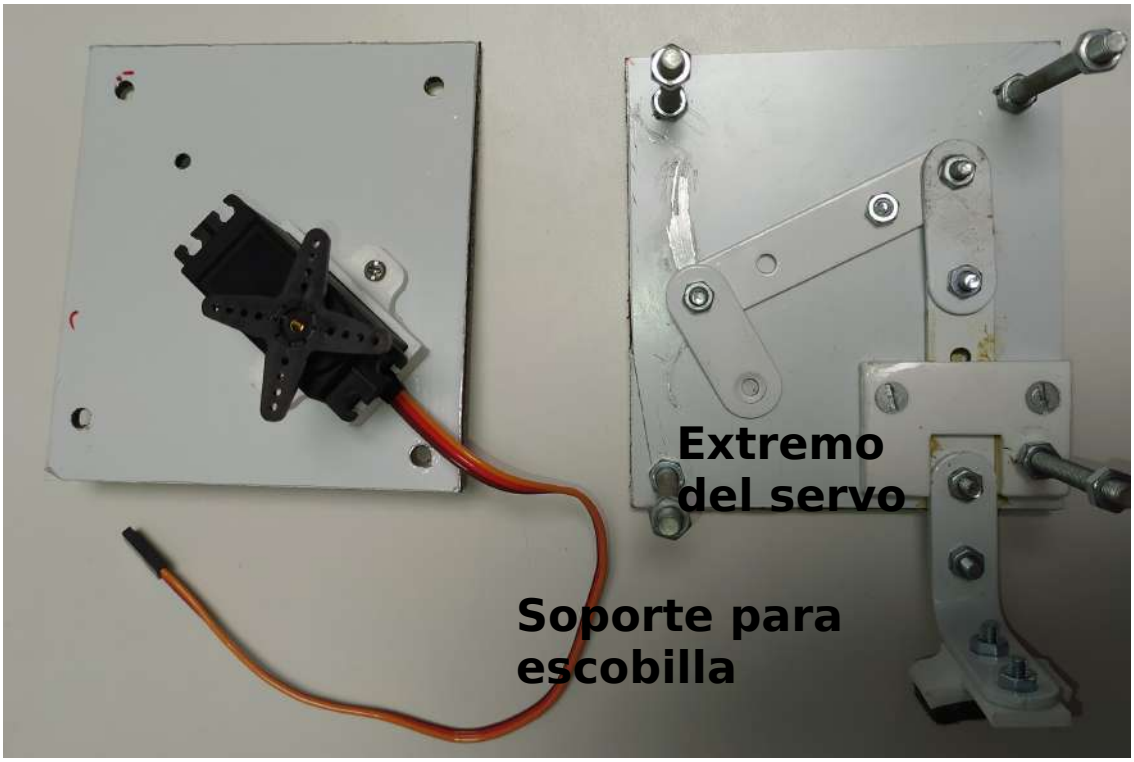


Figura 19. Mecanismo de freno desmontado

3.1.2.3. Cálculos del microprocesador

Para hacer efectiva la reducción de velocidad en la turbina se emplea, dentro del código principal de la placa Arduino, la función **«servo()»**, la cual, como se ha podido ver en los extractos de código anteriores, se encuentra dentro de cada condición **«if(digitalRead('número de pin')=='HIGH/LOW')»**. Para que dicha función pueda ser empleada es necesario incluir previamente la librería **«servo»**, pudiendo hacerse esto escribiendo al principio del código:

```
#include <Servo.h> // Incluye la librería necesaria para el servomotor del freno
```

Código 5. Línea 1 del «programa principal» de la placa Arduino (PARTE III, sección 1.1)

Un vez hecho esto es necesario crear un «objeto» para controlar el servo:

```
Servo freno; // Crea un objeto para controlar el servomotor del freno
```

Código 6. Línea 4 del «programa principal» de la placa Arduino (PARTE III, sección 1.1)

Después, es necesario definir las variables de control del freno:

```
// Freno
int pos, poso=90, poslims=180, poslimi=90;
```

Código 7. Líneas 21-22 del «programa principal» de la placa Arduino (PARTE III, sección 1.1)

La variable **(pos)** define la posición angular brazo en cada momento, la variable **(poso)** fija dicha posición en 90° cada vez que se inicialice el programa principal y las variables **(poslimi)** y **(poslims)** fijan los valores del rango de control en un intervalo comprendido entre 90° y 180°, debiéndose esto al diseño del sistema mecánico de freno. Todas ellas son de tipo entero o «int» ya que la resolución de control de este tipo de motores es de 1°. Una vez definidas las mismas se procede a asignar las condiciones del freno dentro de la sección de inicialización de código o **«void setup()»**:

```
freno.attach(4); // Asigna el pin 4 al servomotor del freno

freno.write(poso); // Coloca el freno en la posición inicial
```

Código 8. Líneas 35-37 del «programa principal» de la placa Arduino (PARTE III, sección 1.1)

Finalmente, habiendo situado la función de freno en los lugares pertinentes del bucle principal o **«void loop()»** se procede a definir dicha función. Dentro de esta, se lee al principo la posición angular actual del brazo del servo: si esta no es la de frenado y las velocidades del viento o de la turbina superan los límites establecidos se asigna a la variable el valor de la posición de frenado, si esta no es la de máxima apertura y las velocidades del viento y de la turbina no superan los límites establecidos se asigna a la variable el valor de la posición de máxima apertura. Una vez asignado dicho valor se procede a cambiar la posición del servo.

En este punto es necesario reconocer el trabajo del alumno Javier Colinas Cano ya que de la forma en la que se cambiaba originalmente la posición del servo (añadiendo en cada iteración 1° hasta llegar al valor límite) se producían escalones en el par generado debido a que la velocidad de la turbina se reducía o aumentaba antes de que se completase la orden de freno en algunos casos. Con este cambio el problema se vio solucionado.

```
//Función de movimiento del freno
void servo()
{
    pos = freno.read(); // Detecta la posición actual del servomotor

    if((va>=vamax || vh>=vhmax) && pos<poslims) // Si la velocidad del viento o la
        velocidad angular de la turbina superan o igualan los valores límite
        establecidos y la posición del servomotor no es la de frenado
    {
        // pos +=1; Inicialmente se encotraba así, por ello el servomotor tenía
            problemas puesto que había perturbaciones en su movimiento y en el par.
        pos = poslims; // El servomotor se sitúa en posición de frenado. Javier
            Colinas Cano sugirió este cambio y ahora funciona corréctamente.
    }

    if(va<vamax && vh<vhmax && pos>poslimi) // Si la velocidad del viento y la
        velocidad angular de la turbina son inferiores a los valores límite
        establecidos y la posición del servomotor no es la de máxima apertura
    {
        // pos -=1;
        pos = poslimi; // El servomotor se sitúa en posición de apertura
    }

    freno.write(pos); // Envía la señal correspondiente al servomotor
}
```

Código 9. «Función de movimiento del freno» de la placa Arduino (PARTE III, sección 1.2)

3.1.3. Envío de datos

La función de envío de datos desde la placa Arduino a un ordenador, así como todos los elementos necesarios para ello, ha sido desarrollada por el alumno Javier Colinas Cano en su PFC: **«Medición, transmisión y análisis de datos de un aerogenerador de eje vertical para uso doméstico»**. Esta se encuentra definida dentro del código principal como **«enviar()»**.

3.1.4. Gestión de datos

Tras conseguir enviar los datos a un ordenador, ya es posible utilizarlos para diversos fines. Tanto si se envían los datos de aceleración, velocidad o intervalos de tiempo entre conmutaciones de cada sensor, todos ellos están calculados a partir de los valores de tiempo en los que sucede cada conmutación, referidos a la placa Arduino. Por ello, para disponer de la máxima flexibilidad en cuanto a su operación es necesario contar con dos vectores de tiempos de conmutación referidos a un origen común; uno para los tiempos del anemómetro y otro para los tiempos del sensor Hall.

La creación de estos dos vectores corre a cuenta del alumno Javier Colinas Cano en su PFC: **«Medición, transmisión y análisis de datos de un aerogenerador de eje vertical para uso doméstico»**.

3.1.4.1. Tecnología empleada

Pudiendo obtener dichos vectores para cualquiera de los ensayos a realizar con la turbina, el programa elegido para operar con ellos ha sido Matlab. La motivación tras esto reside en la versatilidad y sencillez que el mismo ofrece para relizar operaciones complejas con gran cantidad de términos, así como la compatibilidad de la que dispone con otros programas como Arduino o Excel.

3.1.4.2. Cálculo parámetros de la turbina: centro de masas de las palas y momento de inercia de la turbina

Los valores de intervalos de tiempo obtenidos a través de la placa Arduino sirven para calcular tanto velocidades como aceleraciones. Mediante estas es posible obtener valores de la potencia extraída del viento a través de la turbina de manera comparativa, es decir: siendo capaces de observar si en cada momento la potencia en el eje es mayor o menor, pero sin obtener una magnitud medible de la misma.

Para conseguir valores fiables de la potencia mecánica que la turbina transmite a su eje es necesario calcular el momento de inercia de la misma alrededor del eje de rotación. Esto se consigue calculando los momentos de inercia de cada elemento individual que forma la turbina para despúes sumarlos respecto a un punto común de la misma. Para comprender este procedimiento es conveniente describir a fondo la disposición de las partes móviles de dicha turbina, componiéndose de:

- Un eje central hueco de aluminio de $900\,\mathrm{mm}$ de longitud, $7\,\mathrm{mm}$ de radio interior, $8\,\mathrm{mm}$ de radio exterior y $156\,\mathrm{g}$ de masa.
- Cinco palas compuestas por dos tapas de plástico creadas mediante fabricación aditiva siguiendo un perfil de ala dado, varias planchas de poliestireno expandido unidas entre sí con la forma del perfil de ala, situadas entre las dos tapas, y un recubrimiento de papel satinado que las rodea. La masa media de cada pala es de $459\,\mathrm{g}$ y el eje horizontal del perfil de ala mide $220\,\mathrm{mm}$.
- Dos varillas finas de aluminio entre cada pala y el eje de $245\,\mathrm{mm}$ de longitud y $19\,\mathrm{g}$ de masa. Contándose un total de diez varillas.

Una vez definidos todos los elementos se procede a calcular cada momento de inercia respecto al eje vertical central de la turbina. Para ello se comienza con el que presenta una mayor dificultad: el momento de inercia de cada pala.

Figura 20. Perfil de ala de las palas de la turbina

El método empleado en este caso ha sido la discretización del perfil de ala en diversos puntos, situados a la misma distancia entre sí y con un valor discreto de masa asociado a cada uno de ellos, el cual se ha obtenido dividiendo la masa total de la pala entre el número de puntos del sistema. Para ello se tienen que asumir dos suposiciones: la primera es que la forma del perfil de ala es constante a lo largo de toda la pala, lo cual es falso para este prototipo, ya que las planchas de poliestireno solapadas en el interior no tienen una forma perfecta y pueden existir huecos en los contornos del perfil, aun así, la idea en futuras versiones es que la construcción de las palas disponga de un mayor grado de precisión y homogeneidad, solucionándose por tanto este

problema; la segunda es que la masa para cada punto a lo largo del eje vertical sea igual a la del resto de puntos, siendo esto falso debido a la composición de la pala en los contornos, donde el recubrimiento y los posibles huecos pueden hacer que la masa de estos puntos no sea igual a la de los del interior, aun así este tipo de puntos es minoritario y se considera que el sistema empleado para realizar estos cálculos es completamente válido para aquellas versiones con una mejor construcción.

Figura 21. Pala de la turbina

Figura 22. Planta de la pala de la turbina

Mediante este modelo se puede calcular el momento de inercia respecto a cualquier punto, pertenezca o no al perfil de ala. Sin embargo, para facilitar futuros cambios que se puedan producir en el diseño de la turbina, se ha decidido que dicho momento de inercia se calcule

respecto al centro de masas del perfil, para lo cual se ha desarrollado otro programa creado en Matlab que obtiene la posición de dicho punto.

El diseño del perfil de ala empleado en este prototipo pertenece exclusivamente a Juan Romeo Granados, director de este PFC, por lo que para realizar los cálculos se ha empleado un perfil **«clarky-il»** [21], el cual es muy similar. Para poder ejecutar el programa correctamente es necesario disponer del perfil de ala en un archivo de tipo **«.xlsx»** tal y como se muestra a continuación:

x superior	y superior	x inferior	y inferior
0	0	0	0
0.0005	0.002339	0.0005	-0.00467
0.001	0.0037271	0.001	-0.0059418
0.002	0.0058025	0.002	-0.0078113
0.004	0.0089238	0.004	-0.0105126
0.008	0.013735	0.008	-0.0142862
0.012	0.0178581	0.012	-0.0169733
0.02	0.0253735	0.02	-0.0202723
0.03	0.0330215	0.03	-0.0226056
0.04	0.0391283	0.04	-0.0245211
0.05	0.0442753	0.05	-0.0260452
0.06	0.0487571	0.06	-0.0271277
0.08	0.0564308	0.08	-0.0284595
0.1	0.0629981	0.1	-0.0293786
0.12	0.0686204	0.12	-0.0299633
0.14	0.073436	0.14	-0.0302404
0.16	0.0775707	0.16	-0.0302546
0.18	0.0810687	0.18	-0.030049
0.2	0.0839202	0.2	-0.0296656
0.22	0.0861433	0.22	-0.0291445
0.24	0.0878308	0.24	-0.0285181
0.26	0.089084	0.26	-0.0278164
0.28	0.0900016	0.28	-0.0270696
0.3	0.0906804	0.3	-0.0263079
0.32	0.0911857	0.32	-0.0255565
0.34	0.0915079	0.34	-0.0248176
0.36	0.0916266	0.36	-0.024087
0.38	0.0915212	0.38	-0.0233606
0.4	0.0911712	0.4	-0.0226341
0.42	0.0905657	0.42	-0.0219042
0.44	0.0897175	0.44	-0.0211708
0.46	0.0886427	0.46	-0.0204353
0.48	0.0873572	0.48	-0.0196986
0.5	0.0858772	0.5	-0.0189619
0.52	0.0842145	0.52	-0.0182262
0.54	0.0823712	0.54	-0.0174914
0.56	0.080348	0.56	-0.0167572
0.58	0.0781451	0.58	-0.0160232
0.6	0.0757633	0.6	-0.0152893
0.62	0.0732055	0.62	-0.0145551

0.64	0.0704822	0.64	-0.0138207
0.66	0.0676046	0.66	-0.0130862
0.68	0.0645843	0.68	-0.0123515
0.7	0.0614329	0.7	-0.0116169
0.72	0.0581599	0.72	-0.0108823
0.74	0.0547675	0.74	-0.0101478
0.76	0.0512565	0.76	-0.0094133
0.78	0.0476281	0.78	-0.0086788
0.8	0.0438836	0.8	-0.0079443
0.82	0.0400245	0.82	-0.0072098
0.84	0.0360536	0.84	-0.0064753
0.86	0.031974	0.86	-0.0057408
0.88	0.0277891	0.88	-0.0050063
0.9	0.0235025	0.9	-0.0042718
0.92	0.0191156	0.92	-0.0035373
0.94	0.0146239	0.94	-0.0028028
0.96	0.0100232	0.96	-0.0020683
0.97	0.0076868	0.97	-0.0017011
0.98	0.0053335	0.98	-0.0013339
0.99	0.002969	0.99	-0.0009666
1	0.0005993	1	-0.0005993

Tabla 1. Coordenadas de un perfil de ala clarky-il en ejes x e y

Se puede observar como los valores del perfil vienen dados en unidades unitarias, es decir: para obtener las distancias reales es necesario multiplicar dichos valores por la longitud del perfil en el eje x. El encabezado de la tabla no se ha de incluir en el fichero. Las distintas columnas se distribuyen desde la columna «A» hasta la «D». Una vez se disponga del perfil el programa carga dicha tabla en vectores de datos mediante los comandos:

```
%% Extracción de datos del perfil de ala del fichero de Excel a Matlab

xs = xlsread('ClarkY.xlsx','A:A'); % Coordenadas en x de la curva superior
ys = xlsread('ClarkY.xlsx','B:B'); % Coordenadas en y de la curva superior
xi = xlsread('ClarkY.xlsx','C:C'); % Coordenadas en x de la curva inferior
yi = xlsread('ClarkY.xlsx','D:D'); % Coordenadas en y de la curva inferior
```

Código 10. Líneas 1-6 del «programa para el cálculo del centro de masas del perfil de ala» de Matlab (PARTE III, sección 2.1)

En la tabla de valores se puede observar como la resolución de los valores en el eje x no es constante, empezando con diferencias entre valores consecutivos de 0.0005 unidades mientras que en la zona media esta diferencia es de 0.02 unidades. Para poder calcular la posición del centro de masas es necesario reescalar los valores en este eje, para lo cual se ha decidido que la resolución en el mismo sea la más pequeña disponible en la tabla, es decir: 0.0005 unidades. Una vez se divida el eje x en estos valores se les ha de asignar valores en el eje y a los mismos, para lo cual se emplea la función de interpolación de Matlab. Como los perfiles de ala se diseñan siguiendo la ecuación de cierta curva y, en este caso, los valores de la misma que se nos ofrecen no se encuentran distribuidos homogeneamente en el eje x, se ha decidido emplear

la interpolación **«spline»**, aunque la diferencia en la posición del centro de masas respecto a emplear una interpolación lineal es inferior a la milésima. Todo esto se aprecia en el siguiente extracto de código:

```
%% Ajuste de vectores

% Resolución en el eje x
r = 5e-4;

% Reescalado de vectores
X = 0:r:1; % Vector de puntos en el eje x

X = X';

YS = interp1(xs,ys,X,'spline'); % Interpolación de los puntos del eje y de la
    curva superior del perfil para las coordenadas en x
YI = interp1(xi,yi,X,'spline'); % Interpolación de los puntos del eje y de la
    curva inferior del perfil para las coordenadas en x

X = X';
YS = YS';
YI = YI';

nX = numel(X); % Numero de componentes del vector de posición del eje x
```

Código 11. Líneas 8-25 del «programa para el cálculo del centro de masas del perfil de ala» de Matlab (PARTE III, sección 2.1)

Una vez se dispone del perfil de ala definido adecuadamente se procede a calcular la posición del centro de masas en el eje x. Para ello se emplea la ecuación:

$$x_G = \frac{\sum x_i \cdot m_i}{\sum m_i} \tag{13}$$

De este modo, el programa asigna a cada punto del eje x una masa ficticia en función de la distancia entre los límites superior e inferior en el eje y para cada coordenada en x. Tras esto, pondera los valores de estos diferenciales de masa en función de sus coordenadas en x y obtiene la coordenada en x del centro de masas.

```
% Eje x
n = 1;
while (n<=nX)
    dmx(n) = YS(n)-YI(n); % Diferencial de masa ficticio en el eje x
    n = n+1;
end

mf = sum(dmx); % Masa ficticia total del perfil

n = 1;
xg = 0;
while (n<=nX)
    xg = xg+dmx(n)*X(n); % Ponderación de la posición de cada diferencial en el
        eje x por el valor de masa ficticia de cada uno de los mismos
    n = n+1;
end
XGp = xg/mf; % Posición del centro de masas del perfil en el eje x
```

Código 12. Líneas 29-44 del «programa para el cálculo del centro de masas del perfil de ala» de Matlab (PARTE III, sección 2.1)

Como calcular los diferenciales de masa en el eje y es complicado cuando lo que se tiene son dos curvas para las mismas coordenadas en x, el método empleado ha sido calcular las coordenadas en el eje y de los centros de masas de los diferenciales de masa ficticia empleados con anterioridad. Tras esto se ponderan todas ellas en función del valor de masa ficticia que cada una presenta, obteniendo así la coordenada en el eje y del centro de masas.

```
% Eje y
n = 1;
while (n<=nX)
    ygdmx(n) = (YI(n)+YS(n))/2; % Posición del centro de masas en el eje y de cada
        diferencial de masas del eje x
    n = n+1;
end

n = 1;
yg = 0;
while (n<=nX)
    yg = yg+dmx(n)*ygdmx(n); % Ponderación de la posición de cada diferencial en
        el eje y por el valor de masa ficticia de cada uno de los mismos
    n = n+1;
end
YGp = yg/mf; % Posición del centro de masas del perfil en el eje y
```

Código 13. Líneas 46-59 del «programa para el cálculo del centro de masas del perfil de ala» de Matlab (PARTE III, sección 2.1)

En la siguiente imagen se puede apreciar la posición del centro de masas en unidades unitarias para los valores dados con anterioridad:

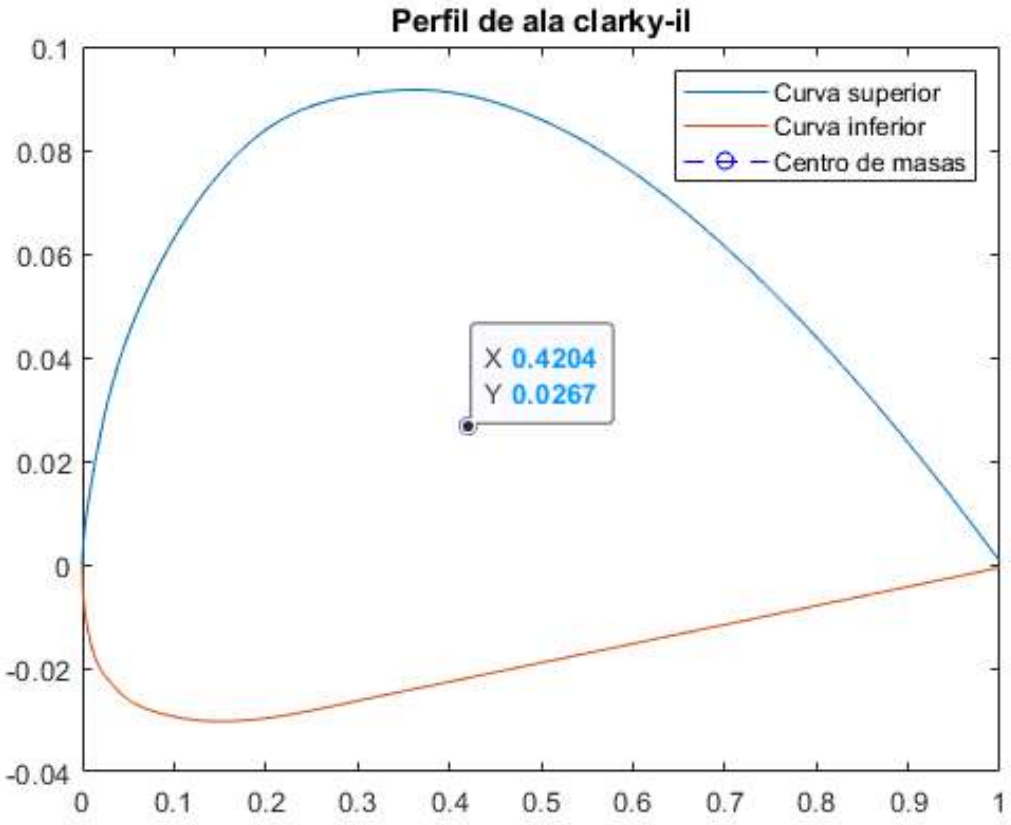


Figura 23. Coordenadas del centro de masas de un perfil de ala clarky-il en unidades unitarias

Una vez cargados en Matlab las coordenadas del centro de masas del perfil de ala, ya se puede ejecutar el programa de cálculo de momentos de inercia. Al principio de este se definen los parámetros pertinentes de la turbina del aerogenerador:

```
%% Parámetros

npg = 5;  % Número de palas del aerogenerador
nbu =2;  % Número de barras de unión entre cada pala y el eje

mp = 459e-3;  % Masa de cada pala en kg
mbu = 19e-3;  % Masa de cada barra de unión entre eje y pala en kg
me = 156e-3;  % Masa del eje de la turbina en kg

Lp = 220e-3;  % Longitud del eje horizontal del perfil de ala en m

Dpg = 245e-3;  % Distancia del centro de masas de la pala al eje de la turbina en m

Re = 8e-3;  % Radio exterior del eje de la turbina en m
Ri = 7e-3;  % Radio interior del eje de la turbina en m

Lbu = 245e-3;  % Longitud de cada barra de unión entre eje y pala en m
```

Código 14. Líneas 1-17 del «programa para el cálculo del momento de inercia de la turbina» de Matlab (PARTE III, sección 2.2)

Después, es necesario reescalar los vectores del perfil de ala, al igual que se hizo para el cálculo del centro de masas. En este caso la resolución elegida ha sido de 0.0001 unidades puesto que el modelo, en este caso, cuenta con muchos más puntos que en el anterior. Tras esto, el programa cambia el origen del sistema de referencia desde el punto frontal del perfil de ala hasta el centro de masas del mismo y aproxima los valores en el eje y de las curvas superior e inferior a la resolución elegida para que todos los puntos del modelo se encuentren separados entre sí a la misma distancia.

```
% Cambio del origen del sistema de referencia al centro de masas
n=1;
while(n<=nX)
    X(n) = X(n)-XGp;
    n = n+1;
end

n=1;
while(n<=nX)
    YS(n) = YS(n)-YGp;
    YI(n) = YI(n)-YGp;
    n = n+1;
end

% Reescalado de vectores
YS = round(YS,4);  % Aproximación de los valores en y de la curva superior del
    perfil a 4 decimales
YI = round(YI,4);  % Aproximación de los valores en y de la curva inferior del
    perfil a 4 decimales
```

Código 15. Líneas 38-54 del «programa para el cálculo del momento de inercia de la turbina» de Matlab (PARTE III, sección 2.2)

Habiendo realizado esto, el programa calcula el momento de inercia de las palas de la turbina en el eje principal z, para un sistema de referencia cuyo origen se encuentra en el centro de masas. Esto lo hace empleando la expresión:

$$I_{zp} = \sum m_i \cdot ((x_i - x_G)^2 + (y_i - y_G)^2) \tag{14}$$

En el extracto de código que se muestra a continuación se aprecia como, para cada punto del eje x, se genera un vector de puntos comprendidos entre la curva inferior y la curva superior separados entre sí por la resolución dada con anterioridad. El código calcula iterativamente el momento de inercia para cada uno de esos puntos en función de la distancia de los mismos al centro de masas de la pala, escalando esta en función de la longitud del eje horizontal del perfil de ala. Finalmente, suma los momentos de inercia de cada punto, en función de las distancias, y la multiplica por la masa de cada uno de ellos, obtenida tras dividir la masa total de la pala entre el número de puntos del sistema.

```
% Momento de inercia de la pala respecto al centro de masas de la pala
Izp = 0;
n1 = 1;
n2 = 1;
n3 = 0;
while(n1<=nX)
    Yn = YI(n1):r:YS(n1); % División del perfil de ala en los puntos comprendidos
        entre las curvas superior e inferior con una resolución r
    nYn = numel(Yn); % Numero de componentes del vector de posición del eje y
        (número de puntos en los que se discretiza el perfil de ala)
    while(n2<=nYn)
        Izp = Izp + ((Lp*Yn(n2))^2+(Lp*X(n1))^2); % Momento de inercia del perfil
            de ala en función de la longitud del eje horizontal y sin tener en
            cuenta la masa
        n2 = n2+1;
    end
    n3 = n3+n2-1;
    n2 = 1;
    n1 = n1+1;
end

dmp = mp/n3; % Masa de cada punto en los que se divide el perfil de ala
Izp = Izp*dmp;
```

Código 16. Líneas 58-76 del «programa para el cálculo del momento de inercia de la turbina» de Matlab (PARTE III, sección 2.2)

Una vez calculado el momento de inercia de las palas de la turbina, se procede a calcular el del resto de elementos que la forman, siendo esto más sencillo al tratarse de momentos de inercia ya estandarizados. En el caso del eje central de la turbina, se calcula empleando la fórmula para un cilindro hueco que gira alrededor de su eje central de revolución con una masa y unos radios, interior y exterior, dados:

$$I_{ze} = \frac{1}{2} \cdot m_e \cdot (R_i^2 + R_e^2) \tag{15}$$

En el caso de las varillas de unión entre las palas y el eje central, se calcula el momento de inercia de cada varilla empleando la fórmula para una varilla delgada con el centro de giro en uno de sus extremos y una longitud y una masa dadas:

$$I_{zbu} = \frac{1}{3} \cdot m_{bu} \cdot L_{bu}^2 \tag{16}$$

Finalmente, se procede a sumar los valores de cada momento de inercia multiplicados por el número de elementos de cada tipo, sin olvidar que para los momentos de las palas será necesario aplicar el «teorema de Steiner» o «de los ejes paralelos», para el cual la distancia a tener en cuenta es la que existe entre el centro de masas de cada pala y el eje central de la turbina. El «teorema de Steiner» estipula que para calcular el momento de inercia de un elemento respecto a cualquier punto del sistema se emplea la expresión:

$$I_{zP} = I_{zG} + m \cdot D^2_{P-G} \tag{17}$$

Siendo D^2_{P-G} la distancia entre el punto P y el centro de masas G. A continuación se puede observar como el código ejecuta tales operaciones:

```
% Momento de inercia del eje de la turbina respecto al eje de la turbina
Ize = (1/2)*me*(Re^2+Ri^2);

% Momento de inercia de cada barra de unión entre el eje y las palas al eje de la
    turbina
Izbu = (1/3)*mbu*Lbu^2;

% Momento de inercia total de la turbina respecto al eje de la turbina en kg*m^2
Jg = Ize+npg*nbu*Izbu+npg*Izp+npg*mp*Dpg^2;
```

Código 17. Líneas 78-85 del «programa para el cálculo del momento de inercia de la turbina» de Matlab (PARTE III, sección 2.2)

Una vez se han ejecutado ambos programas, el valor calculado del momento de inercia de la turbina es de $0.147\,761\,\mathrm{kg\,m^2}$. Una nota importante a tener en cuenta es que este valor no es el del prototipo real de turbina si no uno basado en un perfil de ala diferente, aunque este es un valor orientativo adecuado. Pese a esto, el programa se puede aplicar a cualquier otro modelo de turbina cambiándo los parámetros del mismo de la forma adecuada. Por último, es necesario recordar que dicho valor corresponde únicamente al momento de inercia de la turbina, sin tener en cuenta el del motor acoplado a la misma.

3.2. Análisis de los datos

En esta sección se realiza una explicación teórica de los métodos prácticos discurridos para poder obtener un modelo fiable de la turbina del aerogenerador a partir de los medios disponibles. Es necesario tener en cuenta que los resultados obtenidos de estos ensayos pueden estar sujetos a imprecisiones derivadas de las simplificaciones que en ellos se realizan, sin embargo, al tratarse este PFC de uno de los dos primeros proyectos referidos a este aerogenerador en concreto, se ha optado por asumir y explicar los posibles errores de dichos modelos con el objetivo de orientar y facilitar los futuros proyectos que puedan surgir.

3.2.1. Obtención de la potencia mecánica instantánea en el eje de la turbina

Una vez cumplidos los objetivos de medir y enviar los valores de tiempo de las conmutciones del anemómetro y el sensor Hall, se puede disponer de las magnitudes de velocidad y aceleración, tanto del viento como de la turbina, para los instantes de conmutación. A su vez, tras calcular

el momento de inercia de la turbina, si se combina este con los valores de aceleración de la misma se puede obtener una aproximación del par mecánico que transmite esta al eje de rotación. Finalmente, combinando este par con la velocidad de rotación de la turbina se obtiene una aproximación de la potencia mecánica extraída del viento hasta el eje.

Para la obtención de esta última, solamente son necesarios los tiempos de conmutación obtenidos a través del sensor Hall y el momento de inercia de la turbina. Para obtener la velocidad angular de la misma en $[\mathrm{rad\,s^{-1}}]$ se pueden emplear dos formas, de las que explicaremos sus consecuencias en la sección 4.3. La primera consiste en calcular la velocidad angular mediante el intervalo de tiempo de conmutación entre dos palas adyacentes, teniendo en cuenta para ello el número de palas de la turbina:

$$\omega_{generador} = \frac{\frac{2\pi \cdot 1000}{n_{palas\ generador}}}{t_{conmutacion\ pala\ adyacente} - t_{conmutacion\ inicial}} \tag{18}$$

La segunda forma consiste en emplear los tiempos de conmutación de cada pala consigo misma sin tener en cuenta el número de palas de la turbina:

$$\omega_{generador} = \frac{2\pi \cdot 1000}{t_{conmutacion\ final\ misma\ pala} - t_{conmutacion\ inicial}} \tag{19}$$

Para calcular la aceleración en ambos casos se divide la diferencia entre dos valores de velocidad por la diferencia entre sus respectivos tiempos de conmutación:

$$\alpha_{generador} = \frac{\omega_{generador\ final} - \omega_{generador\ inicial}}{t_{conmutacion\ final} - t_{conmutacion\ inicial}} \tag{20}$$

En el caso del par externo, generado por el viento, que se transmite al eje de la turbina se calcular empleando la ecuación fundamental de la dinámica de rotación, en donde:

$$M = \frac{dL}{dt} = \frac{d(J \cdot \omega)}{dt} = \frac{dJ}{dt} \cdot \omega + \frac{d\omega}{dt} \cdot J = 0 + \alpha \cdot J \tag{21}$$

Siendo L el momento angular y J[6] el momento de inercia. Pese a que la ecuación es correcta, esta no asume las pérdidas producidas por la fricción en los rodamientos, en función de la velocidad de giro, ni el par resistente que los mismos puedan ofrecer. La ecuación fundamental de la dinámica de rotación modelada con la mayor precisión para este caso tendría la siguiente forma:

$$M_{viento} = J \cdot \frac{d\omega}{dt} + (B_0 + B_1 \cdot \omega + B_2 \cdot \omega^2 + ... + B_n \cdot \omega^n) \tag{22}$$

Esta es una ecuación diferencial en donde M es el momento externo transmitido a la turbina y cada B es un factor de rozamiento de la misma. Para aproximar estos habría que ser capaces de resolver la ecuación disponiendo de valores precisos de velocidad, variación de la misma y par externo aplicado sobre la turbina; despreciándose aun así el término de segundo orden y todos

[6]El momento de inercia total de la turbina queda reflejado mediante la letra J a partir de este punto de la memoria para que el mismo no pueda verse confundido con la intensidad eléctrica en futuros proyectos relacionados con este prototipo.

los superiores al mismo, ya que habitualmente sus valores suelen ser varios órdenes de magnitud inferiores a los del resto. Estos valores se pueden conseguir, para este prototipo, tomando los valores de velocidad y aceleración para cada punto de una curva donde la turbina decelere sin el efecto de ningún momento externo, siendo el número de valores de B que se obtengan igual al número de puntos de la curva de los que se disponga, creando para ello un sistema de ecuaciones compatible determinado.

Una vez calculado el par y empleando, a su vez, la velocidad angular se puede calcular la potencia mecánica que la turbina extrae del viento. Al encontrarse esta sin conectar a ninguna máquina eléctrica, toda la energía que extraiga del viento se transformará en la energía cinética de rotación de la propia turbina, también una pequeña parte de esta se disipará en forma de calor, debido a las pérdidas producidas por la fricción con el aire y los rodamientos. Independientemente de si se tienen o no en cuenta las pérdidas por fricción, la potencia extraída del viento por la turbina es:

$$P_{viento} = M_{viento} \cdot \omega \tag{23}$$

Si no se tienen en cuenta las pérdidas la potencia extraída del viento será la misma que se transmita al eje de la turbina, la cual, independientemente de las pérdidas, es en cada instante:

$$P_{eje} = J \cdot \frac{d\omega}{dt} \cdot \omega = J \cdot \alpha \cdot \omega \tag{24}$$

3.2.2. Ensayo para la obtención de la curva característica de potencia mecánica frente a velocidad angular en la turbina

Disponiendo de los valores de potencia instantánea en el eje, la velocidad angular instantánea de la turbina y la velocidad media del viento durante los ensayos, se puede obtener la curva característica de potencia mecánica frente a velocidad angular. Nótese que dicha curva no es la empleada habitualmente, puesto que la potencia en este caso es mecánica. Esto es así ya que no se dispone de una máquina eléctrica para medir la potencia eléctrica generada; sin embargo, se quiere comprobar si el programa funcionaría en el caso de sustituir la potencia mecánica en el eje, especificada con anterioridad, por la potencia eléctrica medida mediante la tensión y la intensidad que generaría una máquina eléctrica conectada al mismo.

La curva de potencia eléctrica frente a velocidad angular de la turbina o «curva de seguimiento óptimo» es la curva que se ha de programar en los dispositivos de control vectorial para conseguir el mayor rendimiento posible modificando el par eléctrico en función de la velocidad del aerogenerador en cada instante; esta se emplea en turbinas de velocidad variable. Para obtenerla es necesario que una corriente de aire con velocidad y caudal constantes atraviese la turbina; tras esto, se arranca la misma mediante la máquina eléctrica, acelerándola hasta su velocidad máxima y tomando medidas de potencia para cada uno de los valores de velocidad de la turbina, manteniendo la misma constante. Para cada valor de velocidad del viento se obtiene una curva de potencia en función de la velocidad de la turbina distinta, cada una de ellas con un valor de potencia máximo. Si se unen y se interpolan dichos valores de cada una de las diferentes curvas se obtiene la «curva de seguimiento óptimo» a programar en el dispositivo de control vectorial.

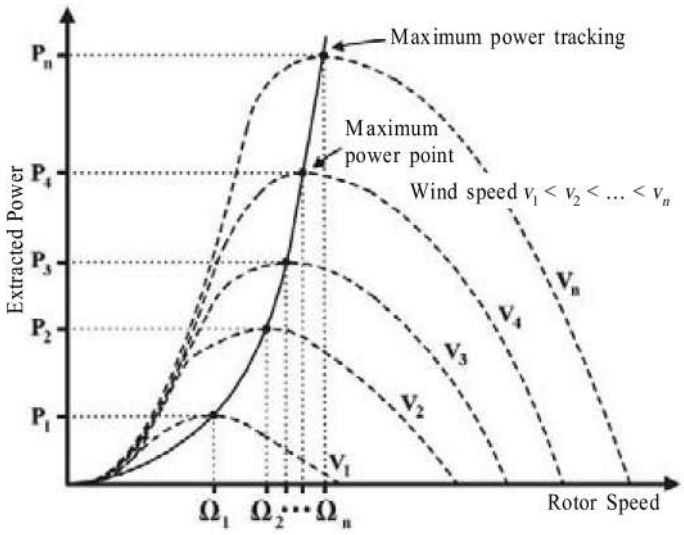


Figura 24. Ejemplo genérico de una curva de seguimiento óptimo [22]

Como curiosidad, cabe resaltar que, en aquellos aerogeneradores donde no se emplea el control vectorial, la curva a tener en cuenta es la de potencia eléctrica generada frente a la velocidad del viento. Esto es así puesto que la velocidad de la turbina es proporcional a la frecuencia de la red eléctrica a la que se encuentre conectada, por lo tanto, si esta gira siempre a la misma velocidad la única variable libre que influye en el par mecánico al que son sometidas las palas, el cual se traduce en energía mecánica en conjunto a la velocidad de giro, es la velocidad del viento, obviando cambios en la presión y la temperatura del aire. [23]

A continuación se muestra la combinación de ambas curvas en forma de superficie truncada para un valor de $20\,\mathrm{kW}$:

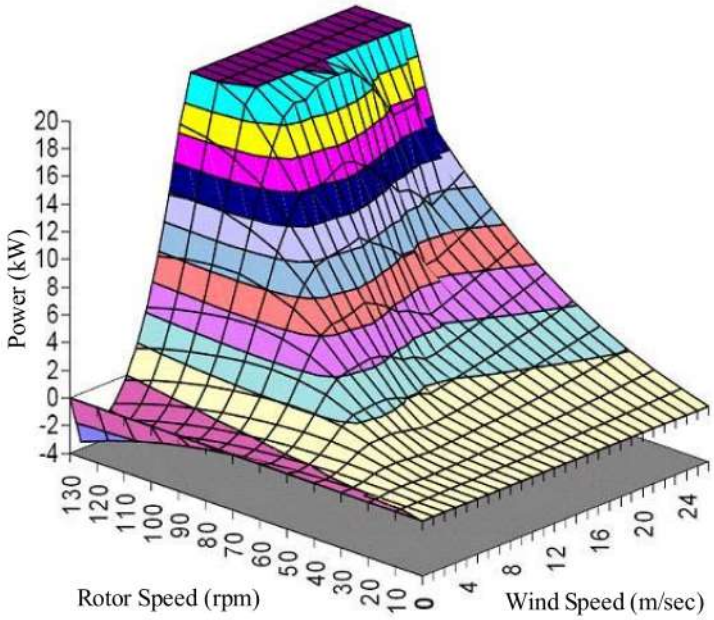


Figura 25. Ejemplo genérico de una superficie de potencia frente a velocidades del viento y de una turbina [22]

Una vez aclarada la motivación de generar una curva con la potencia mecánica en el eje, se puede proceder a explicar la consecución de la misma. Para ello se dispone en el taller de dos ventiladores iguales de eje horizontal, cada uno con tres velocidades distintas. Estos se sitúan de tal manera que el flujo de aire de cada uno incida de igual manera sobre la turbina y sobre el anemómetro. Tras esto, se podrán realizar cuatro ensayos para distintas velocidades del viento: el primero, situando los ventiladores alejados del aerogenerador y encendiéndolos a la mínima velocidad y el resto, con los ventiladores colocados a una distancia próxima al aerogenerador, empleando cada una de las tres velocidades disponibles.

Como para obtener valores de energía mecánica es necesario que aparezcan valores de aceleración distintos de cero, se comienza arrancando los ventiladores mientras se evita que la turbina rote. Con esto se consigue que el anemómetro alcance el régimen permanente sin que se tomen datos en la turbina. Después, se desbloquea la turbina, permitiendo que esta acelere hasta alcanzar su velocidad terminal. Se muestrean los valores de potencia mecánica en función del tiempo y se asocian al valor de velocidad angular de la turbina en cada instante. Tras esto, se obtiene una gráfica de puntos por cada valor de velocidad del viento en donde se relaciona velocidad angular con potencia en el eje. Finalmente, se buscan los máximos de cada gráfica y se interpolan formando una curva.

En el caso del anemómetro, las dos variables importantes a calcular son la velocidad y la aceleración del viento, siendo esta última prescindible en este proyecto, aunque se haya decidido calcularla también. Estas se obtiene de igual manera que en la placa Arduino pero de forma todavía más precisa, ya que Matlab no trunca los valores decimales. Para esto se obtiene la velocidad angular del anemómetro en $[\mathrm{rad\,s^{-1}}]$, en función del número de conmutaciones por vuelta y el tiempo entre conmutaciones en $[\mathrm{ms}]$, para después multiplicarla por el radio y el factor de cazoleta, obteniendo así la velocidad del viento. Como esta oscila en cada instante, es necesario calcular después la velocidad media del viento para cada ensayo.

$$\omega_{anemometro} = \frac{\frac{2\pi \cdot 1000}{n_{conmutaciones\ anemometro}}}{t_{conmutacion\ final} - t_{conmutacion\ inicial}} \tag{25}$$

$$v_{viento} = c_{anemometro} \cdot R_{anemometro} \cdot \omega_{anemometro} \tag{26}$$

La aceleración angular se obtiene como la diferencia de velocidades angulares entre dos conmutaciones dividida entre el tiempo entre conmutaciones. Para obtener la aceleración lineal del viento se multiplica esta aceleración angular por el radio y el factor de cazoleta.

$$\alpha_{anemometro} = \frac{\omega_{anemometro\ final} - \omega_{anemometro\ inicial}}{t_{conmutacion\ final} - t_{conmutacion\ inicial}} \tag{27}$$

$$v_{viento} = c_{anemometro} \cdot R_{anemometro} \cdot \alpha_{anemometro} \tag{28}$$

3.3. Algoritmos empleados

En esta sección se explica el funcionamiento de los distintos códigos empleados para modelar todas las variables explicadas con anterioridad.

3.3.1. Inicialización y ensayo en la turbina

Una vez se dispone de las variables de tiempos de conmutación del anemómetro y del sensor hall almacenados en un fichero de tipo **«.mat»**, se puede proceder a cargarlos en un programa de Matlab que permita su muestreo con un nivel de precisión óptimo. A este se le ha denominado como **«Ensayo en la turbina»** y se encuentra en la sección 2.4 de la PARTE III. Como el mismo incluye funcionalidades diseñadas para calcular puntos de la «curva de seguimiento óptimo» de la turbina, será necesario ejecutarlo varias veces para diferentes ensayos, por lo que es necesario añadir un programa que se ejecute previamente al primero de los ensayo para declarar el primer punto de la curva e inicializar un contador que permita el correcto funcionamiento de dicho programa.

El mismo se denomina como **«Inicialización de los ensayos para obtener la curva de seguimiento óptimo de la turbina»** y solamente declara una variable numérica que actúa como contador de ensayos y un vector columna que contiene velocidad media del viento en $[\mathrm{m\,s^{-1}}]$, potencia mecánica máxima en $[\mathrm{W}]$ y velocidad angular de la turbina en $[\mathrm{rad\,s^{-1}}]$, todas ellas de valor nulo.

```
%% Inicialización de los ensayos de curva característica

ne = 2; % Inicializa el índice del número de ensayos realizados para distintas
    velocidades del viento
M = [0;0;0]; %Almacena las variables velocidad media del viento, potencia mecánica
    máxima y velocidad angular de la turbina para esa potencia en una matriz
    inicializándolas en 0
```

Código 18. «Programa de inicialización de los ensayos para obtener la curva de seguimiento óptimo de la turbina» de Matlab (PARTE III, sección 2.3)

Habiendo ejecutado este código, ya es posible analizar el número de ensayos que se desee. Si se necesita borrar la matriz de valores máximos **(M)** para crear una nueva, solamente es necesario volver a ejecutar el programa anterior.

Una vez comprendido esto, ya se puede ejecutar el programa de análisis de ensayos. Al principio de este aparece una sección de inicialización donde se se borran todas las variables, exceptuándo las del programa de inicialización y el momento de inercia de la turbina ya explicado con anterioridad. A su vez, es necesario definir diversos parámetros físicos del aerogenerador anteriormente explicados.

```
%% Inicialización

clearvars -except Jg ne M % Elimina todas las variables excepto el momento de
    inercia de la turbina y la inicialización de los ensayos de curva
    característica(Evita errores)
close all % Cierra las gráficas (Evita confusiones con gráficas de ensayos
    distintos)
```

Código 19. Líneas 1-4 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

```
%% Parámetros físicos del aerogenerador

% Anemómetro
Ra = 70e-3; % Radio desde el eje del anemómetro a el centro de la pala en [m]
coma = 2; % Número de conmutaciones por vuelta que realiza el anemómetro
ca = 200/(21*pi); % Factor de cazoleta

% Generador
Rg = 265e-3; % Distancia a la que se encuentra el imán que detecta el sensor hall
    del eje del generador en [m]
npg = 5; % Número de conmutaciones por vuelta que realiza el sensor hall (número
    de palas del aerogenerador)
```

Código 20. Líneas 6-15 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Después, se muestran diversas opciones de uso en el programa: pudiéndose elegir el intervalo de tiempo de muestreo; si se desea situar el origen de tiempos en el tiempo de primera conmutación del sensor hall o en el tiempo de primera conmutación de cualquiera de los dispositivos; cuál de los tres «modos» de muestreo se quiere emplear; si se desea corregir los errores de muestreo, en donde el sensor hall no haya detectado una de las conmutaciones cuando debería; o el tipo de interpolación que se quiera utilizar para obtener los valores de velocidad en los tiempos entre conmutaciones de cada sensor.

```
%% Opciones del programa

Tmin = 0; % Tiempo mínimo de muestreo en [ms] (No funciona con "modo=2")
Tmax = 300000; % Tiempo máximo de muestreo en [ms]
origen = 1; % Tiempo inicial en primera conmutación de cualquier dispositivo(0),
    tiempo inicial en primera conmutación del sensor hall(1)
modo = 3; % Sin compensación de palas(1), con compensación de palas(2), con un
    imán en total(3)
correccion = 1; % Sin corrección de error de muestreo(0), con corrección de error
    de muestreo(1)
ncorreccion5 = 0.91; % Factor de corrección de error de muestreo para cinco imanes
ncorreccion1 = 0.51; % Factor de corrección de error de muestreo para un imán
interpolaciona = 'lineal'; % Selecciona el tipo de interpolación en la velocidad
    del viento
interpolaciong = 'spline'; % Selecciona el tipo de interpolación en la velocidad
    de la turbina
```

Código 21. Líneas 17-27 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Los dos primeros «modos» están referidos a el muestreo de variables del sensor hall mediante la colocación de un imán en cada pala, calculando el **«modo = 1»** la velocidad total de la turbina muestreando entre pala y pala y cálculando el **«modo = 2»** la velocidad de cada una de las cinco palas para luego ponderarla; los efectos de esto serán discutidos en Capítulo 4. El **«modo = 3»** se emplea en aquellos ensayos donde solamente estuviese colocado un imán en total.

Una vez especificadas las opciones pertinentes, el programa carga los datos de tiempos de conmutación en el anemómetro **(ta)** y en el sensor hall **(th)**, guardados en un fichero de tipo **«.mat»** mediante la función:

```
%% Carga de vectores de tiempo

load('un_iman_pot3_cerca.mat')
```

Código 22. Líneas 29-31 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

A partir de aquí, se continua fijando el origen de tiempos. En esta parte, el parámetro **(Tmin)** todavía no se tiene en cuenta porque, de hacerlo, los valores de velocidades y aceleraciones se verían distorsionados debido a la alteración en la diferencia de tiempos de conmutaciones iniciales. La única opción implicada es **«origen»**, que en el caso de ser **«origen = 0»** se ejecutaría el siguiente código:

```
if (ta(1)<th(1))
    ta0 = ta(1);
    n=1;
    while(n<=na)
        ta(n) = ta(n)-ta0;
        n = n+1;
    end
    n=1;
    while(n<=nh)
        th(n) = th(n)-ta0;
        n = n+1;
    end
end
```

Código 23. Líneas 41-53 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Teniendo distintas variaciones, que no se muestran en el cuadro anterior, en donde únicamente cambia el tiempo inicial, denominado aquí como **(ta0)** en función del de qué dispositivo es menor. Como puntualización. las variables **(na)** y **(nh)** son el número de conmutaciones de anemómetro y sensor hall, respectivamente, obtenidas mediante las funciones:

```
% Contabilizar el número de conmutaciones en cada dispositivo
na = numel(ta);
nh = numel(th);
```

Código 24. Líneas 35-37 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Si **«origen = 1»** se fuerza directamente a que el tiempo inicial sea el primer tiempo de conmutación del sensor hall y se modifica el vector de tiempos del mismo de una forma similar a como se ha hecho en el código 23. En cambio, para el anemómetro es necesario sobreescribir su vector de tiempos **(ta)** para que no existan valores inferiores a **(t0)**.

```
t0 = th(1);

% Anemómeteo
n1 = 1;
n2 = 1;
while(n1<=na)
    if(ta(n1)>=t0)
        t(n2) = ta(n1); % Escribe los valores de tiempo superiores al origen de
            conmutación del sensor hall en un vector auxiliar t
        n2 = n2+1;
    end
    n1 = n1+1;
end
clearvars ta; % Borra el vector de tiempo antiguo
ta = t; % Crea un nuevo vector de tiempos límitado al tiempo máximo
clearvars t; % Borra el vector de tiempo auxiliar t
na = numel(ta);
n = 1;
while(n<=na)
    ta(n) = ta(n)-t0;
    n = n+1;
end

% Sensor hall
n = 1;
while(n<=nh)
    th(n) = th(n)-t0;
    n = n+1;
end
nh = numel(th);
```

Código 25. Líneas 86-114 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Tras esto, se procede a limitar el tiempo máximo de muestreo, para lo que se tiene en cuenta al parámetro **(Tmax)**. El procedimiento consiste en reescribir en los vectores de tiempo los valores inferiores al tiempo especificado. A continuación, se muestra cómo se hace esto para los tiempos del anemómetro, que es exáctamente igual a cómo se hace en el sensor hall. Como puntualización, cabe destacar que cada vez que se actualice un vector de tiempos se cuentan sus elementos al igual que se muestra en el código 24.

```
n1 = 1;
n2 = 1;
while(n1<=na)
    if(ta(n1)<=Tmax)
        t(n2) = ta(n1); % Escribe los valores de tiempo inferiores al tiempo
            máximo en un vector auxiliar t
        n2 = n2+1;
    end
    n1 = n1+1;
end
clearvars ta; % Borra el vector de tiempo antiguo
ta = t; % Crea un nuevo vector de tiempos límitado al tiempo máximo
clearvars t; % Borra el vector de tiempo auxiliar t
```

Código 26. Líneas 120-131 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

El último ajuste a realizar relacionado con los vectores de tiempo consiste en generar un vector de tiempos de conmutación para cada pala en caso de que se encuentre activado **«modo =**

2». Para esto se crea un índice **(m)** que varía entre 1 y 5 para indicar a qué pala corresponde cada conmutación del sensor hall, guardando estas en vectores de tiempos específicos asociados a cada una.

```
n = 1;
m = 1;
n1 = 1;
n2 = 1;
n3 = 1;
n4 = 1;
n5 = 1;
while(n<=nh)
    if(m==1)
        th1(n1) = th(n);
        n1 = n1+1;
    end
    if(m==2)
        th2(n2) = th(n);
        n2 = n2+1;
    end
    if(m==3)
        th3(n3) = th(n);
        n3 = n3+1;
    end
    if(m==4)
        th4(n4) = th(n);
        n4 = n4+1;
    end
    if(m==5)
        th5(n5) = th(n);
        n5 = n5+1;
    m = 0;
    end
    n = n+1;
    m = m+1;
end
```

Código 27. Líneas 154-185 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Los valores de velocidades y aceleraciones que se pueden calcular a partir de todos estos vectores de tiempo irán asociados únicamente a los valores de dichos vectores, sin disponerse de datos para los puntos intermedios. Es esta la razón por la que a partir de aquí se crea un vector de tiempo en [ms] donde se disponga de todos los valores de tiempos del ensayo para poder almacenar los valores del resto magnitudes a lo largo de los mismos.

```
if(ta(na)<th(nh))
    T = th(nh);
end

if(ta(na)>th(nh))
    T = ta(na);
end

if(ta(na)==th(nh))
    T = ta(na);
end

t = 0:T;  % Genera un vector de tiempo en [ms]
```

Código 28. Líneas 197-209 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

A continuación, se muestra como se calcula la velocidad angular del anemómetro **(wa)** en [$\mathrm{rad\,s^{-1}}$], asociada al vector de tiempo de conmutación **(ta)**, de las cual se obtiene la velocidad en otras unidades, así como la aceleración.

```
n = 1;
while (n<=na)
    if(n==1)
        wa(n) = 0; % El valor de la velocidad inicial es 0
    end
    if(n>1)
        wa(n) = (2*pi*1000/coma)/(ta(n)-ta(n-1)); % Velocidad angular del
            anemómetro en [rad/s]
    end
    n = n+1;
end
```

Código 29. Líneas 214-223 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

El procedimiento de cálculo de las velocidades a partir de las que se obtienen el resto de variables en la turbina es igual que el anterior, cambiando las operaciones de cálculo. En este caso las velocidades se obtienen en [Hz] para disponer de más flexibilidad al operar posteriormente. Las disitas velocidades se calcularán dependiendo del **«modo»** seleccionado en las opciones del pograma[7].

```
wgHz(n) = (1000/npg)/(th(n)-th(n-1)); % Velocidad angular de la turbina en con 5
    imanes[Hz]

wgHz1(n) = 1000/(th1(n)-th1(n-1)); % Velocidad angular de la pala 1 en [Hz]

wgHz(n) = 1000/(th(n)-th(n-1)); % Velocidad angular de la turbina con 1 imán en
    [Hz]
```

Código 30. Líneas 247, 326 y 286 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Puede ocurrir que, debido a la alta velocidad que alcanza el prototipo de la turbina, el sensor de efecto hall no detecte alguna de las conmutaciones, produciendo esto que no se tenga en cuenta ese valor de tiempo en concreto. Esto trae como consecuencia que en la siguiente conmutación se contabilize un intervalo más grande. Ocasionándose con ello que la velocidad se vea gravemente disminuida en esa conmutación y por ello la aceleración entre ese tiempo y los adyacentes se dispare, produciéndose con ello errores de medida que no son despreciables. Para evitar las distorsiones derivadas de este problema se ha ideado un método en donde, cada vez que uno de los valores de velocidad sea inferior o igual a cualquiera de sus velocidades adyacentes multiplicadas por un factor, este es sustituido por la media entre sus dos velocidades adayacentes. Dicho factor es distinto si las medidas se han tomado con un imán en cada pala **(ncorreccion5)** o si solamente se ha empleado un imán en total **(ncorreccion1)**. El motivo tras esto es que las diferencias entre las velocidades erroneas según cada uno de estos modos difiere y se prefiere actuar con la mayor precisión posible a la hora de sustituir medidas. Dicha funcionalidad se activa mediante el parámetro **«correcion»** disponible en las opciones del programa.

[7] Aunque aquí solamente se muestre la velocidad de la primera pala, el programa calcula la de las cinco.

```
n = 1;
while(n<=nh && correccion==1)
    if(n>2)
        if(wgHz(n-1)<=ncorreccion1*wgHz(n-2) && wgHz(n-1)<=ncorreccion1*wgHz(n))
            wgHz(n-1) = (wgHz(n)+wgHz(n-2))/2;
        end
    end
    n = n+1;
end
```

Código 31. Líneas 291-299 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Un hecho importante a tener en cuenta es que, mediante este sistema, se fija un valor de velocidad arbitrario para uno de los tiempos de conmutación en donde la velocidad ha disminuido drásticamente. Sin embargo, no se añade al vector de tiempos el valor de tiempo de conmutación no detectado puesto que habría que recalcular otra vez todo lo anterior en función de los nuevos vectores de tiempo para, aun así, seguir obteniendo un resultado aproximado. Mediante este sistema las aceleraciones se seguirían viendo muy distorsionadas si se calculasen mediante los vectores de tiempo originales, por lo tanto se ha decidido calcular las aceleraciones con los valores de velocidad ya interpolados para cada valor del vector de tiempo **(t)** en [ms]. Aun así, se han calculado también las aceleraciones en función de los vectores de tiempo originales para que sea más fácil detectar los puntos conflictivos. A continuación, se muestra cómo se ha calculado la aceleración del anemómetro sin interpolar previamente las velocidades, siendo el mismo procedimiento al del resto de aceleraciones:

```
n=1;
while (n<=na)
    if(n==1)
        aa(n) = NaN; % El valor de la aceleración inicial no es un número
    end
    if(n>1)
        aa(n) = (wa(n)-wa(n-1))/(ta(n)-ta(n-1)); % Aceleración angular del
            anemómetro en [rad/s^2]
    end
    n = n+1;
end
```

Código 32. Líneas 226-235 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Como bien se ha indicado anteriormente, si se quiere obtener valores de aceleraciones con la mayor precisión posible es necesario interpolar los valores de velocidades disponibles para todos los instantes de tiempo. Esto se consigue mediante la función de interpolación que incluye Matlab. A continuación, se muestra este procedimiento para la velocidad angular del anemómetro, siendo exactamente igual al del resto de velocidades:

```
wat = interp1(ta,wa,t,interpolaciona); % Interpola la velocidad angular del
    anemómetro en [rad/s]
```

Código 33. Línea 508 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

La función **«interp1»** precisa de tres argumentos obligatorios: dos vectores de coordenadas con la misma longitud, en este caso **(ta)** y **(wa)**, y un vector con las coordenadas para las que se quiera obtener los valores interpolados del segundo vector **(wa)**, que en este caso es el vector de tiempo **(t)**. El parámetro **(interpolaciona)** se puede encontrar en la sección de opciones del programa y sirve para especificar el tipo de interpolación; si no se incluye, se realiza una interpolación lineal por defecto.

Una vez dispuestas todas las velocidades para cada instante de tiempo, se pueden calcular los valores de aceleración con la mayor fiabilidad posible. Esto se realiza de la misma forma que se muestra en el extracto de código 32, pero empleando como diferenciales las variaciones de velocidad entre cada [ms]. A continuación, se muestra la aceleración angular del anemómetro para cada instante de tiempo, siendo el procedimiento el mismo para el resto de aceleraciones:

```
n=1;
while (n<=T+1)
    if(n==1)
        aat(n) = NaN; % El valor de la aceleración inicial no es un número
    end
    if(n>1)
        aat(n) = (wat(n)-wat(n-1))/(t(n)-t(n-1)); % Aceleración angular del
            anemómetro en [Hz^2]
    end
    n = n+1;
end
```

Código 34. Líneas 514-523 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Tras haber calculado aceleraciones y velocidades, ya se puede utilizar la funcionalidad que permite mostrar los valores a partir de un tiempo mínimo **(Tmin)**. Esta almacena los valores de velocidades y aceleraciones pertinentes en varios vectores auxiliares. La función se ejecuta en este punto del programa para que los valores de velocidad iniciales no se vean alterados por eliminar tiempos inferiores a **(Tmin)**.

```
n1 = 1;
n2 = 1;
while(n1<=T+1)
    if(t(n1)>=Tmin)
        wauxa(n2) = wat(n1); % Escribe los valores de velocidad para tiempos
            superiores al tiempo mínimo en un vector auxiliar wauxa
        wauxg(n2) = wgHzt(n1); % Escribe los valores de velocidad para tiempos
            superiores al tiempo mínimo en un vector auxiliar wauxg
        aauxa(n2) = aat(n1); % Escribe los valores de aceleración para tiempos
            superiores al tiempo mínimo en un vector auxiliar aauxa
        aauxg(n2) = agHzt(n1); % Escribe los valores de aceleración para tiempos
            superiores al tiempo mínimo en un vector auxiliar aauxg
        n2 = n2+1;
    end
    n1 = n1+1;
end
```

Código 35. Líneas 540-551 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Una vez creados los vectores auxiliares, se reescriben sobre los vectores originales para que sea más fácil llevar su seguimiento con los mismos nombres.

```
clearvars wat; % Borra el vector de velocidad antiguo
clearvars wgHzt; % Borra el vector de velocidad antiguo
clearvars aat; % Borra el vector de aceleración antiguo
clearvars agHzt; % Borra el vector de aceleración antiguo
wat = wauxa; % Crea un nuevo vector de velocidad límitado al tiempo mínimo
wgHzt = wauxg; % Crea un nuevo vector de velocidad límitado al tiempo mínimo
aat = aauxa; % Crea un nuevo vector de aceleración límitado al tiempo mínimo
agHzt = aauxg; % Crea un nuevo vector de aceleración límitado al tiempo mínimo
clearvars wauxa; % Borra el vector de velocidad auxiliar wauxa
clearvars wauxg; % Borra el vector de velocidad auxiliar wauxg
clearvars aauxa; % Borra el vector de aceleración auxiliar aauxa
clearvars aauxg; % Borra el vector de aceleración auxiliar aauxg
```

Código 36. Líneas 552-563 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Tras disponer de los vectores de velocidades y aceleraciones base del anemómetro y la turbina, se procede a calcular los vectores de estas magnitudes en diversas unidades de medida. Para esto se multiplican los vectores por los factores correspondientes en cada caso.

```
% Anemómetro
vvmt = ca*Ra*wat; % Velocidad del viento en [m/s]
vvkmt = 3.6*ca*Ra*wat; % Velocidad del viento en [km/h]
avmt = ca*Ra*aat; % Aceleración del viento en [m/s^2]
avkmt = 3.6*ca*Ra*aat; % Aceleración del viento en [km/h^2]

% Turbina sin ponderación por palas o con un imán por pala
wgt = 2*pi*wgHzt; % Velocidad angular de la turbina en [rad/s]
wgrpmt = 60*wgHzt; % Velocidad angular de la turbina en [rpm]
agt = 2*pi*agHzt; % Aceleración angular de la turbina en [rad/s^2]
agrpmt = 60*agHzt; % Aceleración angular de la turbina en [rpm^2]
```

Código 37. Líneas 568-577 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Es reseñable que esta funcionalidad, asociada al tiempo mínimo de muestreo, solamente se aplica a la velocidad y aceleración ponderada de las palas cuando **«modo = 2»**. Esto es debido a que si se realizase también con las de cada pala por separado los vectores tendrían longitudes distintas, produciéndose errores, siendo el trabajo requerido para solucionar los mismos de nulo valor añadido. Posteriormente, se realiza el mismo proceso para sobreescribir el vector **(t)**.

```
n1 = 1;
n2 = 1;
while(n1<=T+1)
    if(t(n1)>=Tmin)
        taux(n2) = t(n1); % Escribe los valores de tiempo superior al tiempo
            mínimo en un vector auxiliar taux
        n2 = n2+1;
    end
    n1 = n1+1;
end
clearvars t; % Borra el vector de tiempo antiguo
t = taux; % Crea un nuevo vector de tiempo límitado al tiempo mínimo
clearvars taux; % Borra el vector de tiempo auxiliar taux
```

Código 38. Líneas 709-720 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

En cuanto a la velocidad ponderada de las palas del **«modo = 2»**, la misma se calcula realizando una media ponderada de las velocidades de cada pala en cada instante de tiempo. Tras esto se obtienen el resto de variables relacionadas de igual modo que en los casos anteriores. En el Capítulo 4 se explicará por qué habrá que tener especial cuidado a la hora de interpretar estas variables.

```
wgHztm = (wgHzt1+wgHzt2+wgHzt3+wgHzt4+wgHzt4)/5; % Velocidad angular de la turbina
    en [Hz]
```

Código 39. Línea 671 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Tras obtener todas las velocidades y aceleraciones pertinentes para cada instante de tiempo, el programa procede a calcular la potencia mecánica de la turbina en el eje para los distintos modos. Para ello opera según lo expresado en la ecuación (24):

```
% Sin compensación de palas o con un imán por pala
Pmg = Jg*(wgt.*agt);
[Pmgmax,nmax] = max(Pmg); % Indica el valor de la potencia máxima y su posición en
    el vector Pmg

% Con compensación de palas
if(modo==2)
    Pmg1 = Jg*(wgt1.*agt1);
    Pmg2 = Jg*(wgt2.*agt2);
    Pmg3 = Jg*(wgt3.*agt3);
    Pmg4 = Jg*(wgt4.*agt4);
    Pmg5 = Jg*(wgt5.*agt5);

    Pmgm = Jg*(wgtm.*agtm);
    [Pmgmmax,nmax] = max(Pmgm); % Indica el valor de la potencia máxima y su
        posición en el vector Pmg
end
```

Código 40. Líneas 747-761 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Finalmente, el programa almacena los valores de velocidad media del viento en $[\mathrm{m\,s^{-1}}]$, potencia mecánica máxima en $[\mathrm{W}]$ y velocidad angular de la turbina en $[\mathrm{rad\,s^{-1}}]$ en un vector columna que añade a la matriz **(M)** en cada ensayo. Tras esto, se le añade una unidad al contador de ensayos.

```
N = [vmvm;Pmgmax;wgt(nmax)]; %Almacena las variables velocidad media del viento,
    potencia mecánica máxima y velocidad angular de la turbina para esa potencia
    en una matriz
M = [M, N]; % La matriz se va ampliando por columnas a medida que se realizan más
    ensayos
ne = ne+1; % Aumenta el contador de ensayos realizados para distintas velocidades
    del viento
```

Código 41. Líneas 940-942 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

Los valores de velocidad media del viento se han calculado a partir de las medidas del anemómetro, asegurándose de no contabilizar los valores no numéricos del vector velocidad para que no se produzcan errores.

```
nt = numel(t); % Contabiliza el número de elementos del vector de tiempo
n = 1;
nn = isnan(wat);
while (n<=nt)
    if (nn(n)==1)
        wan(n) = 0; % Crea un vector lógico indicando con un 1 los valores no
            numéricos (NaN) y con un 0 los que sí lo son
        n = n+1;
        else
        wan(n) = wat(n);
        n = n+1;
    end
end
wma = sum(wan)/T; % Velocidad angular media del anemómetro durante el ensayo en
    [rad/s]
vmvm = ca*Ra*wma; % Velocidad media del viento durante el ensayo en [m/s]
vmvkm = 3.6*ca*Ra*wma; % Velocidad media del viento durante el ensayo en [km/h]
```

Código 42. Líneas 724-738 del «programa para el análisis de ensayos en la turbina» de Matlab (PARTE III, sección 2.4)

3.3.2. Dibujo de la curva de seguimiento óptimo de la turbina

El funcionamiento del programa para el cálculo de la curva de seguimiento óptimo es sencillo. Solamente necesita la especificación de el número de decimales que se quiere para cada valor de velocidad de la curva. Tras esto, se genera un vector de potencias **(Pm)** y otro de velocidades **(w)**. Finalmente, se interpolan los valores de potencia para cada valor de velocidad angular del vector **(wi)** cuyo valor máximo es la máxima velocidad de la matriz **(M)**. Cuanto más ensayos se realicen mayor precisión tendrá la curva. Llegado a este punto, es necesario recordar de nuevo que los valores de potencia necesarios para que esta curva tenga sentido físico han de ser los valores de potencia eléctrica obtenida de una máquina conectada al eje de la turbina y que actualmente se usan los valores de potencia mecánica de forma provisional.

```
%% Opciones del programa

r = 1e-3; % Resolución para la velocidad angular

%% Ajuste de resolución de la velocidad

n = 1;
while(n<ne)
    Pm(n) = M(2,n);
    w(n) = M(3,n);
    n = n+1;
end

wmax = max(w); % Velocidad angular máxima en [rad/s]

wi = 0:r:wmax; % Genera un vector de velocidades de la resolución "r" dada
Pmi = interp1(w,Pm,wi,'pchip'); % Interpola valores de potencia para cada velocidad
```

Código 43. Líneas 1-17 del «programa de dibujo de la curva de seguimiento óptimo en la turbina» de Matlab (PARTE III, sección 2.5)

3.3.3. Estimación de las pérdidas mecánicas de la turbina

De manera adicional, se ha diseñado un programa de Matlab destinado a calcular de forma empírica los coeficientes de pérdidas mecánicas de la turbina B_0, B_1, B_2...B_n, ya expuestos en la ecuación (22). Esto se hace mediante la creación de un sistema determinado de ecuaciones lineales con la forma genérica $[A] \cdot \vec{x} = \vec{b}$ en donde $[A]$ es la matriz de coeficientes asociada al vector de incógnitas $\vec{x}$ y $\vec{b}$ es el vector de términos independientes.

En este caso el vector de incógintas sería:

$$\vec{x} = \begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} \tag{29}$$

Siendo la matriz de coeficientes asociados a este:

$$[A] = \begin{bmatrix} \omega_1^0 & \omega_1^1 & \omega_1^2 & \cdots & \omega_1^n \\ \omega_2^0 & \omega_2^1 & \omega_2^2 & \cdots & \omega_2^n \\ \omega_3^0 & \omega_3^1 & \omega_3^2 & \cdots & \omega_3^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_{n+1}^0 & \omega_{n+1}^1 & \omega_{n+1}^2 & \cdots & \omega_{n+1}^n \end{bmatrix} \tag{30}$$

Mientras que el vector de términos independientes sería:

$$\vec{b} = \begin{pmatrix} J \cdot \alpha_1 \\ J \cdot \alpha_2 \\ J \cdot \alpha_3 \\ \vdots \\ J \cdot \alpha_{n+1} \end{pmatrix} \tag{31}$$

Una vez definido el sistema a resolver por Matlab, se comienza el programa eligiendo a través del parámetro **(r)** el número de coeficientes a calcular. Este número es el mismo que el de ecuaciones así, como el de puntos de la curva de frenado en ausencia de viento en los que el programa tomará datos de velocidad y aceleración.

```
%% Opciones del programa

r = 10;  % Número de coeficientes a calcular
```

Código 44. Líneas 1-3 del «programa de estimación de las pérdidas mecánicas de la turbina» de Matlab (PARTE III, sección 2.6)

Tras esto, el programa divide el intervalo de tiempos en función del número de coeficientes a calcular, almacenando en un vector **(T)** la posición de los subíndices de estos. Como los

subíndices han de ser números enteros y los valores de tiempos de esos puntos no tienen por qué, es necesario usar la función **«round()»**, la cual aproxima un número a su entero más próximo.

```
%Obtención de los puntos de muestreo (situados a la misma distancia entre ellos)
n = 1;
while(n<=r)
    if(n==1)
        T(n) = 1;
    end
    if(n>1)
        T(n) = round(n*R/r); % Aproximación al número entero más cercano del punto
            de tiempo seleccionado
    end
    if(n==r)
        T(n) = R;
    end
    n = n+1;
end
```

Código 45. Líneas 9-22 del «programa de estimación de las pérdidas mecánicas de la turbina» de Matlab (PARTE III, sección 2.6)

Finalmente, se definen la matriz $[A]$ y el vector $\vec{b}$ seleccionando adecuadamente cada columna y cada fila de los mismos y se opera usando la función de Matlab «\» cuya función es operar matrices directamente a través de sus inversas.

```
%% Resolución del sistema de ecuaciones lineales

% Obtención de la matriz A
c = 1;
while(c<=r)
    f = 1;
    while(f<=r)
        A(f,c) = wgt(T(f))^(c-1);
        f = f+1;
    end
    c = c+1;
end

% Obtención del vector columna b
f = 1;
while(f<=r)
    b(f,1) = -Jg*agt(T(f));
    f = f+1;
end

% Resolución del sistema de ecuaciones lineales
x = A\b;
```

Código 46. Líneas 24-45 del «programa de estimación de las pérdidas mecánicas de la turbina» de Matlab (PARTE III, sección 2.6)

3.3.4. Muestreo de variables en tiempo real

El programa de Matlab para el muestreo de las variables relevantes en tiempo real ha sido desarrollado por el alumno Javier Colinas Cano en su PFC: **«Medición, transmisión y análisis de datos de un aerogenerador de eje vertical para uso doméstico»**. Todos los comentarios que aparecen en dicho código han sido redactados por él. Las únicas partes a comentar en esta

sección son aquellas en donde se calculan valores de velocidad o aceleración, para lo cual se han empleado los mismos fragmentos de código que en los ensayos de la turbina vistos en la PARTE III, sección 2.4. La única diferencia existente es que los tiempos empleados para ello han sido los que iba recibiendo Matlab en tiempo real, denominados como **(cont1)** y **(cont2)** según si provenían del anemómetro o del sensor de efecto Hall, respectivamente.

Capítulo 4

Análisis de resultados

En este capítulo se ilustrarán todos los aspectos relevantes de uso de los programas desarrollados en el capítulo anterior. Todos los resultados que se mostrarán a continuación derivan de ensayos reales reales realizados en el espacio de La Nave en Madrid. Para el desarrollo de los mismos, se han empleado los ventiladores ya presentados en la subsección 3.2.2 en un espacio abierto en ausencia de corrientes aleatorias de aire. Las variables alteradas durante los mismos han sido: la distancia de los ventiladores a la turbina y el anemómetro, el ángulo de incidencia del viento sobre las palas de la turbina, la velocidad de los ventiladores y el número de imanes en cada pala de la turbina.

4.1. Análisis de la velocidad del viento

En esta sección se analizarán las gráficas de velocidad y aceleración del viento para distintos ensayos. El muestreo del mismo solamente se ve afectado por las opciones relacionadas con la selección del intervalo de tiempos de muestreo y por el tipo de interpolación que se realiza a la velocidad a lo largo del tiempo. Se recomienda que este último sea de tipo «lineal» ya que la velocidad del viento presenta variaciones muy grandes entre cada conmutación, las cuales suceden entre intervalos de tiempo del orden de las centésimas de segundo. A continuación se muestran distintos ensayos en donde o bien se mantiene constante la velocidad de los ventiladores o bien se cambia una vez la misma.

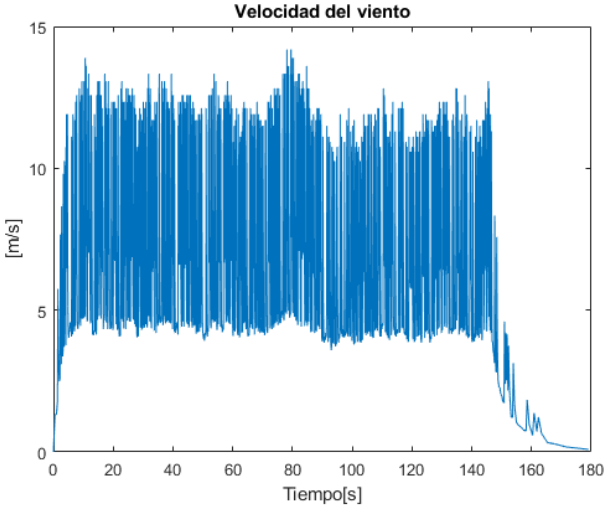


Figura 26. Velocidad del viento con el ventilador próximo a la turbina en su primera velocidad

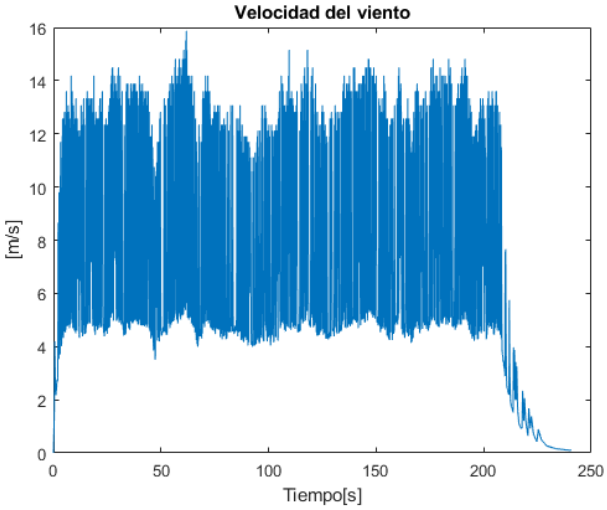


Figura 27. Velocidad del viento con el ventilador próximo a la turbina en su segunda velocidad

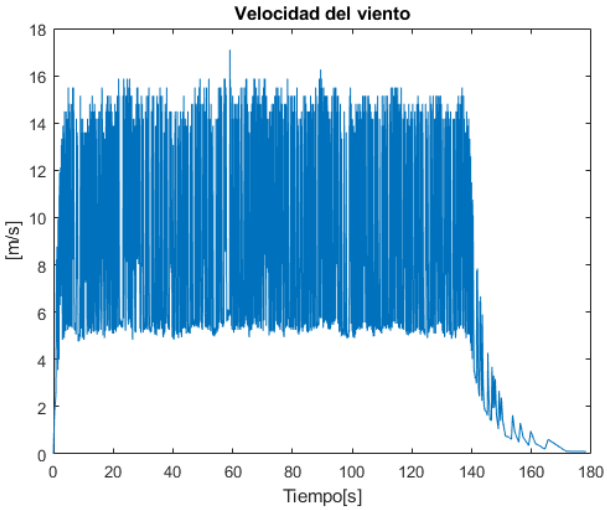


Figura 28. Velocidad del viento con el ventilador próximo a la turbina en su tercera velocidad

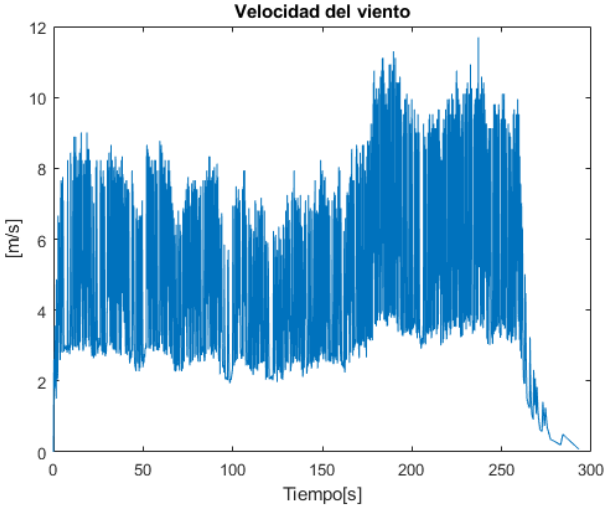


Figura 29. Velocidad del viento con el ventilador alejado de la turbina con un cambio de su primera velocidad a la segunda

Se puede observar como los valores de velocidad a lo largo del tiempo se encuentran comprendidos en un intervalo que varía en función de si aumenta o disminuye el caudal de aire que atraviesa al anemómetro. Estas fluctuaciones pueden deberse a dos motivos: el primero es que, como el anemómetro se encontraba acoplado al compartimento superior de la turbina durante los ensayos, las vibraciones de la carcasa ocasionasen dichas perturbaciones; esta teoría no se descarta ya que sería necesario realizar un ensayo con el anemómetro sobre una superficie fija.

Sin embargo, es más plausible que las oscilaciones se deban a que el anemómetro se encontraba en el centro de la corriente de aire producida por el ventilador. Esto sería el funcionamiento normal en condiciones ambientales, ya que sería muy improbable que el viento afectase solamente a un lado del anemómetro impulsándolo de forma constante. Como el anemómetro tiene tres palas, el mismo se ve frenado cuando a una de estas le incide el viento en su contra y se ve acelerado cuando esto no sucede; adicionalmente, como el anemómetro conmuta solamente dos veces por vuelta, sin ser este número múltiplo de tres, este efecto puede ser detectado comparando conmutaciones adyacentes en el tiempo.

4.2. Efecto de la función de corrección de muestreo

En esta sección se discutirá el efecto de la función de corrección de errores de muestreo **«correccion»** vista en el programa de análisis de ensayos en la turbina (PARTE III, sección 2.4). Los ajustes seleccionados pueden verse en el extracto de código 21.

A continuación, se muestra un ensayo genérico en la turbina. En este se ha arrancado la misma hasta su velocidad terminal con los ventiladores a su máxima velocidad. Finalmente, se han apagado los ventiladores y se ha dejado decelerar a la turbina sin oponer ninguna resistencia. Los ajustes del programa han sido:

Tmin	0
Tmax	Superior al tiempo de ensayo
Origen la primera conmutación del sensor Hall	No
Modo	Un imán en total (3)
Corrección de errores de muestreo	No
Interpolación de la velocidad del viento	Lineal
Interpolación de la velocidad de la turbina	Lineal

Tabla 2. Ajustes de un ensayo con el ventilador próximo a la turbina en su tercera velocidad

La primera gráfica a comentar es la de la velocidad de la turbina. En esta puede observarse como arranca con una curva aparentemente uniforme hasta llegar a una velocidad en torno a las $170\,\mathrm{rpm}$, a partir de la cual se producen una serie de descensos abruptos puntuales hasta aproximadamente la mitad de la velocidad previa a cada incidente. Esto se debe a que el sensor Hall ocasionalmente no detecta una de las conmutaciones, bien porque no reacciona lo suficientemente rápido o bien por las vibraciones que se producen en la turbina cuando se alcanzan altas velocidades, como puede verse reflejado a la velocidad terminal, donde se produce un ligero rizado en la velocidad. Se sabe que el fallo se produce en el sensor y no en los cálculos del programa porque la magnitud del descenso difiere si se han tomado las medidas de tiempos con un imán en lugar de con cinco. Si se produce con un imán el tiempo hasta la siguiente

conmutación es aproximadamente el doble de lo normal, por lo que la velocidad desciende aproximadamente a la mitad.

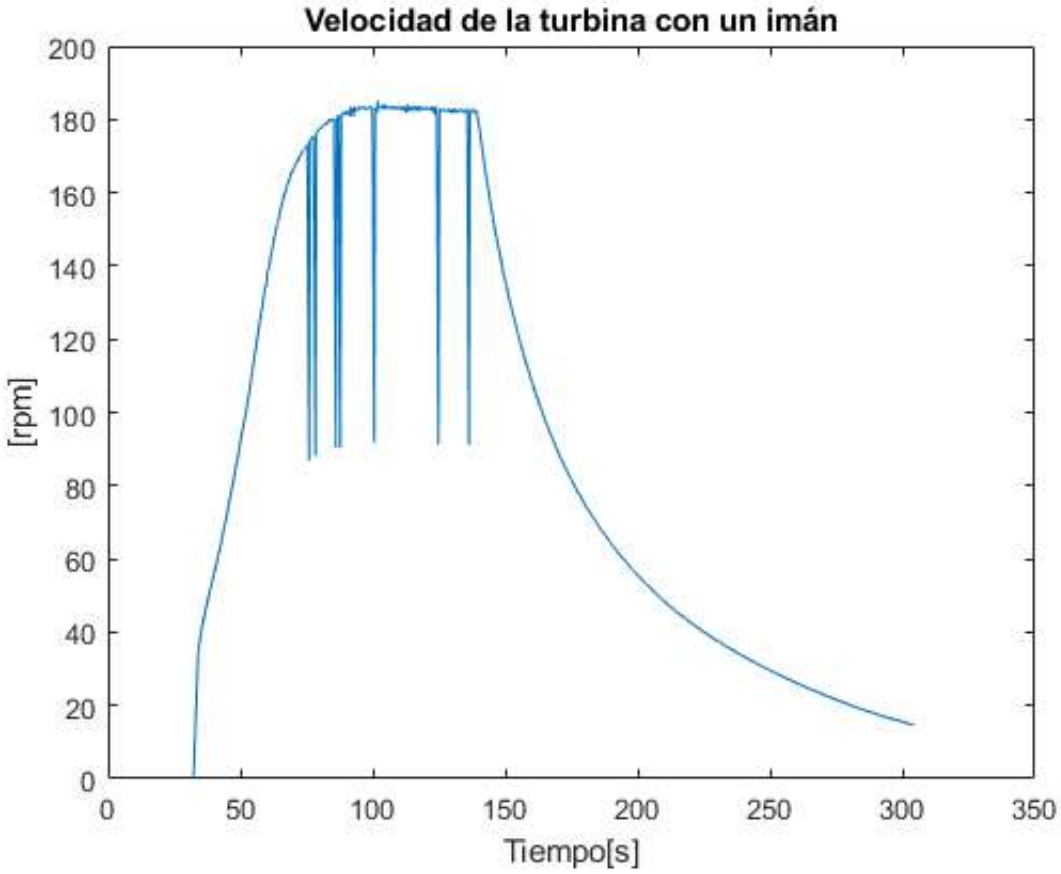


Figura 30. Velocidad de la turbina con un imán, con el ventilador próximo a la turbina en su tercera velocidad y sin corrección de errores de muestreo

Sin embargo, si el fallo se produce con cinco imanes y se cuentan los tiempos individuales de cada pala empleando el **«modo = 2»**, se detecta la conmutación de la pala adyacente, por lo que el intervalo de tiempo medido es 0.2 veces mayor aproximadamente, descendiendo con ello la velocidad en una quinta parte. Esto se puede apreciar en un ejemplo con los siguientes ajustes:

Tmin	0
Tmax	Superior al tiempo de ensayo
Origen la primera conmutación del sensor Hall	No
Modo	Un imán por pala con compensación (2)
Corrección de errores de muestreo	No
Interpolación de la velocidad del viento	Lineal
Interpolación de la velocidad de la turbina	Lineal

Tabla 3. Ajustes de un ensayo con el ventilador alejado de la turbina variando su velocidad

A continuación, se muestra la velocidad de cada pala para dicho ejemplo. Si se amplia la imagen a vista detalle de uno de los errores de medida se puede comprobar como los valores de velocidad de cada pala descienden bruscamente uno detrás de otro, ya que los intervalos entre conmutaciones se ven alterados para todas las palas.

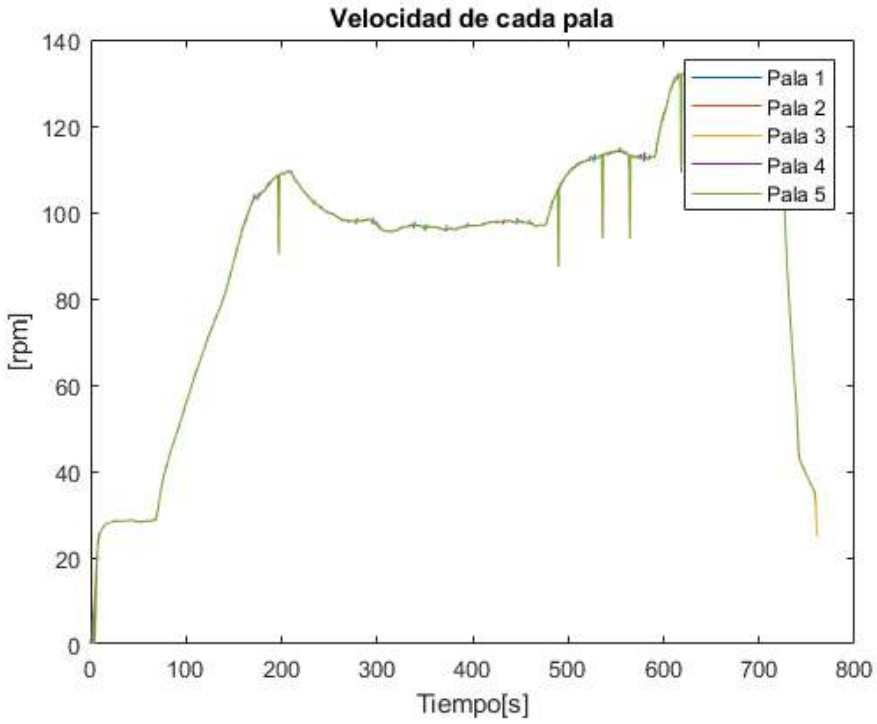


Figura 31. Velocidad de cada pala con cinco imanes, con el ventilador alejado de la turbina variando su velocida y sin corrección de errores de muestreo

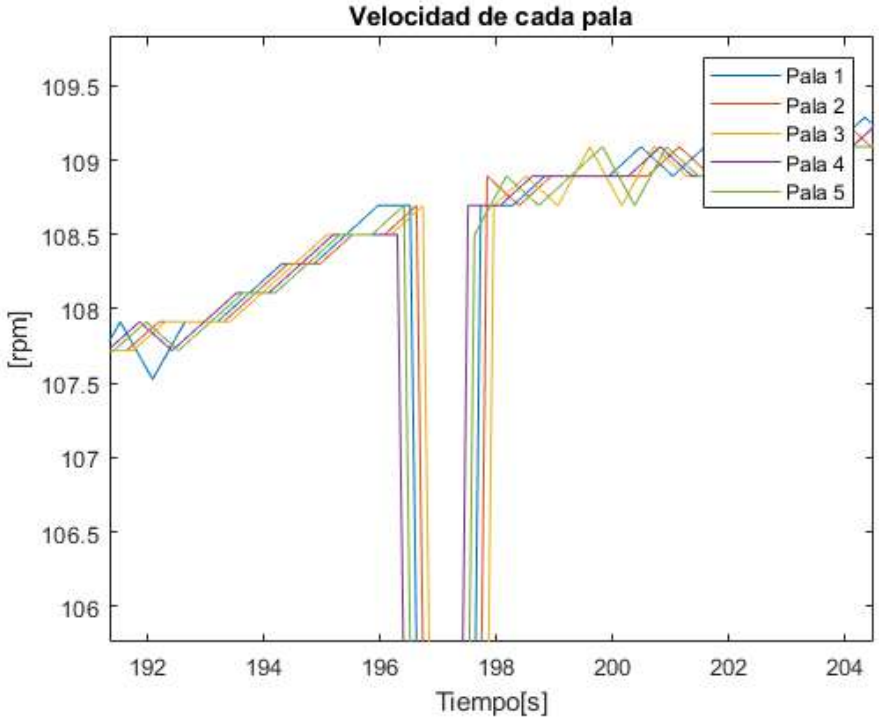


Figura 32. Vista detalle de un error de muestreo para la velocidad de cada pala

El principal problema de estos picos es que distorsionan gravemente los valores de aceleración, ya que durante los mismos esta se dispara varios órdenes de magnitud por encima de las aceleraciones adyacentes, distorsionando a su vez los valores de potencia mecánica calculados. Esto puede observarse en la siguiente gráfica asociada al ejemplo descrito en la tabla 2:

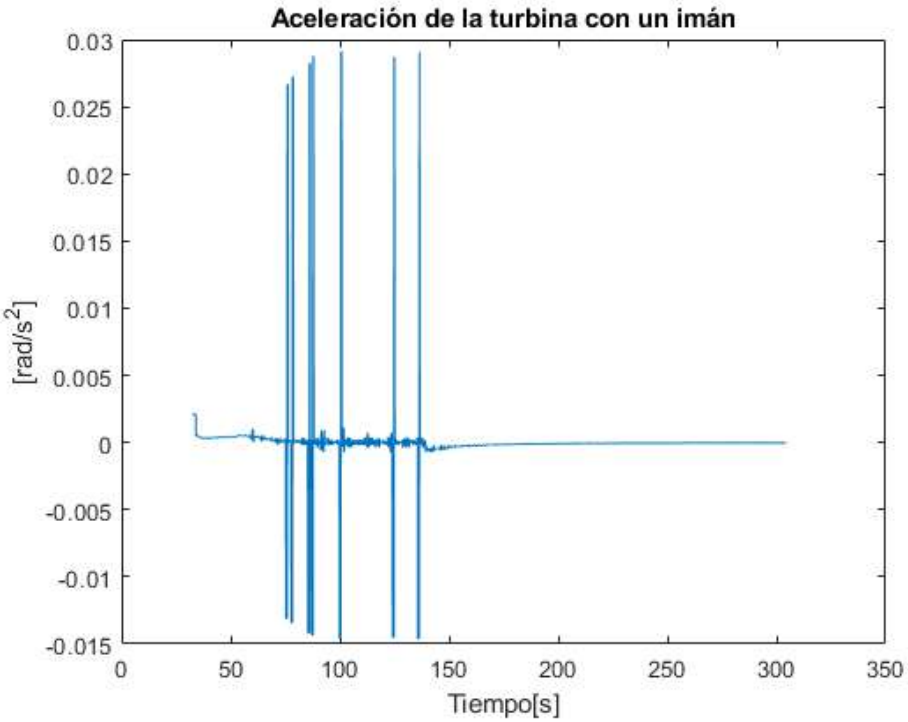


Figura 33. Aceleración de la turbina con un imán, con el ventilador próximo a la turbina en su tercera velocidad y sin corrección de errores de muestreo

Una vez expuestos los resultados con este tipo de error, se puede proceder a ilustrar el aspecto de los datos una vez corregidos. Para esto se muestran a continuación los resultados del caso descrito en la tabla 2 con el parámetro **«correccion = 1»**:

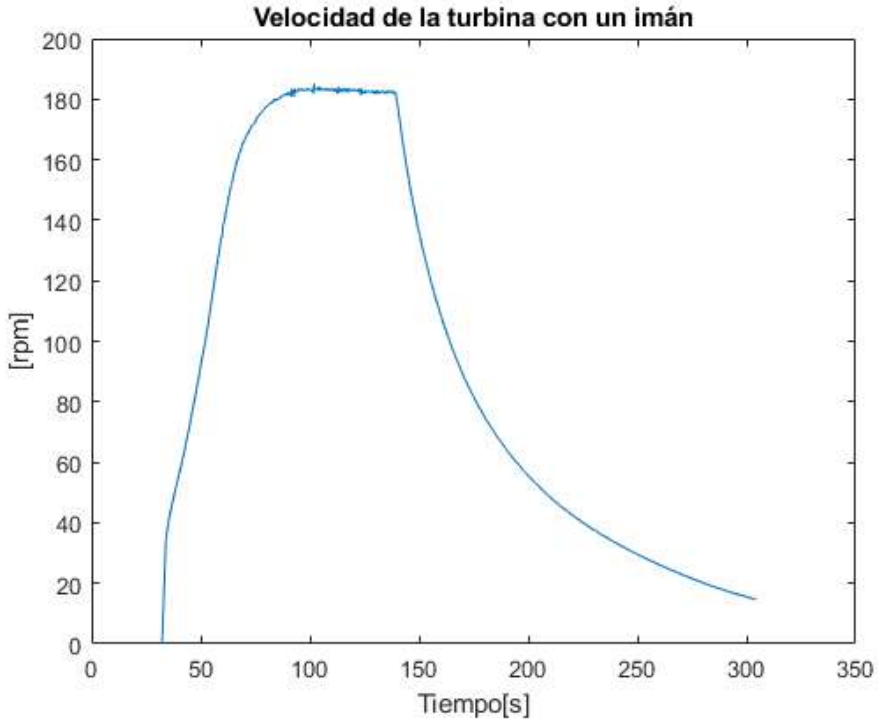


Figura 34. Velocidad de la turbina con un imán, con el ventilador próximo a la turbina en su tercera velocidad y con corrección de errores de muestreo

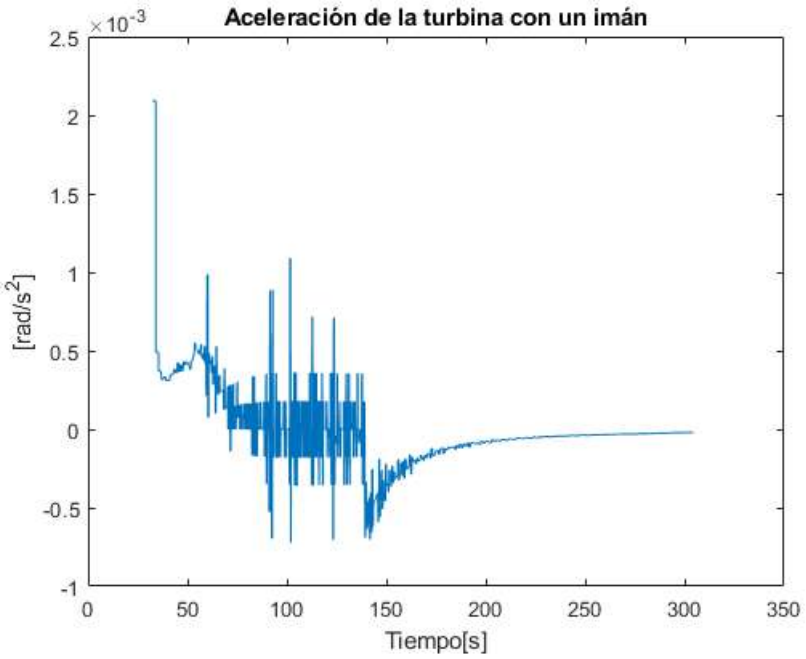


Figura 35. Aceleración de la turbina con un imán, con el ventilador próximo a la turbina en su tercera velocidad y con corrección de errores de muestreo

Finalmente, habiendo visto los efectos y causas de los errores de muestreo se puede concluir con que, en caso de que aparezcan, es conveniente corregirlos editando el parámetro **«correccion»** de la sección de opciones del programa. Si bien es verdad que los mismos aumentan su frecuencia cuanto mayor sea la velocidad de la turbina, siendo este uno de los motivos que llevó a reducir el número de imanes acoplados a la misma tras realizar varios ensayos.

4.3. Efecto de los distintos modos de uso del programa de análisis de ensayos

En esta sección se discutirá el efecto de los tres «modos» de uso vistos en el programa de análisis de ensayos en la turbina (PARTE III, sección 2.4). La selección de cada uno de los mismos puede verse en el extracto de código 21.

A continuación, se muestra un ensayo genérico en la turbina en donde se han realizado cambios en el ángulo de incidencia del viento sobre las palas de la turbina para observar la variación de velocidad de la misma. El ensayo en sí carece de valor físico, pero es útil para explicar conceptos. Los ajustes del programa han sido:

Tmin	0
Tmax	Superior al tiempo de ensayo
Origen la primera conmutación del sensor Hall	No
Modo	Un imán por pala sin compensación (1)
Corrección de errores de muestreo	Sí
Interpolación de la velocidad del viento	Lineal
Interpolación de la velocidad de la turbina	Lineal

Tabla 4. Ajustes de un ensayo con el ventilador alejado de la turbina variando el ángulo de incidencia del viento sobre las palas de la turbina

Las gráfica a comentar son las de velocidad y aceleración de la turbina. Para obtener las mismas se ha aplicado la corrección del error de muestreo, ya que se producían picos de velocidad y no interesa ver aquí el efecto de los mismos.

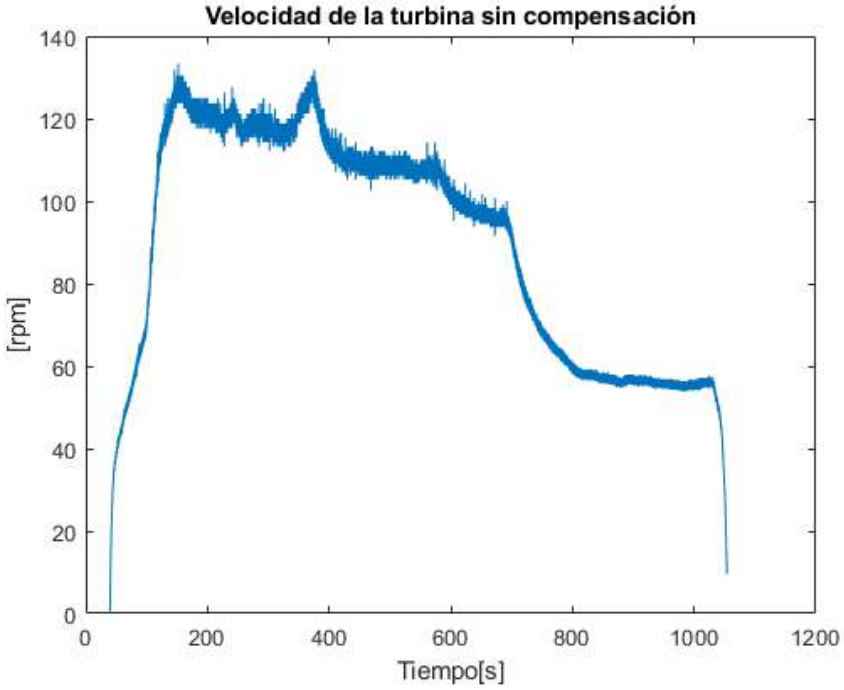


Figura 36. Velocidad de la turbina con cinco imanes, con el ventilador alejado de la turbina variando el ángulo de incidencia del viento sobre las palas de la turbina y con corrección de errores de muestreo

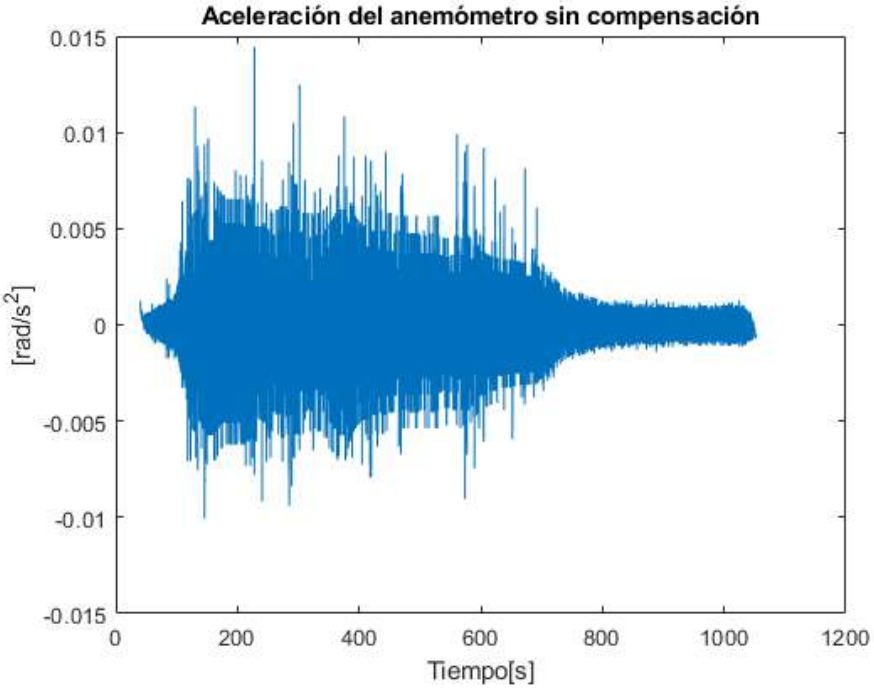


Figura 37. Velocidad de la turbina con cinco imanes, con el ventilador alejado de la turbina variando el ángulo de incidencia del viento sobre las palas de la turbina y con corrección de errores de muestreo

Se puede apreciar un rizado en la velocidad a lo largo de todo el ensayo en la primera gráfica. Este repercute directamente en la aceleración, puesto que la misma mide la pendiente entre velocidades adyacentes en el tiempo, siendo la mayoría de ellas muy elevadas debido a este efecto. La causa tras la aparición del rizado reside en la construcción de la turbina y el método empleado para obtener la velocidad.

Este último se basa en almacenar diferencias de tiempos de conmutación entre palas para realizar los cálculos. El tiempo de obtención de los datos depende de la velocidad de la turbina, es decir, a mayor velocidad angular más datos son recibidos en el mismo periodo de tiempo, por lo que es imposible conocer la posición angular de la turbina para instantes de tiempo determinados. De esta manera, el funcionamiento del sistema de toma de datos de la turbina es similar al de un «encoder» o «codificador rotatorio» pero no el mismo, ya que estos, en la mayoría de modelos, realizan la toma de datos con una frecuencia fija, obteniendo la posición angular del eje en cada instante para después calcular la velocidad y la aceleración [24].

Dicha diferenciación produce que se disponga de una resolución irregular en la obtención de los datos, lo cual implica una menor precisión en un principio, pudiéndose solventar esto mediante la interpolación de variables para obtener una aproximación. Sin embargo, obtener los datos de esta forma conlleva el riesgo de errar en las velocidades al no disponer de la posición angular exacta del eje de la turbina.

En este caso, se supuso al principio del proyecto que el ángulo entre las palas era el mismo para todas ellas, de tal forma que cada vez que se detectase una conmutación significaría que la turbina había realizado un quinto de vuelta, calculándose el resto de variables en consecuencia. Los primeros ensayos que se realizaron con la turbina probaron que esto era falso, ya que una pequeña diferencia en los ángulos entre las palas provocaba un error no despreciable al final del radio situado entre cada pala y el eje.

Para solucionar este problema se decidió crear un modo de uso del programa de toma de datos denominado de «compensación de palas» ya explicado con anterioridad y denominado como **«modo = 2»** en la sección de opciones del programa. Este calcula la velocidad de cada pala consigo misma empleando la diferencia de tiempo entre cada vuelta; tras esto, interpola los datos de velocidad angular en cada vector y finalmente, calcula la media aritmética de los cinco vectores en cada instante de tiempo. Las aceleraciones se obtienen de este último vector.

A continuación, se muestran las gráficas de velocidad angular y aceleración del caso anterior con la «compensación de palas» activada. Puede comprobarse como el rizado en la velocidad se ha visto reducido, con el debido cambio en los valores de aceleración que eso conlleva. Sin embargo, pese a que mediante la activación de este sistema los valores de velocidad se encuentran más cercanos a la realidad, no ocurre lo mismo con los de aceleración. Evidentemente, ya no aparecen picos de aceleración debidos al rizado de la velocidad, pero sí aparecen valores en función de pendientes que no se corresponden con la realidad ya que los puntos donde se han tomado las cinco distintas velocidades están situados de forma arbitraria e irregular a lo largo de una revolución y en función de esto tienen más repercusión sobre ciertos instantes de tiempo al calcular la ponderación de velocidades.

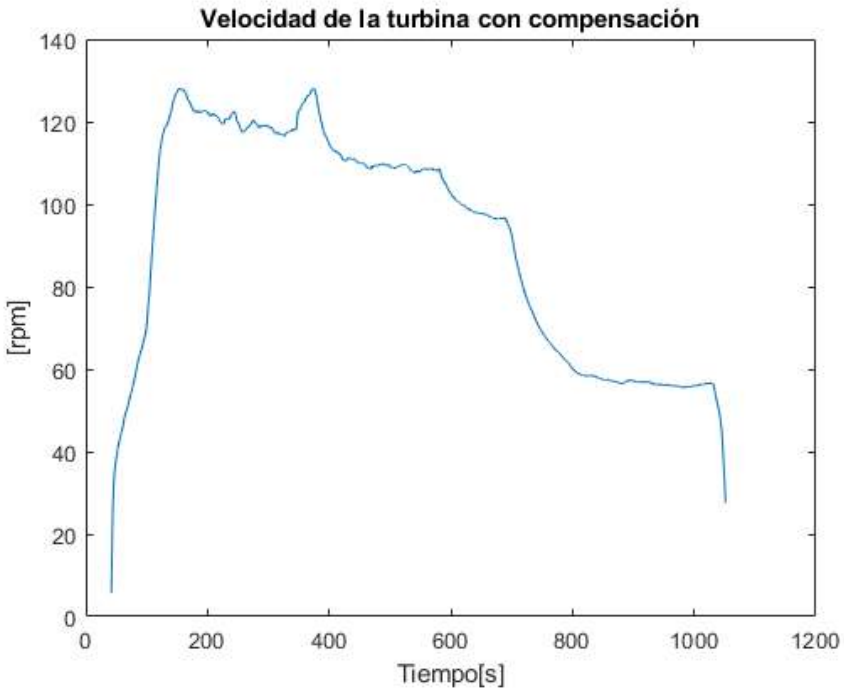


Figura 38. Velocidad de la turbina con cinco imanes y con compensación de palas, con el ventilador alejado de la turbina variando el ángulo de incidencia del viento sobre las palas de la turbina y con corrección de errores de muestreo

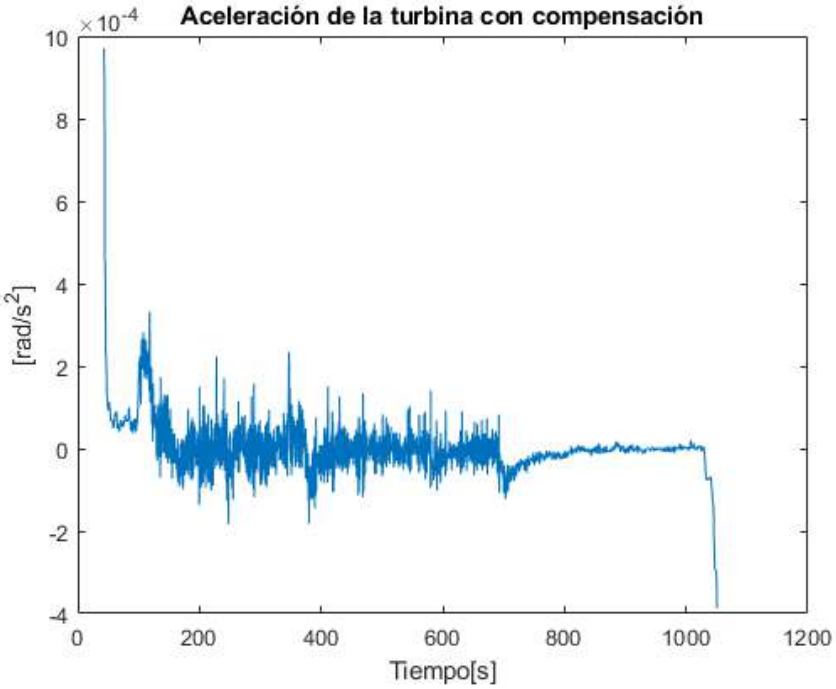


Figura 39. Velocidad de la turbina con cinco imanes y con compensación de palas, con el ventilador alejado de la turbina variando el ángulo de incidencia del viento sobre las palas de la turbina y con corrección de errores de muestreo

Puesto que la «compensación de palas» podía inducir a errores relacionados con la aceleración y en el muestreo de velocidades bajo el efecto de las vibraciones de la carcasa de la turbina y,

a su vez, el hecho de tener cinco imanes aumentaba las probabilidades de que se produjesen errores en el muestreo de la velocidad, se decidió instalar solamente un imán en una pala de la turbina y eliminar el resto. Como consecuencia de esto, se crea el **«modo = 3»**, del que se pueden ver los resultados asociados a la tabla 2 con anterioridad.

Aunque de esta manera se disponga de una menor frecuencia de obtención de datos, todas las vibraciones que puedan aparecer en las gráficas se corresponden con la realidad y no dependen de la posición en el espacio de los puntos de muestreo de la turbina, ya que este es siempre el mismo. El único cambio a tener en cuenta, si se emplea un solo imán, es en la constante **(kh)**, empleada en el programa principal de Arduino y descrita en la ecuación (11), la cual ha de ser multiplicada por 5.

4.4. Análisis de funcionamiento del programa de estimación de pérdidas mecánicas de la turbina

Para comprobar el funcionamiento del programa de estimación de pérdidas, es necesario disponer de una curva de frenado en ausencia de momento de fuerza externo o con un momento conocido en cada momento del tiempo; para poder simplificar procedimientos se ha optado por la primera opción. Los parámetro de las curvas de las que se ha dispuesto se muestran en las dos tablas siguientes:

Tmin	118 000 ms
Tmax	250 000 ms
Origen la primera conmutación del sensor Hall	Sí
Modo	Un imán en total (3)
Corrección de errores de muestreo	No
Interpolación de la velocidad del viento	Lineal
Interpolación de la velocidad de la turbina	Lineal

Tabla 5. Ajustes de la primera curva de frenado

Tmin	108 900 ms
Tmax	264 100 ms
Origen la primera conmutación del sensor Hall	Sí
Modo	Un imán en total (3)
Corrección de errores de muestreo	No
Interpolación de la velocidad del viento	Lineal
Interpolación de la velocidad de la turbina	Lineal

Tabla 6. Ajustes de la segunda curva de frenado

A continuación, se muestran las velocidades y aceleraciones de dichos ensayos. En estas se puede observar como, aunque aparentemente la curva de velocidad parece que sufre un descenso uniforme, existe una oscilación en los valores de aceleración. Esto se debe a las vibraciones que presenta la turbina a lo largo de cada ensayo. Pese a esto se observa como las mismas siguen cierta tendencia ascendente a lo largo del tiempo.

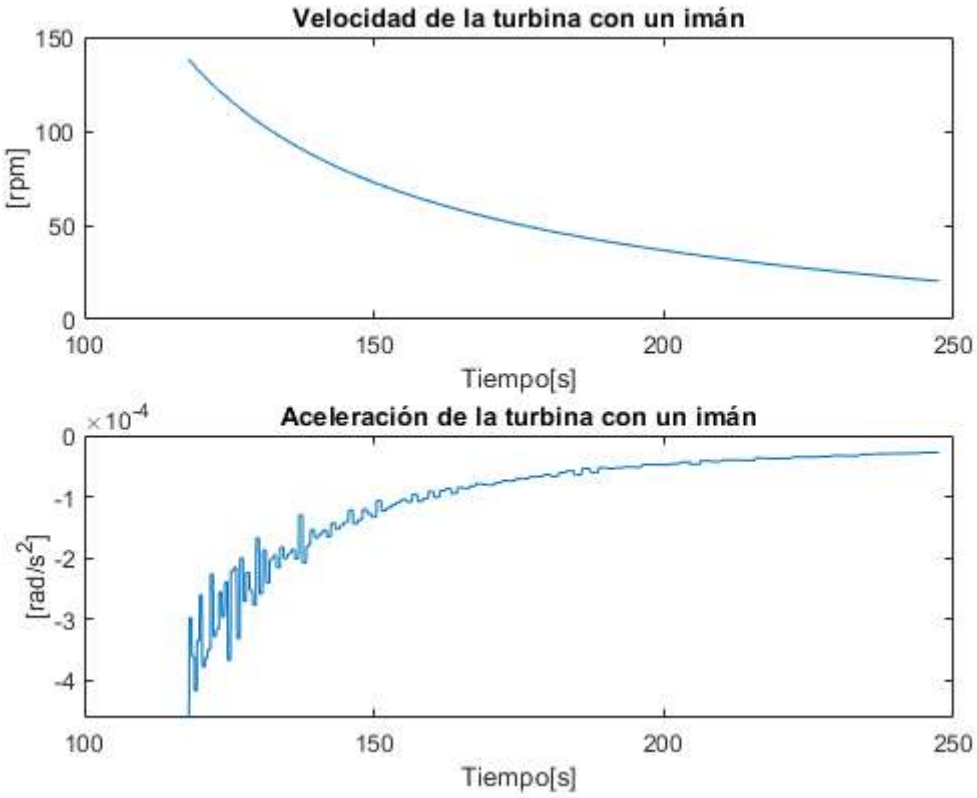


Figura 40. Velocidad y aceleración de la turbina durante la primera curva de frenado

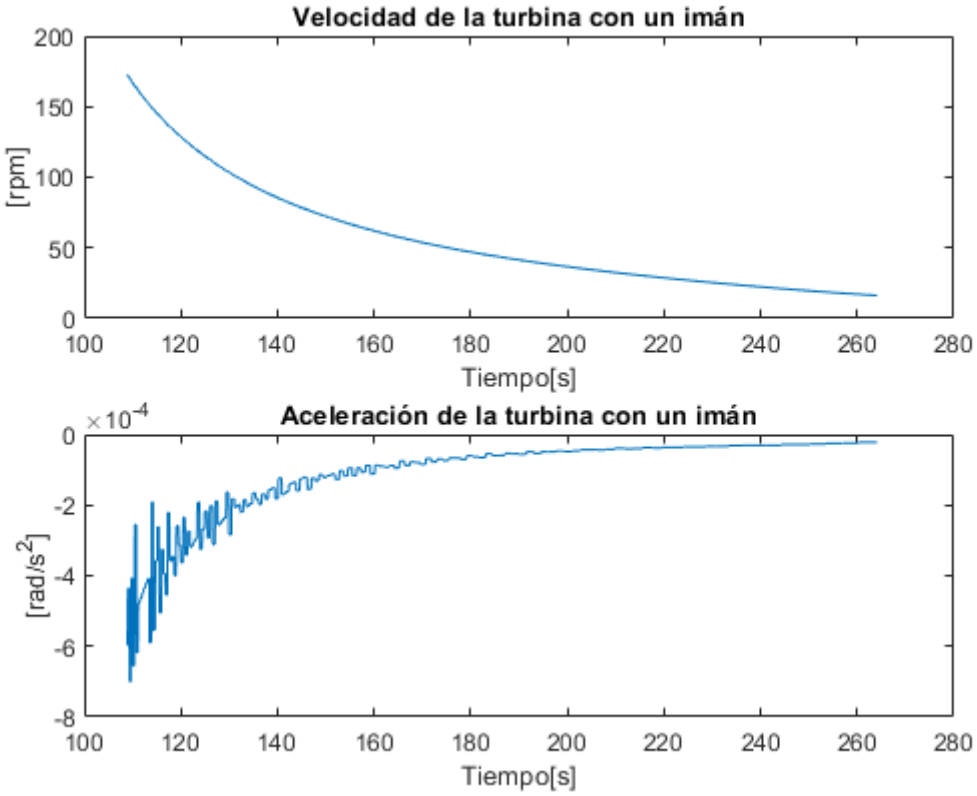


Figura 41. Velocidad y aceleración de la turbina durante la segunda curva de frenado

Al final de la página se muestran los resultados del programa para los diez primeros coeficientes de pérdidas en ambos ensayos. Los mismos se han calculado tres veces para cada uno de ellos utilizando un número determinado de puntos y por tanto de ecuaciones, es decir: para los ensayos de 50 y 100 puntos existen coeficientes de orden superior al de grado 9 que no se muestran en la tabla.

Se puede observar como los valores de coeficientes oscilan entre números positivos y negativos a lo largo de cada ensayo, siendo la magnitud de estos números mayor cuantos más puntos tenga el sistema. Esto es debido a las oscilaciones en la aceleración que aparecen durante el ensayo, en donde el programa es capaz de detectarlas e intenta modelarlas mediante los puntos de muestreo que obtiene al azar, en función de en qué instante comience el ensayo. Es decir, este programa modela los coeficientes en función de las pérdidas mecánicas por fricción que posee la turbina, pero también en función de las vibraciones a las que es sometida, las cuales distorsionan las medidas. Para obtener los coeficientes debidos exclusivamente a la fricción de los rodamientos y con el aire sería necesario eliminar el «ruido» que aparece en la aceleración, ya que sin este, la misma sigue una tendencia, al igual que la velocidad. Para esto se podrían modificar los valores que supusieran picos en la aceleración, pero, mediante el empleo de este método, se estarían alterando los datos de medida y no se tendría manera de saber hasta que punto los mismos seguirían siendo fiables.

Aun así, una vez modificados los datos se podría utilizar el programa para realizar cambios constructivos en la turbina, comparando la magnitud de los coeficientes de pérdidas, siempre que estos cambios no conllevasen mayores vibraciones en la misma, las cuales también son problemáticas. Independientemente de todo esto, los coeficientes debidos a la fricción han de ser unicamente positivos, ya que los mismos se oponen siempre al movimiento, independientemente de la velocidad. Si alguno de los términos es negativo y de una magnitud similar a la del resto de términos positivos, implica que se están produciendo vibraciones no despreciables.

Ensayo	**1**			**2**		
Puntos	**10**	**50**	**100**	**10**	**50**	**100**
B0	0.002735	-0.705412	-7.754650	0.000339	0.029621	0.449104
B1	-0.005909	1.259135	10.256032	-0.000830	-0.090022	-1.673475
B2	0.005499	0.260903	17.934530	0.000857	0.105781	2.768279
B3	-0.002885	-2.876830	-60.762148	-0.000486	-0.045056	-2.664812
B4	0.000939	3.970017	77.881513	0.000167	-0.025326	1.632065
B5	-0.000197	-3.076077	-61.041053	-3.6e-05	0.048146	-0.639705
B6	2.6e-05	1.603355	33.080344	4.9e-06	-0.034081	0.1403634
B7	-2.2e-06	-0.602491	-13.064217	-4e-07	0.0150912	-0.000452
B8	1e-07	0.168695	3.850253	1.8e-08	-0.004648	-0.011537
B9	-1.9e-09	0.035641	-0.851810	-3.2e-10	0.001032	0.004481

Tabla 7. Resultados de la estimación de pérdidas mecánicas de la turbina

4.5. Análisis de funcionamiento del programa para el cálculo de la curva de seguimiento óptimo de la turbina

Para el análisis de resultados del programa para el cálculo de la curva de seguimiento óptimo se emplearán cuatro curvas de arranque, en función de las distintas velocidades de los ventiladores, siendo la primera de ellas con los ventiladores alejados y en su mínima velocidad, mientras que para el resto se encontrarán próximos a la turbina, siendo la velocidad lo que cambie. Las potencias de estas curvas serán las obtenidas de combinar las medidas de momento de inercia, aceleración y velocidad de la turbina en cada instante, es decir: la potencia mecánica en el eje sin conectar. Para el ensayo real se debería emplear potencia eléctrica. Por eso los ensayos no tienen sentido físico y en ellos se ha limitado el tiempo máximo hasta las zonas en donde las aceleraciones comenzaban a oscilar lo suficiente como para que apareciesen potencias negativas, cuyo significado físico reside tras las vibraciones de la turbina. Así pues, los parámetros de cada ensayo han sido:

Tmin	0
Tmax	40 000 ms
Origen la primera conmutación del sensor Hall	Sí
Modo	Un imán en total (3)
Corrección de errores de muestreo	Sí
Interpolación de la velocidad del viento	Lineal
Interpolación de la velocidad de la turbina	Spline

Tabla 8. Ajustes de la primera curva de arranque

Tmin	0
Tmax	25 000 ms
Origen la primera conmutación del sensor Hall	Sí
Modo	Un imán en total (3)
Corrección de errores de muestreo	Sí
Interpolación de la velocidad del viento	Lineal
Interpolación de la velocidad de la turbina	Spline

Tabla 9. Ajustes de la segunda curva de arranque

Tmin	0
Tmax	20 000 ms
Origen la primera conmutación del sensor Hall	Sí
Modo	Un imán en total (3)
Corrección de errores de muestreo	Sí
Interpolación de la velocidad del viento	Lineal
Interpolación de la velocidad de la turbina	Spline

Tabla 10. Ajustes de la tercera curva de arranque

Tmin	0
Tmax	20 000 ms
Origen la primera conmutación del sensor Hall	Sí
Modo	Un imán en total (3)
Corrección de errores de muestreo	Sí
Interpolación de la velocidad del viento	Lineal
Interpolación de la velocidad de la turbina	Spline

Tabla 11. Ajustes de la cuarta curva de arranque

Mientras que las distintas gráficas de cada ensayo se muestran a continuación:

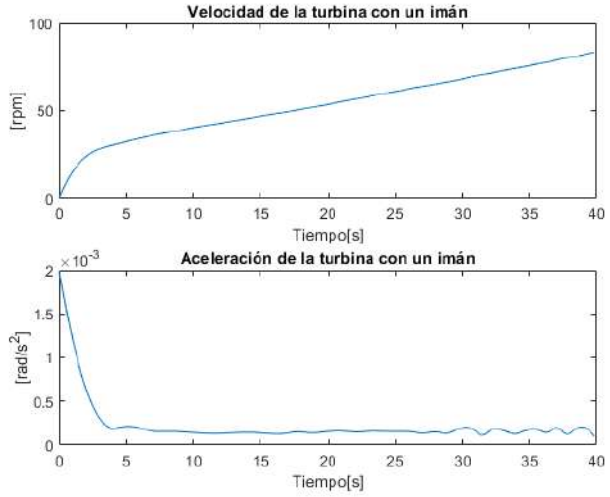


Figura 42. Velocidad y aceleración de la turbina durante la primera curva de arranque

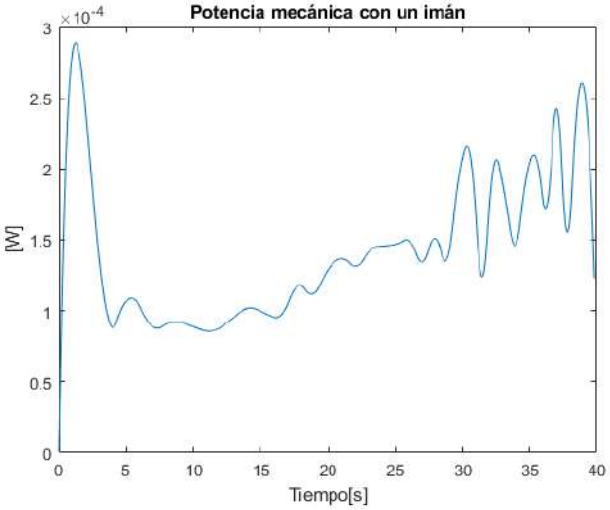


Figura 43. Potencia mecánica en el eje de la turbina durante la primera curva de arranque

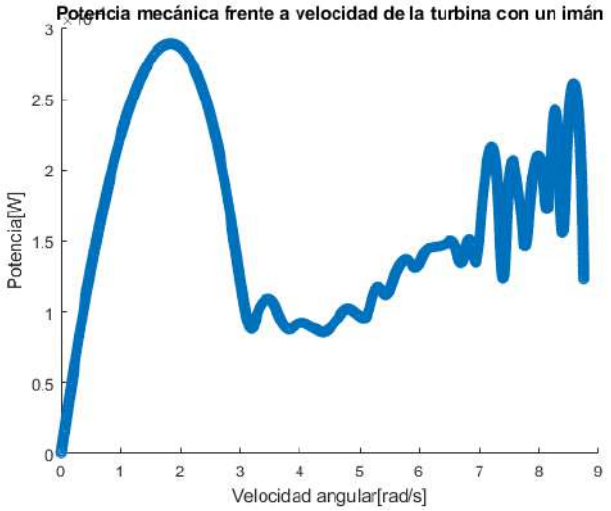


Figura 44. Potencia mecánica en el eje de la turbina frente a velocidad de la turbina durante la primera curva de arranque

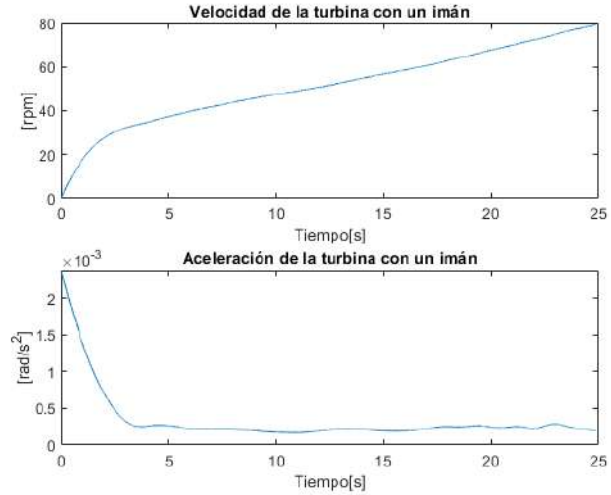


Figura 45. Velocidad y aceleración de la turbina durante la segunda curva de arranque

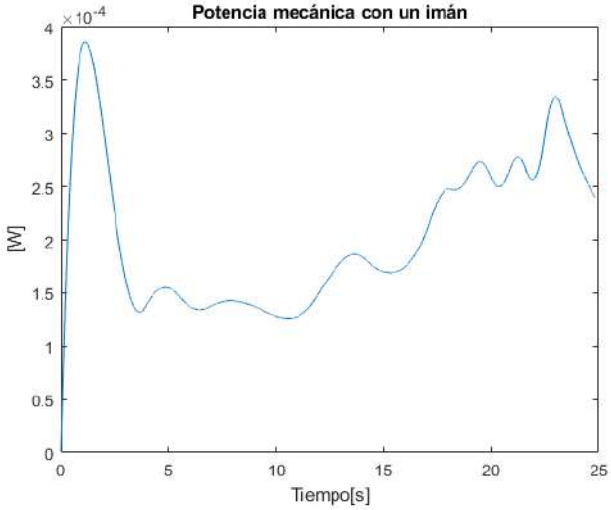


Figura 46. Potencia mecánica en el eje de la turbina durante la segunda curva de arranque

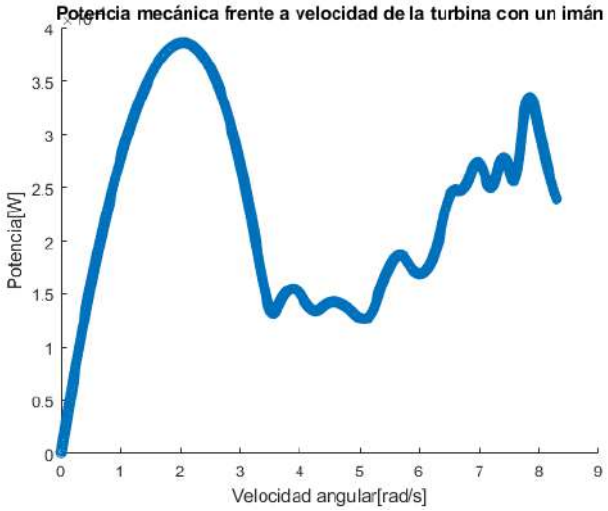


Figura 47. Potencia mecánica en el eje de la turbina frente a velocidad de la turbina durante la segunda curva de arranque

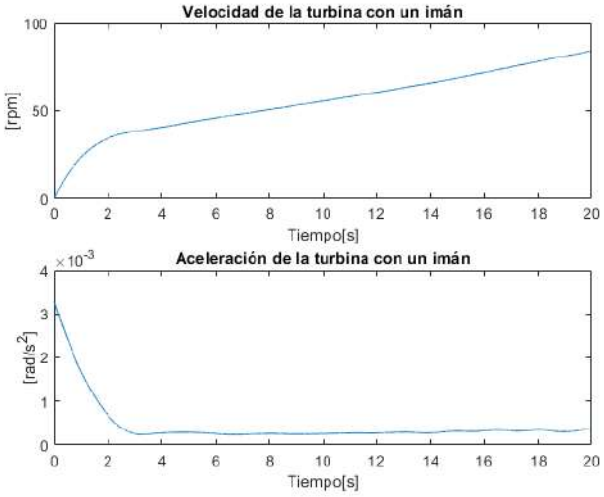


Figura 48. Velocidad y aceleración de la turbina durante la tercera curva de arranque

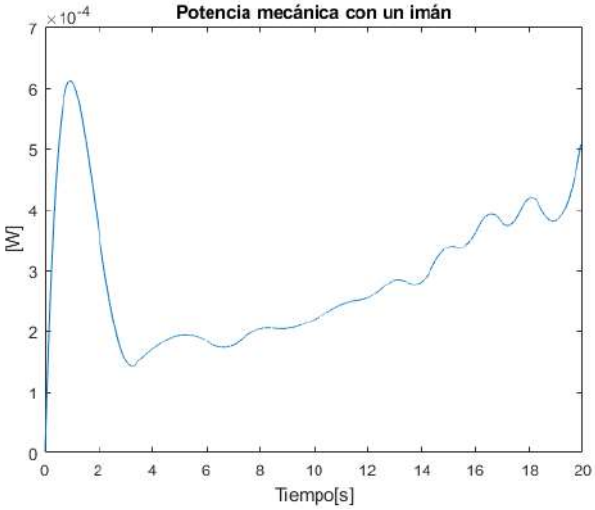


Figura 49. Potencia mecánica en el eje de la turbina durante la tercera curva de arranque

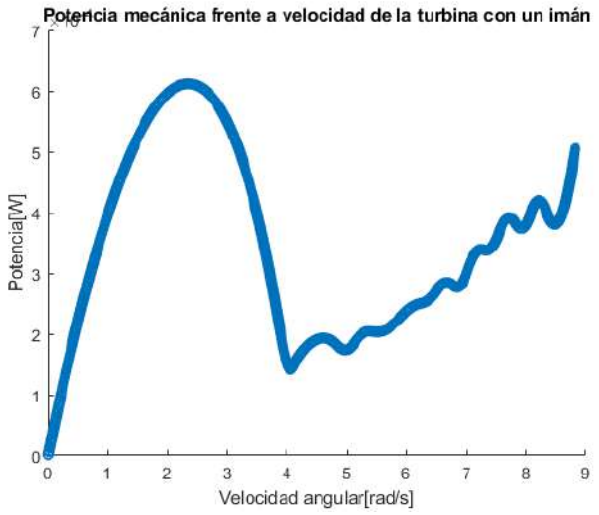


Figura 50. Potencia mecánica en el eje de la turbina frente a velocidad de la turbina durante la tercera curva de arranque

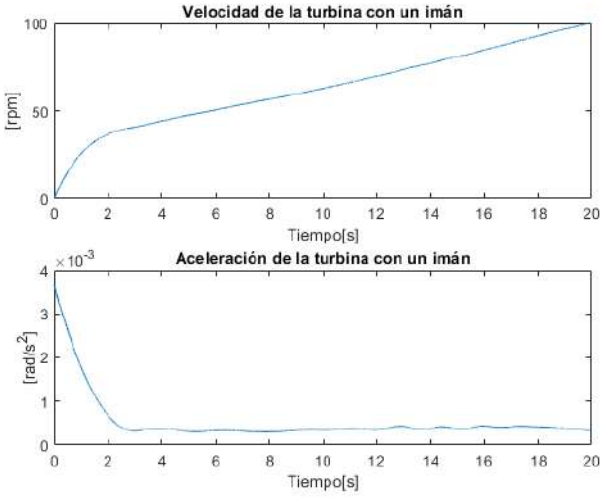


Figura 51. Velocidad y aceleración de la turbina durante la cuarta curva de arranque

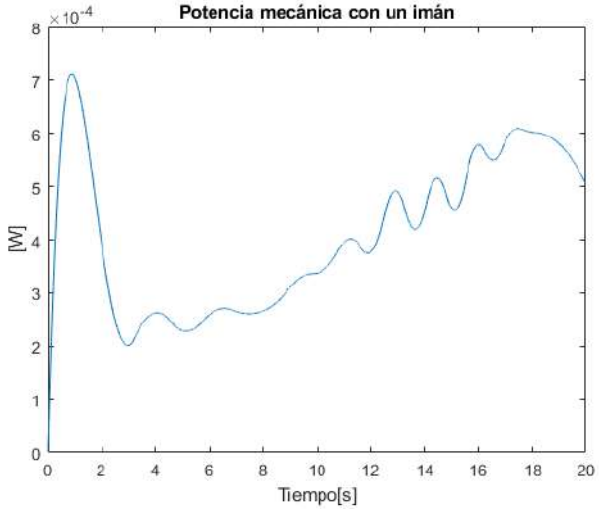


Figura 52. Potencia mecánica en el eje de la turbina durante la cuarta curva de arranque

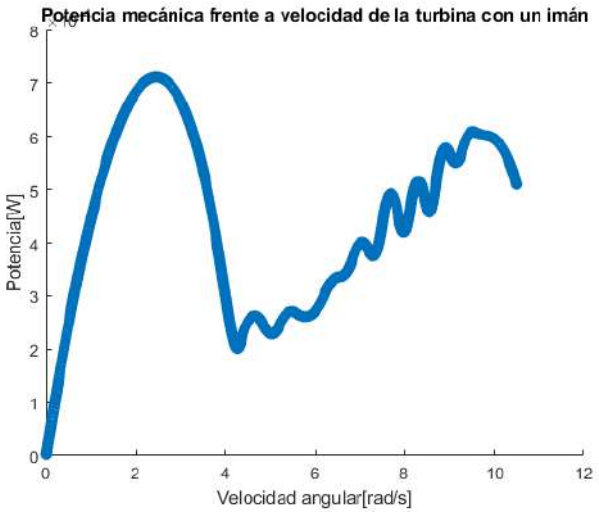


Figura 53. Potencia mecánica en el eje de la turbina frente a velocidad de la turbina durante la cuarta curva de arranque

Tras obtener todas las curvas de potencia frente a velocidad angular, el programa almacena los puntos máximos y los une definiendo la potencia para el resto de los puntos de velocidad con una resolución especificada.

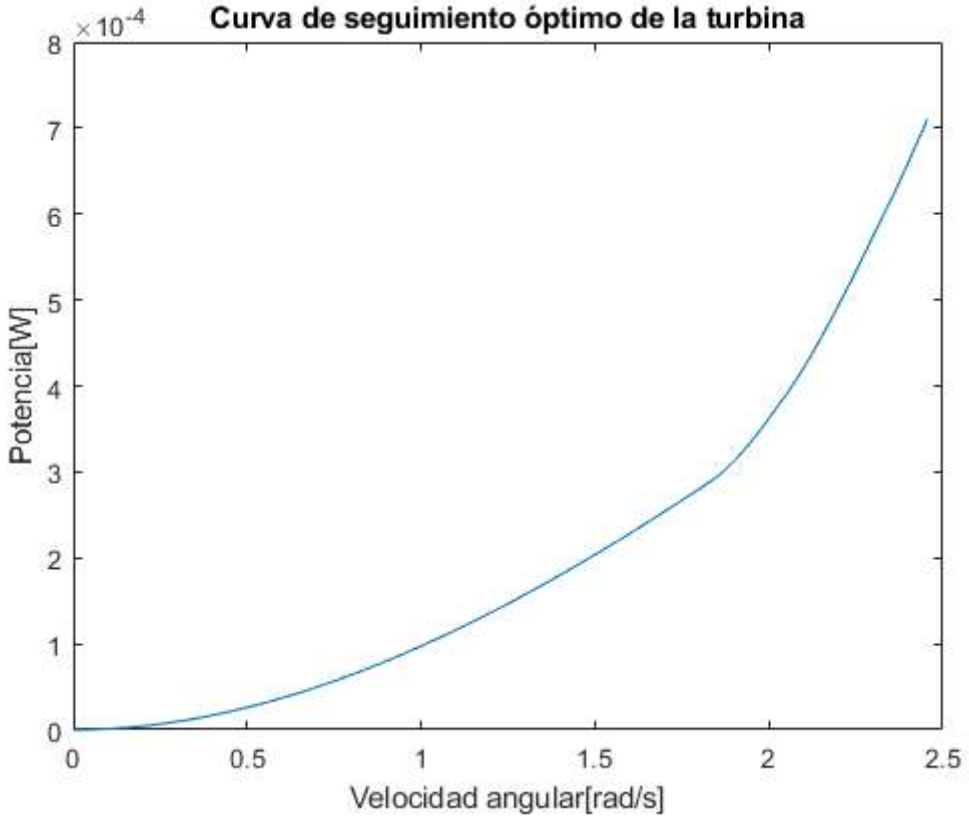


Figura 54. Resultado del programa para la obtención de la curva de seguimiento óptimo

En este caso, los valores de potencia son muy bajos ya que proviene de la potencia mecánica extraída de unos ventiladores. Aun así, se puede concluir que el programa funciona correctamente ya que solamente sería necesario sustituir los valores de potencia por los obtenidos en el tiempo

a partir de un vatímetro. Por último, pueden visualizarse los puntos de interpolación de la curva contenidos en la matriz **(M)** a continuación:

Curva de arranque	**0**	**1**	**2**	**3**	**4**
Velocidad media del viento [m/s]	0	4.516042	7.142443	7.157960	8.069332
Potencia máxima [W]	0	0.000289	0.000386	0.000612	0.000711
Velocidad angular de la turbina [rad/s]	0	1.832327	2.043114	2.344916	2.456813

Tabla 12. Matriz **(M)**

Capítulo 5

Conclusiones

A lo largo de este proyecto se han ido relatando los primeros pasos seguidos para desarrollar un modelo de aerogenerador de uso doméstico desde el principio. Inicialmente, se encontraba un prototipo de turbina con cinco imanes y un sensor de efecto Hall instalado, así como la placa Arduino a la que se conectaba.

Ha sido necesario diseñar un programa capaz de medir los tiempos de conmutación de un anemómetro y del sensor Hall, calcular las velocidades derivadas de estos y activar un servomotor para frenar la turbina en caso de que fuera necesario. A su vez, el alumno Javier Colinas Cano se ha encargado del envío de los datos de tiempo a Matlab mediante *bluetooth* y de su procesamiento en tiempo real. Una vez se disponía de los datos de tiempos de conmutación almacenados, se ha diseñado un programa capaz de calcular las velocidades del viento y de la turbina en esos instantes de tiempo para después extrapolarlas a cada instante del ensayo, con una resolución de [ms], la cual es la máxima de la que dispone Arduino para contar tiempo. Con los valores de velocidad en la turbina se ha podido calcular la aceleración.

A su vez, se ha creado un programa capaz de calcular el centro de masas de un perfil de ala dado mediante las coordenadas de diversos puntos de su contorno. Este ha sido empleado para facilitar la toma de medidas que permiten, mediante otro programa desarrollado en este proyecto, calcular el momento de inercia de la turbina. Este momento de inercia ha sido empleado, junto a la aceleración, para poder calcular el momento de fuerza al que se ve sometido la turbina cuando no se encuentra su eje conectado a ninguna máquina eléctrica. Mediante el momento de inercia, la aceleración y la velocidad de la turbina se ha diseñado un programa para el cálculo de los factores de pérdidas de la misma. Finalmente, sustituyendo lo que debería ser potencia eléctrica generada por potencia mecánica en el eje, se ha comprobado el funcionamiento de otro programa empleado para el cálculo de la curva de seguimiento óptimo, que sería necesario instalar en los dispositivos de control vectorial.

5.1. Metodología

La metodología seguida a lo largo del proyecto ha sido de constante ensayo y error, ya que no se encuentran muchos estudios sobre el desarrollo de un proyecto de estas características. Así pues, se han creado herramientas que han de ser útiles para la continuación del proyecto principal: comercializar un modelo de aerogenerador inteligente para uso doméstico. Se ha explicado el funcionamiento de dichas herramientas y la motivación tras el mismo. Finalmente, la bibliografía, además de servir para hacer referencia a los documentos de los que se ha obtenido

información, cuenta con una lista muy útil de proyectos similares de los que seguir aprendiendo para continuar con el desarrollo iniciado con este proyecto.

5.2. Resultados

Los resultados del proyecto, por regla general, han sido favorables. Dentro de los elementos relacionados con la placa Arduino se puede asegurar que la misma es perfectamente capaz de hacer frente a los valores numéricos que ha de gestionar, sin errar en su magnitud y siendo lo suficientemente precisa como para que los mismos dispongan de utilidad. A su vez, cuenta con la suficiente rapidez como para visualizar adecuadamente los datos en tiempo real. A esto habría que añadirle su bajo precio.

El anemómetro es capaz de captar adecuadamente la velocidad del viento con una frecuencia de muestreo del orden de la centésima de segundo. Se compararon las medidas con las de un anemómetro manual de mayor precisión y los valores medios solamente se diferenciaban en unas pocas décimas. El anemómetro de cazoletas también consigue que se visualicen las variaciones de velocidad que el mismo presenta a lo largo de una revolución, por el efecto de frenado que ofrecen las palas al ir a contracorriente. Si se quiere diferenciar este efecto de las variaciones que puedan existir en las ráfagas de viento será necesario sustituir este modelo por uno que tenga un número de conmutaciones por vuelta que sea un múltiplo del número de palas del mismo.

El sensor Hall erraba ocasionalmente saltándose alguna de las conmutaciones de los imanes. Esto ocurría a altas velocidades se la turbina. No se sabe si debido a que, con la alta velocidad, uno de los imanes era capaz de pasar por el sensor antes de que este hubiese acabado de enviar la señal del imán anterior o si se producía debido a vibraciones que pudieran alejar uno de los imanes del sensor de manera puntual y aleatoria. La primera opción podría ser descartada en favor de la segunda, ya que los errores de muestreo se producían a velocidad terminal de forma ocasional y no continuada, cuando se podía observar un rizado en la velocidad debido a vibraciones; sin embargo no se dispone de pruebas concluyentes. El resto de valores los enviaba adecuadamente y sin dar ningún otro problema.

El servomotor respondía cuando se superaban las velocidades límite especificadas en el programa de Arduino, sin embargo, cuando el módulo *bluetooth* se encontraba conectado, presentaba problemas de alimentación que producían que el par motor no fuese el adecuado y oscilase. Esto se puede solucionar conectando el mismo a una fuente de alimentación externa, ya que el Arduino no dispone de suficiente potencia. Aun así, el método de control del freno es de lazo abierto, por lo que cuando la velocidad se encuentra alrededor del límite especificado el freno tiende a abrirse y cerrarse continuamente. Independientemente del funcionamiento del servomotor, el sistema de freno quedó pendiente de su colocación en la carcasa así como de su comprobación con la turbina en movimiento.

Los programas para el cálculo del centro de masas de la pala y el momento de inercia de la turbina no se ajustan exáctamente a la realidad del prototipo pero ofrecen una buena aproximación. Además pueden ser muy útiles para comparar momentos de inercia en función de los cambios en el perfil de ala que se realicen en la turbina. A medida que el diseño de las palas vaya siendo perfeccionado estos programas pueden ofrecer una mayor precisión; si los modelos futuros de las palas son macizos, como hasta ahora, pero con un interior más homogéneo los programas podrán utilizarse sin ofrecer variaciones y respondiendo con una mayor precisión;

si los futuros modelos son huecos podrá emplearse el principio de superposición, realizándose pequeños cambios en los mismos ya que estos operan discretizando el perfil en puntos.

El programa de muestreo de datos en Matlab funciona adecuadamente para todas sus opciones de uso y además dispone de suficiente flexibilidad para seguir adaptándose a nuevas funciones en el futuro, como por ejemplo incluir la potencia eléctrica medida con un vatímetro para ciertos instantes de tiempo e incluirla en los cálculos. Para obtener adecuadamente los coeficientes de pérdidas de la turbina será necesario reducir las vibraciones o filtrar la aceleración como ya se indicó en la sección 4.4. Mientras que para obtener la curva de seguimiento óptimo será necesario emplear la potencia eléctrica para cada velocidad de la turbina con distintos valores de velocidad del viento.

5.3. Recomendaciones para futuros estudios

En función de los resultados expuestos a con anterioridad se recomienda:

- Buscar un modelo de anemómetro con un número de conmutaciones que sea múltiplo del número de palas. Con esto se podrán detectar variaciones repentinas en las rachas de viento. No es imprescindible puesto que las variables relacionadas con el viento van en función de su velocidad media a lo largo del tiempo.
- Sustituir o complementar el sensor de efecto Hall con un «encoder» o «codificador rotatorio» conectado al eje de la turbina para disponer de una frecuencia de muestreo constante.
- Incluir una fuente de alimentación independiente para el servomotor del freno o sustituir el mismo por un freno mecánico de inercia.
- Realizar un estudio sobre las vibraciones sufridas por la turbina, intentando reducir la magnitud de las mismas.
- Profundizar en el estudio para la obtención de los coeficientes de pérdidas de la turbina, con independencia las vibraciones y con el objetivo de optimizar los mismos.
- Diseñar e implementar un dispositivo capaz de medir valores de tensión e intensidad, para instantes de tiempo dados, en donde se puedan asociar los mismos a valores de velocidad instantánea de la turbina y del anemómetro.
- Buscar modelos de motores eléctricos y modelos de dispositivos de electrónica de potencia de control con los que se puedan realizar los ensayos de potencia frente a velocidad una vez acoplados a la turbina.
- Realizar un estudio económico de la rentabilidad de dichos modelos.

Bibliografía

[1] **RED ELÉCTRICA DE ESPAÑA. (2018).** *El sistema eléctrico español 2018 [archivo PDF].* Recuperado de https://www.ree.es/sites/default/files/11_PUBLICACIONES/Documentos/InformesSistemaElectrico/2018/inf_sis_elec_ree_2018.pdf

[2] **eurostat Statistics Explained. (2019).** *Estadísticas de los precios de la electricidad.* Recuperado de https://ec.europa.eu/eurostat/statistics-explained/index.php/Electricity_price_statistics/es#Precios_de_la_electricidad_para_los_consumidores_dom.C3.A9sticos

[3] **IDAE, Eurostat. (2011).** *Proyecto SECH-SPAHOUSEC [archivos PDF].* Recuperado de https://www.idae.es/uploads/documentos/documentos_Informe_SPAHOUSEC_ACC_f68291a3.pdf https://www.idae.es/uploads/documentos/documentos_Documentacion_Basica_Residencial_Unido_c93da537.pdf

[4] **eurostat Statistics Explained. (2019).** *Energy consumption in households.* Recuperado de https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Energy_consumption_in_households#Energy_consumption_in_households_by_type_of_end-use

[5] **Ley 24/2013, de 26 de diciembre, del Sector Eléctrico.**
Publicado en: «BOE» núm. 310, de 27/12/2013.
Entrada en vigor: 28/12/2013
Departamento: Jefatura del Estado
Referencia: BOE-A-2013-13645
Permalink ELI: https://www.boe.es/eli/es/l/2013/12/26/24/con

[6] **Real Decreto 900/2015, de 9 de octubre, por el que se regulan las condiciones administrativas, técnicas y económicas de las modalidades de suministro de energía eléctrica con autoconsumo y de producción con autoconsumo.**
Publicado en: «BOE» núm. 243, de 10/10/2015.
Entrada en vigor: 11/10/2015
Departamento: Ministerio de Industria, Energía y Turismo
Referencia: BOE-A-2015-10927
Permalink ELI: https://www.boe.es/eli/es/rd/2015/10/09/900/con

[7] **Real Decreto-ley 15/2018, de 5 de octubre, de medidas urgentes para la transición energética y la protección de los consumidores.**
Publicado en: «BOE» núm. 242, de 6 de octubre de 2018, páginas 97430 a 97467 (38 págs.)
Sección: I. Disposiciones generales
Departamento: Jefatura del Estado
Referencia: BOE-A-2018-13593
Permalink ELI: https://www.boe.es/eli/es/rdl/2018/10/05/15

[8] **Real Decreto 244/2019, de 5 de abril, por el que se regulan las condiciones administrativas, técnicas y económicas del autoconsumo de energía eléctrica.**
Publicado en: «BOE» núm. 83, de 6 de abril de 2019, páginas 35674 a 35719 (46 págs.)
Sección: I. Disposiciones generales
Departamento: Ministerio para la Transición Ecológica
Referencia: BOE-A-2019-5089
Permalink ELI: https://www.boe.es/eli/es/rd/2019/04/05/244

[9] **Martín Santana, L. (2017).** *Fiscalidad del autoconsumo eléctrico de fuentes renovables (Tesis Doctoral).* Universidad Carlos III de Madrid.
http://hdl.handle.net/10016/26446

[10] **IDAE. (2019).** *El Gobierno aprueba el Real Decreto por el que se regulan las condiciones del autoconsumo.* Recuperado de
https://www.idae.es/noticias/el-gobierno-aprueba-el-real-decreto-por-el-que-se-regulan-las-condiciones-del-autoconsumo

[11] **IDAE, EnerAgen. (2019).** *Guía Profesional de Tramitación del Autoconsumo [archivo PDF].* Recuperado de
https://www.idae.es/publicaciones/guia-profesional-de-tramitacion-del-autoconsumo

[12] **Real de la Barreda, I. (2019).** *Autoconsumo fotovoltaico en España tras el Real Decreto 244/2019. Ejemplos de instalaciones (Tesis de Master).* Universidad Politécnica de Madrid.
http://oa.upm.es/62537/

[13] **Calefacción SOLAR. (2018).** *Cómo elegir baterías para paneles solares.* Recuperado de
http://calefaccion-solar.com/como-elegir-baterias-para-paneles-solares.html

[14] **Twenergy, el portal de eficiencia de Endesa. (2019).** *Aerogeneradores domésticos, ¿una alternativa rentable?* Recuperado de
https://twenergy.com/energia/energia-eolica/aerogeneradores-domesticos-1807/l

[15] **Siemens.** *Motores Siemens Trifásicos.* Recuperado de
https://www.motoressiemens.com/motores-siemens-trifasicos.html

[16] **ABB.** *Motores de baja tensión IEC.* Recuperado de https://new.abb.com/motors-generators/es/motores-de-baja-tension-iec

[17] **WEG.** *Motores para Aplicación Industrial.* Recuperado de https://www.weg.net/catalog/weg/ES/es/Motores-El%C3%A9ctricos/Motores-para-Aplicaci%C3%B3n-Industrial/c/EU_MT_LV_IEC_SPECIALAPPLICATION

[18] **Siemens.** *Low Voltage Converters.* Recuperado de https://new.siemens.com/global/en/products/drives/sinamics/low-voltage-converters.html

[19] **ABB.** *DTC: Control vectorial avanzado de convertidores de frecuencia ABBC.* Recuperado de https://new.abb.com/drives/es/dtc

[20] **Ramos Cenzano, Á. (2014).** *Análisis mediante cálculo numérico (CFD) del comportamiento de anemómetros de cazoletas (Proyecto Fin de Carrera).* Universidad Politécnica de Madrid. http://oa.upm.es/32407/

[21] **UIUC Airfoil Coordinates Database.** *(clarky-il) CLARK Y AIRFOIL.* Recuperado de http://www.airfoiltools.com/airfoil/details?airfoil=clarky-il

[22] **Journal of Electrical Systems. (2007).** *Generators for Wind Energy Conversion Systems: State of the Art and Coming Attractions [archivo PDF].* Recuperado de http://journal.esrgroups.org/jes/papers/3_1_3.pdf

[23] **Carmona Sanz, M. (2016).** *Calificación cualitativa del estado de salud de un aerogenerador de un parque eólico a través del análisis de su curva de potencia y factores que influyen en ella (Proyecto Fin de Carrera).* Universidad Pontificia Comillas. http://hdl.handle.net/11531/17727

[24] **Sepúlveda García, G. (2015).** *Integración de encoders absolutos en el control distribuido de Manfred 3 (Proyecto Fin de Carrera).* Universidad Carlos III de Madrid. http://hdl.handle.net/10016/23334

PARTE II

OBJETIVOS DE DESARROLLO SOSTENIBLE

Capítulo 1

Objetivos de Desarrollo Sostenible

ESTE proyecto tiene como objetivo último la comercialización de un dispositivo capaz de generar energía eléctrica a través de una fuente renovable y libre de emisiones de CO_2, esta es el viento. Otro aspecto importante del mismo es la flexibilidad que ha de ofrecer para situar el mismo en cualquier localización, debido a su pequeño tamaño. A su vez, también es importante de cara al futuro el coste reducido que el modelo de aerogenerador ha de presentar para que sea más rentable su uso que obtener la energía directamente de la red. Es por estos tres motivos por los que el proyecto se desarrolla en corcondancia y favoreciendo ciertos «Objetivos de Desarrollo Sostenible» especificados por la ONU. Estos son diecisiete objetivos, adoptados por todos los Estados Miembros en 2015, y cuyo cumplimiento absoluto está planeado para el año 2030. Los mismos se muestran a continuación:

Figura 55. «Objetivos de Desarrollo Sostenible» según la ONU

De todos los objetivos, este proyecto hace una alusión directa en mayor o menor medida a los seis siguientes:

- 7. Energía asequible y no contaminante.
- 8. Trabajo decente y crecimiento económico.
- 9. Industria, innovación e infraestructura.
- 11. Ciudades y comunidades sostenibles.
- 12. Producción y consumo responsables.
- 13. Acción por el clima.

1.1. 7. Energía asequible y no contaminante.

El séptimo objetivo tiene como finalidad principal mejorar la productividad de aquellas fuentes de energía no contaminantes, en vista del incremento del acceso de la población mundial a fuentes de electricidad; a su vez, se busca que las mismas sean accesibles para que su uso pueda extenderse a nivel global. Datos como que actualmente una séptima parte de los habitantes del planeta no tiene acceso a electricidad, debido a los emplazamientos en lo que viven, carentes de infraestructuras para el transporte de la energía, o que, su vez, el 40 % de la población mundial depende de combustibles contaminantes e insalubres para cocinar, así como que el 60 % de las emisiones globales de gases de efecto invernadero son consecuencia de la producción de energía hacen que este sea un objetivo cuyo cumplimiento depende en su mayor parte de la tecnología.

El proyecto aquí presente se encuentra estrechamente relacionado con este objetivo, siendo uno de los principales a tener en cuenta. Si el modelo de «Aerogenerador Inteligente» hiciera efectiva su comercialización podría ser capaz de dotar de energía eléctrica a aquellas zonas remotas en donde se dieran rachas de viento normales, así como hacer más rentable el acceso a la energía, gracias a la competencia por ofrecer precios más bajos, en los lugares con acceso a redes eléctricas.

1.2. 8. Trabajo decente y crecimiento económico

El octavo objetivo tiene como finalidad principal la creación de empleo sostenible y de calidad mediante el aumento de los niveles de productividad, fomentando para ello la innovación tecnológica. Los motivos tras la importancia de este objetivo son el aumento de la desigualdad; la tasa global de desempleo, la cual alcanza el 5 %; el porcentaje de trabajadores que viven por debajo del umbral de la pobreza, es decir, con menos de 3.20$ internacionales al día, el cual corresponde aproximadamente con el 20 % de la población activa; y la gran proporción de trabajadores con empleos informales en los últimos años, de alrededor del 61 %.

El grado de relación de este objetivo con el proyecto sería secundario, puesto que no supone un cambio estructural en las condiciones materiales frente a esta realidad. Aun así, sí que supondría un aumento en el número de puestos de trabajo totales al generar un modelo de negocio viable y rentable, para el que se necesitan trabajadores formados con un perfil técnico.

1.3. 9. Industria, innovación e infraestructura

El noveno objetivo tiene como finalidad principal la mejora de infraestructuras mediante la aplicación de avances tecnológicos. Con esto se busca una mejora general en las condiciones de vida, un modelo de producción sostenible y un acceso más igualitario al conocimiento y a las oportunidades. Algunos de los problemas sobre los que se centra este objetivo son: el acceso de la población mundial a internet, en donde actualmente aproximadamente un 51 % de la población no dispone de él; el acceso a saneamiento básico y agua potable, del que carecen en torno al 29 % y al 10 % de la población, respectivamente; y la falta de acceso permanente a electricidad, de alrededor del 33 %.

El nivel de implicación de este proyecto con dicho objetivo es secundario ya que solo tiene en cuenta la mejora de las infraestructuras eléctricas, otorgando un mayor grado de accesibilidad a este tipo de energía. Sin embargo, este es un aspecto muy importante para un gran número de personas, por lo que mejorar la tecnología en este ámbito puede ser crucial.

1.4. 11. Ciudades y comunidades sostenibles

El undécimo objetivo tiene como finalidad principal la transformación de los entornos urbanos en lugares habitables, seguros, asequibles y sostenibles para la vida humana. Esto surge como consecuencia del aumento constante de la población mundial, así como de la concentración de más del 55 % de la misma en ciudades, ocupando solamente el 3 % de la superficie terrestre pero consumiendo el entre el 60 y el 80 % del total de la energía.

Este proyecto tiene un grado de implicación alto en relación a este objetivo. Esto es debido al cambio de paradigma que supondría este tipo de generación en las ciudades, haciendo autosuficientes a la mayor parte de las mismas y rebajando los costes de vida en general.

1.5. 12. Producción y consumo responsables

El duodécimo objetivo tiene como finalidad principal reducir la huella ecológica derivada de la producción de bienes industriales y de consumo. Pese al gran impacto ecológico producido por el consumo se calcula que una gran parte de la población no ve sus necesidades cubiertas a través de este. Otro dato importante son las cantidades desperdiciadas de alimento, estimadas en 1300 millones de toneladas al año, así como los altos niveles de consumo de energético.

Este proyecto se ve ligeramente relacionado con este objetivo al abogar por el «autoconsumo» de energía eléctrica, en donde el propio consumidor es más consciente de la importancia de la misma y de su ahorro.

1.6. 13. Acción por el clima

El decimotercer objetivo tiene como finalidad principal reducir el efecto del cambio climático producido por las emisiones de gases de efecto invernadero, atacando tanto a sus causas como a sus consecuencias. Se calcula que la temperatura media de la tierra ha subido $1\,^{\circ}\mathrm{C}$ desde la revolución industrial, si no se desea que esta aumente otro grado y medio más, las emisiones

deben disminuir en un 45 % antes de 2030 y ser nulas para 2050. De no cumplirse esto se estima que el nivel del mar subirá alrededor de $30\,\mathrm{cm}$.

El proyecto se encuentra fuertemente relacionado con este objetivo puesto que en él se presenta un sistema de generación eléctrica a través de una fuente de energía limpia y renovable. Es decir, presenta un elemento técnico de apoyo a todas las medidas en favor de retrasar el cambio climático.

PARTE III

CÓDIGO FUENTE

Capítulo 1

Arduino

1.1. Programa principal

```
#include <Servo.h> // Incluye la librería necesaria para el servomotor del freno
#include <SoftwareSerial.h>   // Incluimos la librería SoftwareSerial
SoftwareSerial BT(10,11); // Define los pines RX y TX del Arduino conectados al Bluetooth (necesario para la función de envío de datos)
Servo freno; // Crea un objeto para controlar el servomotor del freno


// Variables

// Sensor hall
int ah; // Comprobación de inicio de conmutación en el sensor hall
unsigned long toh=0, tfh, dth, kh=12000, vh, vhmax=200;
// Variables no empleadas
// unsigned long vh2, dvh, aah;

// Anemómetro
int aan; // Comprobación de inicio de conmutación en el anemómetro
unsigned long toa=0, tfa, dta, ka=666667, va, vamax=10000;
// Variables no empleadas
// unsigned long va2, dva, aa;

// Freno
int pos, poso=90, poslims=180, poslimi=90;


// Programa principal (Main)

// Inicialización
```

```
28 void setup()
29 {
30   BT.begin(9600); // Inicializamos el puerto serie BT (Para
        Modo AT 2) (necesario para la función de envío de datos)
31   Serial.begin(9600);
32
33   pinMode(2, INPUT); // Asigna el pin 2 al sensor Hall
34   pinMode(3, INPUT); // Asigna el pin 3 al anemómetro
35   freno.attach(4); // Asigna el pin 4 al servomotor del freno
36
37   freno.write(poso); // Coloca el freno en la posición inicial
38
39   delay (1000); // Espera 1s antes de iniciar el bucle
40 }
41
42 // Bucle principal
43 void loop()
44 {
45   if(digitalRead(3)==HIGH) // Si el anemómetro no detecta
        conmutación
46   {
47     aan=1;
48     servo(); // Ejecuta la función de movimiento del freno
49   }
50
51   if(digitalRead(2)==HIGH) // Si el sensor hall no detecta
        conmutación
52   {
53     ah=1;
54     servo(); // Ejecuta la función de movimiento del freno
55   }
56
57   if(digitalRead(3)==LOW) // Si el anemómetro detecta
        conmutación
58   {
59     if(aan==1) // Si la conmutación es detectada justo después
          de la ausencia de conmutación del anemómetro
60     {
61       tfa = millis(); // Guarda el tiempo de la conmutación
            actual del anemómetro
62       dta = tfa-toa; // Calcula el diferencial de tiempo
            respecto a la conmutación anterior del anemómetro
63       toa = millis(); // Guarda el tiempo de la conmutación
            actual para ser empleado en la siguiente conmutación
            del anemómetro
64
65       if(dta!=0) // Si el diferencial de tiempo del anemómetro
            no es 0
66       {
```

```
67        va = ka/dta; // Calcula la velocidad del viento
68        //dva = va-va2;
69        //va2 = va;
70        //aa = dva/dta;
71        }
72        else // Si el diferencial de tiempo del anemómetro es 0
73        {
74          va = 9999; // Fija una velocidad del viento muy alta
75          Serial.print("error va"); // Indica mediante un mensaje
                en el "Serial" que se ha producido un error en el
              anemómetro
76          //dva = va-va2;
77          //va2 = va;
78          //aa = 10e6;
79        }
80
81        // Indica en el "Serial" la velocidad del viento
82        Serial.print("va:");
83        Serial.print(va);
84        Serial.println('\t');
85
86        enviar(1); //Función de envío de datos (desarrollada por
             Javier Colinas Cano)
87      }
88
89      aan=0; // Asegura que no se vuelva a calcular el
           diferencial de tiempo hasta la siguiente conmutación
           real del anemómetro
90
91      servo(); // Ejecuta la función de movimiento del freno
92    }
93
94    if(digitalRead(2)==LOW) // Si el sensor hall detecta
         conmutación
95    {
96      if(ah==1) // Si la conmutación es detectada justo después
           de la ausencia de conmutación del sensor hall
97      {
98        tfh = millis(); // Guarda el tiempo de la conmutación
             actual del sensor hall
99        dth = tfh-toh; // Calcula el diferencial de tiempo
             respecto a la conmutación anterior del sensor hall
100       toh = millis(); // Guarda el tiempo de la conmutación
             actual para ser empleado en la siguiente conmutación
             del sensor hall
101
102       if(dth!=0) // Si el diferencial de tiempo del sensor hall
             no es 0
103       {
```

```
104         vh = kh/dth; // Calcula la velocidad angular de la
                turbina
105         //dvh = vh-vh2;
106         //vh2 = vh;
107         //aah = dvh/dth;
108       }
109       else // Si el diferencial de tiempo del sensor hall es 0
110       {
111         vh = 199; // Fija una velocidad de la turbina muy alta
112         Serial.print("error vh");  // Indica mediante un
                mensaje en el "Serial" que se ha producido un error
                en el sensor hall
113         //dvh = vh-vh2;
114         //vh2 = vh;
115         //aah = 10e6;
116       }
117
118       // Indica en el "Serial" la velocidad angular de la
             turbina
119       Serial.print("vh:");
120       Serial.print(vh);
121       Serial.println('\t');
122
123       enviar(2); //Función de envío de datos (desarrollada por
             Javier Colinas Cano)
124     }
125
126     ah=0; // Asegura que no se vuelva a calcular el diferencial
          de tiempo hasta la siguiente conmutación real del
         sensor hall
127
128     servo(); // Ejecuta la función de movimiento del freno
129   }
130 }
```

1.2. Función de movimiento del freno

```
1 // Función de movimiento del freno
2 void servo()
3 {
4   pos = freno.read(); // Detecta la posición actual del
       servomotor
5
6   if((va>=vamax || vh>=vhmax) && pos<poslims) // Si la
       velocidad del viento o la velocidad angular de la turbina
       superan o igualan los valores límite establecidos y la
       posición del servomotor no es la de frenado
7   {
```

```
8     // pos +=1; Inicialmente se encotraba así, por ello el
         servomotor tenía problemas puesto que había
         perturbaciones en su movimiento y en el par.
9     pos = poslims; // El servomotor se sitúa en posición de
         frenado. Javier Colinas Cano sugirió este cambio y ahora
          funciona corréctamente.
10  }
11
12  if(va<vamax && vh<vhmax && pos>poslimi) // Si la velocidad
       del viento y la velocidad angular de la turbina son
       inferiores a los valores límite establecidos y la posición
        del servomotor no es la de máxima apertura
13  {
14    // pos -=1;
15    pos = poslimi; // El servomotor se sitúa en posición de
         apertura
16  }
17
18  freno.write(pos); // Envía la señal correspondiente al
       servomotor
19 }
```

1.3. Función de envío de datos (desarrollada por Javier Colinas Cano)

```
1 //Función de envío de datos (desarrollada por Javier Colinas
     Cano)
2 void enviar(int e)//esta función recibe un 1 o un 2 según si el
     dato a enviar es del anemómetro o del sensor de efecto hall
     respectivamente.
3 {
4   //El módulo bluetooth HC-05 solo puede enviar datos que
       ocupen 8 bits como máximo, por tanto, para poder enviar
       los datos mayores se usarán variables auxiliares para
       permitir el envío de dato completo.
5   static int d;
6   static int f;
7   if (e==1)
8   {
9     f=tfa>>16;
10    d=tfa>>8;
11    BT.write(65);//se envía un número que al ser recibido en
         matlab se lee según el código ASCII y se traduce como
         una A el 65 y una H el 72, haciendo referencia a los
         datos recibidos del anemómetro y del sensor de efecto
         Hall respectivamente.
12    BT.write(f);
```

```
13      f = 0; //se actualizan las variables para evitar posibles
            errores.
14      BT.write(d);
15      d = 0;
16      BT.write(tfa);
17    }
18    else
19    {
20      f=tfh>>16;
21      d=tfh>>8;
22      BT.write(72);
23      BT.write(f);
24      f = 0;
25      BT.write(d);
26      d = 0;
27      BT.write(tfh);
28    }
29  }
```

Capítulo 2

Matlab

2.1. Cálculo del centro de masas del perfil de ala

```
%% Extracción de datos del perfil de ala del fichero de Excel a
   Matlab

xs = xlsread('ClarkY.xlsx','A:A'); % Coordenadas en x de la curva
   superior
ys = xlsread('ClarkY.xlsx','B:B'); % Coordenadas en y de la curva
   superior
xi = xlsread('ClarkY.xlsx','C:C'); % Coordenadas en x de la curva
   inferior
yi = xlsread('ClarkY.xlsx','D:D'); % Coordenadas en y de la curva
   inferior

%% Ajuste de vectores

% Resolución en el eje x
r = 5e-4;

% Reescalado de vectores
X = 0:r:1; % Vector de puntos en el eje x

X = X';

YS = interp1(xs,ys,X,'spline'); % Interpolación de los puntos del
   eje y de la curva superior del perfil para las coordenadas en x
YI = interp1(xi,yi,X,'spline'); % Interpolación de los puntos del
   eje y de la curva inferior del perfil para las coordenadas en x

X = X';
YS = YS';
YI = YI';

nX = numel(X); % Numero de componentes del vector de posición del
   eje x

%% Cálculo de la posición del centro de masas de las tapas

```

```
% Eje x
n = 1;
while (n<=nX)
    dmx(n) = YS(n)-YI(n); % Diferencial de masa ficticio en el eje x
    n = n+1;
end

mf = sum(dmx); % Masa ficticia total del perfil

n = 1;
xg = 0;
while (n<=nX)
    xg = xg+dmx(n)*X(n); % Ponderación de la posición de cada
       diferencial en el eje x por el valor de masa ficticia de cada
       uno de los mismos
    n = n+1;
end
XGp = xg/mf; % Posición del centro de masas del perfil en el eje x

% Eje y
n = 1;
while (n<=nX)
    ygdmx(n) = (YI(n)+YS(n))/2; % Posición del centro de masas en el
       eje y de cada diferencial de masas del eje x
    n = n+1;
end

n = 1;
yg = 0;
while (n<=nX)
    yg = yg+dmx(n)*ygdmx(n); % Ponderación de la posición de cada
       diferencial en el eje y por el valor de masa ficticia de cada
       uno de los mismos
    n = n+1;
end
YGp = yg/mf; % Posición del centro de masas del perfil en el eje y

%% Dibujo

figure
plot(X,YS,X,YI,XGp,YGp,'b--o')
title('Perfil de ala clarky-il')
legend('Curva superior','Curva inferior','Centro de masas')
```

2.2. Cálculo del momento de inercia de la turbina

```
%% Parámetros

npg = 5; % Número de palas del aerogenerador
nbu =2; % Número de barras de unión entre cada pala y el eje

```

```
6  mp = 459e-3; % Masa de cada pala en [kg]
7  mbu = 19e-3; % Masa de cada barra de unión entre eje y pala en [kg]
8  me = 156e-3; % Masa del eje de la turbina en [kg]
9
10 Lp = 220e-3; % Longitud del eje horizontal del perfil de ala en [m]
11
12 Dpg = 245e-3; % Distancia del centro de masas de la pala al eje de
       la turbina en [m]
13
14 Re = 8e-3; % Radio exterior del eje de la turbina en [m]
15 Ri = 7e-3; % Radio interior del eje de la turbina en [m]
16
17 Lbu = 245e-3; % Longitud de cada barra de unión entre eje y pala en
       [m]
18
19 %% Ajuste de vectores
20
21 % Resolución
22 r = 1e-4;
23
24 % Reescalado de vectores
25 X = 0:r:1; % Vector de puntos en el eje x
26
27 X = X';
28
29 YS = interp1(xs,ys,X); % Interpolación de los puntos del eje y de la
       curva superior del perfil para las coordenadas en x
30 YI = interp1(xi,yi,X); % Interpolación de los puntos del eje y de la
       curva inferior del perfil para las coordenadas en x
31
32 X = X';
33 YS = YS';
34 YI = YI';
35
36 nX = numel(X); % Numero de componentes del vector de posición del
       eje x
37
38 % Cambio del origen del sistema de referencia al centro de masas
39 n=1;
40 while(n<=nX)
41    X(n) = X(n)-XGp;
42    n = n+1;
43 end
44
45 n=1;
46 while(n<=nX)
47    YS(n) = YS(n)-YGp;
48    YI(n) = YI(n)-YGp;
49    n = n+1;
50 end
51
52 % Reescalado de vectores
```

```
53 YS = round(YS,4); % Aproximación de los valores en y de la curva
       superior del perfil a 4 decimales
54 YI = round(YI,4); % Aproximación de los valores en y de la curva
       inferior del perfil a 4 decimales
55
56 %% Cálculo momento de inercia de la turbina
57
58 % Momento de inercia de la pala respecto al centro de masas de la
       pala
59 Izp = 0;
60 n1 = 1;
61 n2 = 1;
62 n3 = 0;
63 while(n1<=nX)
64    Yn = YI(n1):r:YS(n1); % División del perfil de ala en los puntos
         comprendidos entre las curvas superior e inferior con una
         resolución r
65    nYn = numel(Yn); % Numero de componentes del vector de posición
         del eje y (número de puntos en los que se discretiza el perfil
         de ala)
66    while(n2<=nYn)
67        Izp = Izp + ((Lp*Yn(n2))^2+(Lp*X(n1))^2); % Momento de
            inercia del perfil de ala en función de la longitud del
            eje horizontal y sin tener en cuenta la masa
68        n2 = n2+1;
69    end
70    n3 = n3+n2-1;
71    n2 = 1;
72    n1 = n1+1;
73 end
74
75 dmp = mp/n3; % Masa de cada punto en los que se divide el perfil de
       ala
76 Izp = Izp*dmp;
77
78 % Momento de inercia del eje de la turbina respecto al eje de la
       turbina
79 Ize = (1/2)*me*(Re^2+Ri^2);
80
81 % Momento de inercia de cada barra de unión entre el eje y las palas
       al eje de la turbina
82 Izbu = (1/3)*mbu*Lbu^2;
83
84 % Momento de inercia total de la turbina respecto al eje de la
       turbina en [kg*m^2]
85 Jg = Ize+npg*nbu*Izbu+npg*Izp+npg*mp*Dpg^2;
```

2.3. Inicialización de los ensayos para obtener la curva de seguimiento óptimo de la turbina

```
%% Inicialización de los ensayos de curva característica

ne = 2; % Inicializa el índice del número de ensayos realizados para distintas velocidades del viento
M = [0;0;0]; %Almacena las variables velocidad media del viento, potencia mecánica máxima y velocidad angular de la turbina para esa potencia en una matriz inicializándolas en 0
```

2.4. Ensayo en la turbina

```
%% Inicialización

clearvars -except Jg ne M % Elimina todas las variables excepto el momento de inercia de la turbina y la inicialización de los ensayos de curva característica(Evita errores)
close all % Cierra las gráficas (Evita confusiones con gráficas de ensayos distintos)

%% Parámetros físicos del aerogenerador

% Anemómetro
Ra = 70e-3; % Radio desde el eje del anemómetro a el centro de la pala en [m]
coma = 2; % Número de conmutaciones por vuelta que realiza el anemómetro
ca = 200/(21*pi); % Factor de cazoleta

% Generador
Rg = 265e-3; % Distancia a la que se encuentra el imán que detecta el sensor hall del eje del generador en [m]
npg = 5; % Número de conmutaciones por vuelta que realiza el sensor hall (número de palas del aerogenerador)

%% Opciones del programa

Tmin = 0; % Tiempo mínimo de muestreo en [ms] (No funciona con "modo=2")
Tmax = 300000; % Tiempo máximo de muestreo en [ms]
origen = 1; % Tiempo inicial en primera conmutación de cualquier dispositivo(0), tiempo inicial en primera conmutación del sensor hall(1)
modo = 3; % Sin compensación de palas(1), con compensación de palas(2), con un imán en total(3)
correccion = 1; % Sin corrección de error de muestreo(0), con corrección de error de muestreo(1)
ncorreccion5 = 0.91; % Factor de corrección de error de muestreo para cinco imanes
ncorreccion1 = 0.51; % Factor de corrección de error de muestreo para un imán
```

```
26 interpolaciona = 'lineal'; % Selecciona el tipo de interpolación en
       la velocidad del viento
27 interpolaciong = 'spline'; % Selecciona el tipo de interpolación en
       la velocidad de la turbina
28
29 %% Carga de vectores de tiempo
30
31 load('un_iman_pot3_cerca.mat')
32
33 %% Fijar el origen de tiempos en 0
34
35 % Contabilizar el número de conmutaciones en cada dispositivo
36 na = numel(ta);
37 nh = numel(th);
38
39 % Si se quiere que el origen de tiempo se sitúe en la primera
       conmutación de cualquier dispositivo
40 if(origen==0)
41     if (ta(1)<th(1))
42         ta0 = ta(1);
43         n=1;
44         while(n<=na)
45             ta(n) = ta(n)-ta0;
46             n = n+1;
47         end
48         n=1;
49         while(n<=nh)
50             th(n) = th(n)-ta0;
51             n = n+1;
52         end
53     end
54
55     if (ta(1)>th(1))
56         th0 = th(1);
57         n=1;
58         while(n<=na)
59             ta(n) = ta(n)-th0;
60             n = n+1;
61         end
62         n=1;
63         while(n<=nh)
64             th(n) = th(n)-th0;
65             n = n+1;
66         end
67     end
68
69     if (ta(1)==th(1))
70         ta0 = ta(1);
71         n=1;
72         while(n<=na)
73             ta(n) = ta(n)-ta0;
74             n = n+1;
```

```
75          end
76          n=1;
77          while(n<=nh)
78              th(n) = th(n)-ta0;
79              n = n+1;
80          end
81      end
82  end
83
84  % Si se quiere que el origen de tiempo se sitúe en la primera
        conmutación del sensor hall
85  if(origen==1)
86      t0 = th(1);
87
88      % Anemómeteo
89      n1 = 1;
90      n2 = 1;
91      while(n1<=na)
92          if(ta(n1)>=t0)
93              t(n2) = ta(n1); % Escribe los valores de tiempo
                    superiores al origen de conmutación del sensor hall
                    en un vector auxiliar t
94              n2 = n2+1;
95          end
96          n1 = n1+1;
97      end
98      clearvars ta; % Borra el vector de tiempo antiguo
99      ta = t; % Crea un nuevo vector de tiempos límitado al tiempo
            máximo
100     clearvars t; % Borra el vector de tiempo auxiliar t
101     na = numel(ta);
102     n = 1;
103     while(n<=na)
104         ta(n) = ta(n)-t0;
105         n = n+1;
106     end
107
108     % Sensor hall
109     n = 1;
110     while(n<=nh)
111         th(n) = th(n)-t0;
112         n = n+1;
113     end
114     nh = numel(th);
115 end
116
117 %% Limitación del tiempo máximo de muestreo
118
119 % Anemómetro
120 n1 = 1;
121 n2 = 1;
122 while(n1<=na)
```

```
123     if(ta(n1)<=Tmax)
124         t(n2) = ta(n1); % Escribe los valores de tiempo inferiores
               al tiempo máximo en un vector auxiliar t
125         n2 = n2+1;
126     end
127     n1 = n1+1;
128 end
129 clearvars ta; % Borra el vector de tiempo antiguo
130 ta = t; % Crea un nuevo vector de tiempos límitado al tiempo máximo
131 clearvars t; % Borra el vector de tiempo auxiliar t
132
133 % Sensor Hall
134 n1 = 1;
135 n2 = 1;
136 while(n1<=nh)
137     if(th(n1)<=Tmax)
138         t(n2) = th(n1); % Escribe los valores de tiempo inferiores
               al tiempo máximo en un vector auxiliar t
139         n2 = n2+1;
140     end
141     n1 = n1+1;
142 end
143 clearvars th; % Borra el vector de tiempo antiguo
144 th = t; % Crea un nuevo vector de tiempos límitado al tiempo máximo
145 clearvars t; % Borra el vector de tiempo auxiliar t
146
147 % Contabilizar el número de conmutaciones en cada dispositivo de
       nuevo
148 na = numel(ta);
149 nh = numel(th);
150
151 %% Vectores de tiempo para cada pala
152
153 if(modo==2)
154     n = 1;
155     m = 1;
156     n1 = 1;
157     n2 = 1;
158     n3 = 1;
159     n4 = 1;
160     n5 = 1;
161     while(n<=nh)
162         if(m==1)
163             th1(n1) = th(n);
164             n1 = n1+1;
165         end
166         if(m==2)
167             th2(n2) = th(n);
168             n2 = n2+1;
169         end
170         if(m==3)
171             th3(n3) = th(n);
```

```
172             n3 = n3+1;
173         end
174         if(m==4)
175             th4(n4) = th(n);
176             n4 = n4+1;
177         end
178         if(m==5)
179             th5(n5) = th(n);
180             n5 = n5+1;
181             m = 0;
182         end
183         n = n+1;
184         m = m+1;
185     end
186
187     % Contabilizar el número de conmutaciones en cada pala
188     nh1 = numel(th1);
189     nh2 = numel(th2);
190     nh3 = numel(th3);
191     nh4 = numel(th4);
192     nh5 = numel(th5);
193 end
194
195 %% Vectores de tiempo con misma referencia
196
197 if(ta(na)<th(nh))
198     T = th(nh);
199 end
200
201 if(ta(na)>th(nh))
202     T = ta(na);
203 end
204
205 if(ta(na)==th(nh))
206     T = ta(na);
207 end
208
209 t = 0:T;  % Genera un vector de tiempo en [ms]
210
211 %% Velocidades y aceleraciones
212
213 % Anemómetro
214 n = 1;
215 while (n<=na)
216     if(n==1)
217         wa(n) = 0;  % El valor de la velocidad inicial es 0
218     end
219     if(n>1)
220         wa(n) = (2*pi*1000/coma)/(ta(n)-ta(n-1));  % Velocidad
              angular del anemómetro en [rad/s]
221     end
222     n = n+1;
```

```
223 end
224 vvm = ca*Ra*wa; % Velocidad del viento en [m/s]
225 vvkm = 3.6*ca*Ra*wa; % Velocidad del viento en [km/h]
226 n=1;
227 while (n<=na)
228     if(n==1)
229         aa(n) = NaN; % El valor de la aceleración inicial no es un
                número
230     end
231     if(n>1)
232         aa(n) = (wa(n)-wa(n-1))/(ta(n)-ta(n-1)); % Aceleración
                angular del anemómetro en [rad/s^2]
233     end
234     n = n+1;
235 end
236 avm = ca*Ra*aa; % Aceleración del viento en [m/s^2]
237 avkm = 3.6*ca*Ra*aa; % Aceleración del viento en [km/h^2]
238
239 % Hall con y sin compensación de palas
240 if(modo==1 || modo==2)
241     n = 1;
242     while (n<=nh)
243         if(n==1)
244             wgHz(n) = 0; % El valor de la velocidad inicial es 0
245         end
246         if(n>1)
247             wgHz(n) = (1000/npg)/(th(n)-th(n-1)); % Velocidad
                    angular de la turbina en con 5 imanes[Hz]
248         end
249         n = n+1;
250     end
251     % Corrección de error de muestreo
252     n = 1;
253     while(n<=nh && correccion==1)
254         if(n>2)
255             if(wgHz(n-1)<=ncorreccion5*wgHz(n-2) &&
                    wgHz(n-1)<=ncorreccion5*wgHz(n))
256                 wgHz(n-1) = (wgHz(n)+wgHz(n-2))/2;
257             end
258         end
259         n = n+1;
260     end
261     %
262     wg = 2*pi*wgHz; % Velocidad angular de la turbina en [rad/s]
263     wgrpm = 60*wgHz; % Velocidad angular de la turbina en [rpm]
264     n=1;
265     while (n<=nh)
266         if(n==1)
267             agHz(n) = NaN; % El valor de la aceleración inicial no
                    es un número
268         end
269         if(n>1)
```

```
270             agHz(n) = (wgHz(n)-wgHz(n-1))/(th(n)-th(n-1)); %
                    Aceleración angular de la turbina en [Hz^2]
271         end
272         n = n+1;
273     end
274     ag = 2*pi*agHz; % Aceleración angular de la turbina en [rad/s^2]
275     agrpm = 60*agHz; % Aceleración angular de la turbina en [rpm^2]
276 end
277
278 % Hall con un imán en total
279 if(modo==3)
280     n = 1;
281     while (n<=nh)
282         if(n==1)
283             wgHz(n) = 0; % El valor de la velocidad inicial es 0
284         end
285         if(n>1)
286             wgHz(n) = 1000/(th(n)-th(n-1)); % Velocidad angular de
                    la turbina con 1 imán en [Hz]
287         end
288         n = n+1;
289     end
290     % Corrección de error de muestreo
291     n = 1;
292     while(n<=nh && correccion==1)
293         if(n>2)
294             if(wgHz(n-1)<=ncorreccion1*wgHz(n-2) &&
                    wgHz(n-1)<=ncorreccion1*wgHz(n))
295                 wgHz(n-1) = (wgHz(n)+wgHz(n-2))/2;
296             end
297         end
298         n = n+1;
299     end
300     %
301     wg = 2*pi*wgHz; % Velocidad angular de la turbina en [rad/s]
302     wgrpm = 60*wgHz; % Velocidad angular de la turbina en [rpm]
303     n=1;
304     while (n<=nh)
305         if(n==1)
306             agHz(n) = NaN; % El valor de la aceleración inicial no
                    es un número
307         end
308         if(n>1)
309             agHz(n) = (wgHz(n)-wgHz(n-1))/(th(n)-th(n-1)); %
                    Aceleración angular de la turbina en [Hz^2]
310         end
311         n = n+1;
312     end
313     ag = 2*pi*agHz; % Aceleración angular de la turbina en [rad/s^2]
314     agrpm = 60*agHz; % Aceleración angular de la turbina en [rpm^2]
315 end
316
```

```
317 % Hall por cada pala
318 if(modo==2)
319     % Pala 1
320     n = 1;
321     while (n<=nh1)
322         if(n==1)
323             wgHz1(n) = 0; % El valor de la velocidad inicial es 0
324         end
325         if(n>1)
326             wgHz1(n) = 1000/(th1(n)-th1(n-1)); % Velocidad angular
                    de la pala 1 en [Hz]
327         end
328         n = n+1;
329     end
330     % Corrección de error de muestreo
331     n = 1;
332     while(n<=nh1 && correccion==1)
333         if(n>2)
334             if(wgHz1(n-1)<=ncorreccion5*wgHz1(n-2) &&
                    wgHz1(n-1)<=ncorreccion5*wgHz1(n))
335                 wgHz1(n-1) = (wgHz1(n)+wgHz1(n-2))/2;
336             end
337         end
338         n = n+1;
339     end
340     %
341     wg1 = 2*pi*wgHz1; % Velocidad angular de la pala 1 en [rad/s]
342     wgrpm1 = 60*wgHz1; % Velocidad angular de la pala 1 en [rpm]
343     n=1;
344     while (n<=nh1)
345         if(n==1)
346             agHz1(n) = NaN; % El valor de la aceleración inicial no
                    es un número
347         end
348         if(n>1)
349             agHz1(n) = (wgHz1(n)-wgHz1(n-1))/(th1(n)-th1(n-1)); %
                    Aceleración angular de la pala 1 en [Hz^2]
350         end
351         n = n+1;
352     end
353     ag1 = 2*pi*agHz1; % Aceleración angular de la pala 1 en [rad/s^2]
354     agrpm1 = 60*agHz1; % Aceleración angular de la pala 1 en [rpm^2]
355
356     % Pala 2
357     n = 1;
358     while (n<=nh2)
359         if(n==1)
360             wgHz2(n) = 0; % El valor de la velocidad inicial es 0
361         end
362         if(n>1)
363             wgHz2(n) = 1000/(th2(n)-th2(n-1)); % Velocidad angular
                    de la pala 2 en [Hz]
```

```
364         end
365         n = n+1;
366     end
367     % Corrección de error de muestreo
368     n = 1;
369     while(n<=nh2 && correccion==1)
370         if(n>2)
371             if(wgHz2(n-1)<=ncorreccion5*wgHz2(n-2) &&
                    wgHz2(n-1)<=ncorreccion5*wgHz2(n))
372                 wgHz2(n-1) = (wgHz2(n)+wgHz2(n-2))/2;
373             end
374         end
375         n = n+1;
376     end
377     %
378     wg2 = 2*pi*wgHz2;
379     wgrpm2 = 60*wgHz2;
380     n=1;
381     while (n<=nh2)
382         if(n==1)
383             agHz2(n) = NaN; % El valor de la aceleración inicial no
                    es un número
384         end
385         if(n>1)
386             agHz2(n) = (wgHz2(n)-wgHz2(n-1))/(th2(n)-th2(n-1)); %
                    Aceleración angular de la pala 2 en [Hz^2]
387         end
388         n = n+1;
389     end
390     ag2 = 2*pi*agHz2;
391     agrpm2 = 60*agHz2;
392
393     % Pala 3
394     n = 1;
395     while (n<=nh3)
396         if(n==1)
397             wgHz3(n) = 0; % El valor de la velocidad inicial es 0
398         end
399         if(n>1)
400             wgHz3(n) = 1000/(th3(n)-th3(n-1)); % Velocidad angular
                    de la pala 3 en [Hz]
401         end
402         n = n+1;
403     end
404     % Corrección de error de muestreo
405     n = 1;
406     while(n<=nh3 && correccion==1)
407         if(n>2)
408             if(wgHz3(n-1)<=ncorreccion5*wgHz3(n-2) &&
                    wgHz3(n-1)<=ncorreccion5*wgHz3(n))
409                 wgHz3(n-1) = (wgHz3(n)+wgHz3(n-2))/2;
410             end
```

```
411         end
412         n = n+1;
413     end
414     %
415     wg3 = 2*pi*wgHz3;
416     wgrpm3 = 60*wgHz3;
417     n=1;
418     while (n<=nh3)
419         if(n==1)
420             agHz3(n) = NaN; % El valor de la aceleración inicial no
                es un número
421         end
422         if(n>1)
423             agHz3(n) = (wgHz3(n)-wgHz3(n-1))/(th3(n)-th3(n-1)); %
                Aceleración angular de la pala 3 en [Hz^2]
424         end
425         n = n+1;
426     end
427     ag3 = 2*pi*agHz3;
428     agrpm3 = 60*agHz3;
429
430     % Pala 4
431     n = 1;
432     while (n<=nh4)
433         if(n==1)
434             wgHz4(n) = 0; % El valor de la velocidad inicial es 0
435         end
436         if(n>1)
437             wgHz4(n) = 1000/(th4(n)-th4(n-1)); % Velocidad angular
                de la pala 4 en [Hz]
438         end
439         n = n+1;
440     end
441     % Corrección de error de muestreo
442     n = 1;
443     while(n<=nh4 && correccion==1)
444         if(n>2)
445             if(wgHz4(n-1)<=ncorreccion5*wgHz4(n-2) &&
                wgHz4(n-1)<=ncorreccion5*wgHz4(n))
446                 wgHz4(n-1) = (wgHz4(n)+wgHz4(n-2))/2;
447             end
448         end
449         n = n+1;
450     end
451     %
452     wg4 = 2*pi*wgHz4;
453     wgrpm4 = 60*wgHz4;
454     n=1;
455     while (n<=nh4)
456         if(n==1)
457             agHz4(n) = NaN; % El valor de la aceleración inicial no
                es un número
```

```
458         end
459         if(n>1)
460             agHz4(n) = (wgHz4(n)-wgHz4(n-1))/(th4(n)-th4(n-1)); %
                Aceleración angular de la pala 4 en [Hz^2]
461         end
462         n = n+1;
463     end
464     ag4 = 2*pi*agHz4;
465     agrpm4 = 60*agHz4;
466
467     % Pala 5
468     n = 1;
469     while (n<=nh5)
470         if(n==1)
471             wgHz5(n) = 0; % El valor de la velocidad inicial es 0
472         end
473         if(n>1)
474             wgHz5(n) = 1000/(th5(n)-th5(n-1)); % Velocidad angular
                de la pala 5 en [Hz]
475         end
476         n = n+1;
477     end
478     % Corrección de error de muestreo
479     n = 1;
480     while(n<=nh5 && correccion==1)
481         if(n>2)
482             if(wgHz5(n-1)<=ncorreccion5*wgHz5(n-2) &&
                wgHz5(n-1)<=ncorreccion5*wgHz5(n))
483                 wgHz5(n-1) = (wgHz5(n)+wgHz5(n-2))/2;
484             end
485         end
486         n = n+1;
487     end
488     %
489     wg5 = 2*pi*wgHz5;
490     wgrpm5 = 60*wgHz5;
491     n=1;
492     while (n<=nh5)
493         if(n==1)
494             agHz5(n) = NaN; % El valor de la aceleración inicial no
                es un número
495         end
496         if(n>1)
497             agHz5(n) = (wgHz5(n)-wgHz5(n-1))/(th5(n)-th5(n-1)); %
                Aceleración angular de la pala 5 en [Hz^2]
498         end
499         n = n+1;
500     end
501     ag5 = 2*pi*agHz5;
502     agrpm5 = 60*agHz5;
503 end
504
```

```
505 %% Velocidades y aceleraciones en el vector de tiempo t en [ms]
506
507 % Velocidad angular en [Hz]
508 wat = interp1(ta,wa,t,interpolaciona); % Interpola la velocidad
       angular del anemómetro en [rad/s]
509 wgHzt = interp1(th,wgHz,t,interpolaciong); % Interpola la velocidad
       angular de la turbina en [Hz] (Para poder emplear distintos tipos
       de interpolación diferentes a la lineal es necesario que el
       parámetro "origen=1")
510
511 % Aceleración angular en [Hz^2]
512
513     % Anemómetro
514     n=1;
515     while (n<=T+1)
516         if(n==1)
517             aat(n) = NaN; % El valor de la aceleración inicial no es
                   un número
518         end
519         if(n>1)
520             aat(n) = (wat(n)-wat(n-1))/(t(n)-t(n-1)); % Aceleración
                   angular del anemómetro en [Hz^2]
521         end
522         n = n+1;
523     end
524     %aat = interp1(ta,aa,t); % Aceleración angular del anemómetro en
          [Hz^2]
525
526     % Turbina sin ponderación por palas o con un imán por pala
527     n=1;
528     while (n<=T+1)
529         if(n==1)
530             agHzt(n) = NaN; % El valor de la aceleración inicial no
                   es un número
531         end
532         if(n>1)
533             agHzt(n) = (wgHzt(n)-wgHzt(n-1))/(t(n)-t(n-1)); %
                   Aceleración angular de la turbina en [Hz^2]
534         end
535         n = n+1;
536     end
537     %agHzt = interp1(th,agHz,t); % Aceleración angular de la turbina
          en [Hz^2]
538
539 % Limitación del tiempo mínimo de muestreo
540     n1 = 1;
541     n2 = 1;
542     while(n1<=T+1)
543         if(t(n1)>=Tmin)
544             wauxa(n2) = wat(n1); % Escribe los valores de velocidad
                   para tiempos superiores al tiempo mínimo en un vector
                   auxiliar wauxa
```

```
545             wauxg(n2) = wgHzt(n1); % Escribe los valores de
                   velocidad para tiempos superiores al tiempo mínimo en
                   un vector auxiliar wauxg
546             aauxa(n2) = aat(n1); % Escribe los valores de
                   aceleración para tiempos superiores al tiempo mínimo
                   en un vector auxiliar aauxa
547             aauxg(n2) = agHzt(n1); % Escribe los valores de
                   aceleración para tiempos superiores al tiempo mínimo
                   en un vector auxiliar aauxg
548             n2 = n2+1;
549         end
550         n1 = n1+1;
551     end
552     clearvars wat; % Borra el vector de velocidad antiguo
553     clearvars wgHzt; % Borra el vector de velocidad antiguo
554     clearvars aat; % Borra el vector de aceleración antiguo
555     clearvars agHzt; % Borra el vector de aceleración antiguo
556     wat = wauxa; % Crea un nuevo vector de velocidad límitado al
           tiempo mínimo
557     wgHzt = wauxg; % Crea un nuevo vector de velocidad límitado al
           tiempo mínimo
558     aat = aauxa; % Crea un nuevo vector de aceleración límitado al
           tiempo mínimo
559     agHzt = aauxg; % Crea un nuevo vector de aceleración límitado al
           tiempo mínimo
560     clearvars wauxa; % Borra el vector de velocidad auxiliar wauxa
561     clearvars wauxg; % Borra el vector de velocidad auxiliar wauxg
562     clearvars aauxa; % Borra el vector de aceleración auxiliar aauxa
563     clearvars aauxg; % Borra el vector de aceleración auxiliar aauxg
564
565 % Resto de velocidades y aceleraciones
566
567     % Anemómetro
568     vvmt = ca*Ra*wat; % Velocidad del viento en [m/s]
569     vvkmt = 3.6*ca*Ra*wat; % Velocidad del viento en [km/h]
570     avmt = ca*Ra*aat; % Aceleración del viento en [m/s^2]
571     avkmt = 3.6*ca*Ra*aat; % Aceleración del viento en [km/h^2]
572
573     % Turbina sin ponderación por palas o con un imán por pala
574     wgt = 2*pi*wgHzt; % Velocidad angular de la turbina en [rad/s]
575     wgrpmt = 60*wgHzt; % Velocidad angular de la turbina en [rpm]
576     agt = 2*pi*agHzt; % Aceleración angular de la turbina en
           [rad/s^2]
577     agrpmt = 60*agHzt; % Aceleración angular de la turbina en [rpm^2]
578
579
580 % Hall con ponderación por palas
581 if(modo==2)
582     wgHzt1 = interp1(th1,wgHz1,t); % Velocidad angular de la pala 1
           en [Hz]
583     n=1;
584     while (n<=T+1)
```

```
585         if(n==1)
586             agHzt1(n) = NaN; % El valor de la aceleración inicial no
                    es un número
587         end
588         if(n>1)
589             agHzt1(n) = (wgHzt1(n)-wgHzt1(n-1))/(t(n)-t(n-1)); %
                    Aceleración angular de la pala 1 en [Hz^2]
590         end
591         n = n+1;
592     end
593     %agHzt1 = interp1(th1,agHz1,t); % Aceleración angular de la pala
            1 en [Hz^2]
594     wgt1 = 2*pi*wgHzt1; % Velocidad angular de pala 1 en [rad/s]
595     wgrpmt1 = 60*wgHzt1; % Velocidad angular de la pala 1 en [rpm]
596     agt1 = 2*pi*agHzt1; % Aceleración angular de la pala 1 en
            [rad/s^2]
597     agrpmt1 = 60*agHzt1; % Aceleración angular de la pala 1 en
            [rpm^2]
598
599     wgHzt2 = interp1(th2,wgHz2,t); % Velocidad angular de la pala 2
            en [Hz]
600     n=1;
601     while (n<=T+1)
602         if(n==1)
603             agHzt2(n) = NaN; % El valor de la aceleración inicial no
                    es un número
604         end
605         if(n>1)
606             agHzt2(n) = (wgHzt2(n)-wgHzt2(n-1))/(t(n)-t(n-1)); %
                    Aceleración angular de la pala 2 en [Hz^2]
607         end
608         n = n+1;
609     end
610     %agHzt2 = interp1(th2,agHz2,t); % Aceleración angular de la pala
            2 en [Hz^2]
611     wgt2 = 2*pi*wgHzt2; % Velocidad angular de pala 2 en [rad/s]
612     wgrpmt2 = 60*wgHzt2; % Velocidad angular de la pala 2 en [rpm]
613     agt2 = 2*pi*agHzt2; % Aceleración angular de la pala 2 en
            [rad/s^2]
614     agrpmt2 = 60*agHzt2; % Aceleración angular de la pala 2 en
            [rpm^2]
615
616     wgHzt3 = interp1(th3,wgHz3,t); % Velocidad angular de la pala 3
            en [Hz]
617     n=1;
618     while (n<=T+1)
619         if(n==1)
620             agHzt3(n) = NaN; % El valor de la aceleración inicial no
                    es un número
621         end
622         if(n>1)
```

```
623             agHzt3(n) = (wgHzt3(n)-wgHzt3(n-1))/(t(n)-t(n-1)); % Aceleración angular de la pala 3 en [Hz^2]
624         end
625         n = n+1;
626     end
627     %agHzt3 = interp1(th3,agHz3,t); % Aceleración angular de la pala 3 en [Hz^2]
628     wgt3 = 2*pi*wgHzt3; % Velocidad angular de pala 3 en [rad/s]
629     wgrpmt3 = 60*wgHzt3; % Velocidad angular de la pala 3 en [rpm]
630     agt3 = 2*pi*agHzt3; % Aceleración angular de la pala 3 en [rad/s^2]
631     agrpmt3 = 60*agHzt3; % Aceleración angular de la pala 3 en [rpm^2]
632
633     wgHzt4 = interp1(th4,wgHz4,t); % Velocidad angular de la pala 4 en [Hz]
634     n=1;
635     while (n<=T+1)
636         if(n==1)
637             agHzt4(n) = NaN; % El valor de la aceleración inicial no es un número
638         end
639         if(n>1)
640             agHzt4(n) = (wgHzt4(n)-wgHzt4(n-1))/(t(n)-t(n-1)); % Aceleración angular de la pala 4 en [Hz^2]
641         end
642         n = n+1;
643     end
644     %agHzt4 = interp1(th4,agHz4,t); % Aceleración angular de la pala 4 en [Hz^2]
645     wgt4 = 2*pi*wgHzt4; % Velocidad angular de pala 4 en [rad/s]
646     wgrpmt4 = 60*wgHzt4; % Velocidad angular de la pala 4 en [rpm]
647     agt4 = 2*pi*agHzt4; % Aceleración angular de la pala 4 en [rad/s^2]
648     agrpmt4 = 60*agHzt4; % Aceleración angular de la pala 4 en [rpm^2]
649
650     wgHzt5 = interp1(th5,wgHz5,t); % Velocidad angular de la pala 5 en [Hz]
651     n=1;
652     while (n<=T+1)
653         if(n==1)
654             agHzt5(n) = NaN; % El valor de la aceleración inicial no es un número
655         end
656         if(n>1)
657             agHzt5(n) = (wgHzt5(n)-wgHzt5(n-1))/(t(n)-t(n-1)); % Aceleración angular de la pala 5 en [Hz^2]
658         end
659         n = n+1;
660     end
```

```
661     %agHzt5 = interp1(th5,agHz5,t); % Aceleración angular de la pala
            5 en [Hz^2]
662     wgt5 = 2*pi*wgHzt5; % Velocidad angular de pala 5 en [rad/s]
663     wgrpmt5 = 60*wgHzt5; % Velocidad angular de la pala 5 en [rpm]
664     agt5 = 2*pi*agHzt5; % Aceleración angular de la pala 5 en
            [rad/s^2]
665     agrpmt5 = 60*agHzt5; % Aceleración angular de la pala 5 en
            [rpm^2]
666
667     % Vector de tiempos para las palas
668     tp = t;
669
670     % Compensación de palas
671     wgHztm = (wgHzt1+wgHzt2+wgHzt3+wgHzt4+wgHzt4)/5; % Velocidad
            angular de la turbina en [Hz]
672     n=1;
673     while (n<=numel(t))
674         if(n==1)
675             agHztm(n) = NaN;
676         end
677         if(n>1)
678             agHztm(n) = (wgHztm(n)-wgHztm(n-1))/(t(n)-t(n-1)); %
                    Aceleración angular de la turbina en [Hz^2]
679         end
680         n = n+1;
681     end
682
683     % Limitación del tiempo mínimo de muestreo
684     n1 = 1;
685     n2 = 1;
686     while(n1<=T+1)
687         if(t(n1)>=Tmin)
688             wauxgm(n2) = wgHztm(n1); % Escribe los valores de
                    velocidad para tiempos superiores al tiempo mínimo en
                    un vector auxiliar wauxg
689             aauxgm(n2) = agHztm(n1); % Escribe los valores de
                    aceleración para tiempos superiores al tiempo mínimo
                    en un vector auxiliar aauxg
690             n2 = n2+1;
691         end
692         n1 = n1+1;
693     end
694     clearvars wgHztm; % Borra el vector de velocidad antiguo
695     clearvars agHztm; % Borra el vector de aceleración antiguo
696     wgHztm = wauxgm; % Crea un nuevo vector de velocidad límitado al
            tiempo mínimo
697     agHztm = aauxgm; % Crea un nuevo vector de aceleración límitado
            al tiempo mínimo
698     clearvars wauxgm; % Borra el vector de velocidad auxiliar wauxgm
699     clearvars aauxgm; % Borra el vector de aceleración auxiliar
            aauxgm
700
```

```
701     % Resto de velocidades y aceleraciones
702     wgtm = 2*pi*wgHztm; % Velocidad angular de la turbina en [rad/s]
703     wgrpmtm = 60*wgHztm; % Velocidad angular de la turbina en [rpm]
704     agtm = 2*pi*agHztm; % Aceleración angular de la turbina en
            [rad/s^2]
705     agrpmtm = 60*agHztm; % Aceleración angular de la turbina en
            [rpm^2]
706 end
707
708 % Limitación del tiempo mínimo de muestreo
709 n1 = 1;
710 n2 = 1;
711 while(n1<=T+1)
712     if(t(n1)>=Tmin)
713         taux(n2) = t(n1); % Escribe los valores de tiempo superior
                al tiempo mínimo en un vector auxiliar taux
714         n2 = n2+1;
715     end
716     n1 = n1+1;
717 end
718 clearvars t; % Borra el vector de tiempo antiguo
719 t = taux; % Crea un nuevo vector de tiempo límitado al tiempo mínimo
720 clearvars taux; % Borra el vector de tiempo auxiliar taux
721
722 %% Velocidad media del viento
723
724 nt = numel(t); % Contabiliza el número de elementos del vector de
        tiempo
725 n = 1;
726 nn = isnan(wat);
727 while (n<=nt)
728     if (nn(n)==1)
729         wan(n) = 0; % Crea un vector lógico indicando con un 1 los
                valores no numéricos (NaN) y con un 0 los que sí lo son
730         n = n+1;
731     else
732         wan(n) = wat(n);
733         n = n+1;
734     end
735 end
736 wma = sum(wan)/T; % Velocidad angular media del anemómetro durante
        el ensayo en [rad/s]
737 vmvm = ca*Ra*wma; % Velocidad media del viento durante el ensayo en
        [m/s]
738 vmvkm = 3.6*ca*Ra*wma; % Velocidad media del viento durante el
        ensayo en [km/h]
739
740 %% Potencia en [W]
741
742 % Sin compensación de palas o con un imán por pala
743 Pmg = Jg*(wgt.*agt);
```

```
744 [Pmgmax,nmax] = max(Pmg); % Indica el valor de la potencia máxima y
       su posición en el vector Pmg
745
746 % Con compensación de palas
747 if(modo==2)
748     Pmg1 = Jg*(wgt1.*agt1);
749     Pmg2 = Jg*(wgt2.*agt2);
750     Pmg3 = Jg*(wgt3.*agt3);
751     Pmg4 = Jg*(wgt4.*agt4);
752     Pmg5 = Jg*(wgt5.*agt5);
753
754     Pmgm = Jg*(wgtm.*agtm);
755     [Pmgmmax,nmax] = max(Pmgm); % Indica el valor de la potencia
           máxima y su posición en el vector Pmg
756 end
757
758 %% Gráficas
759
760 if(modo==1)
761     figure (1)
762     plot(t/1000,vvmt)
763     title('Velocidad del viento')
764     xlabel('Tiempo[s]')
765     ylabel('[m/s]')
766
767     figure (2)
768     plot(t/1000,wgrpmt)
769     title('Velocidad de la turbina sin compensación')
770     xlabel('Tiempo[s]')
771     ylabel('[rpm]')
772
773     figure (3)
774     plot(t/1000,avmt)
775     title('Aceleració del viento')
776     xlabel('Tiempo[s]')
777     ylabel('[m/s^2]')
778
779     figure (4)
780     plot(t/1000,agt)
781     title('Aceleración del anemómetro sin compensación')
782     xlabel('Tiempo[s]')
783     ylabel('[rad/s^2]')
784
785     figure (5)
786     plot(t/1000,Pmg)
787     title('Potencia mecánica sin compensación')
788     xlabel('Tiempo[s]')
789     ylabel('[W]')
790
791     figure(6)
792     scatter(wgt,Pmg)
```

```
793     title('Potencia mecánica frente a velocidad de la turbina sin
            compensación')
794     xlabel('Velocidad angular[rad/s]')
795     ylabel('Potencia[W]')
796
797     figure(7)
798     subplot(2,1,1);
799     yyaxis left
800     plot(t/1000,wgrpmt);
801     title('Velocidades')
802     xlabel('Tiempo[s]')
803     ylabel('Velocidad de la turbina con un imán[rpm]')
804     yyaxis right
805     plot(t/1000,vvmt);
806     ylabel('Velocidad del viento[m/s]')
807     subplot(2,1,2);
808     plot(t/1000,agt);
809     title('Aceleraciones')
810     xlabel('Tiempo[s]')
811     ylabel('Aceleración de la turbina con un imán[rad/s^2]')
812 end
813
814 if(modo==2)
815     figure (1)
816     plot(t/1000,vvmt)
817     title('Velocidad del viento')
818     xlabel('Tiempo[s]')
819     ylabel('[m/s]')
820
821     figure (2)
822     plot(t/1000,wgrpmtm)
823     title('Velocidad de la turbina con compensación')
824     xlabel('Tiempo[s]')
825     ylabel('[rpm]')
826
827     figure (3)
828     plot(t/1000,avmt)
829     title('Aceleración del viento')
830     xlabel('Tiempo[s]')
831     ylabel('[m/s^2]')
832
833     figure (4)
834     plot(t/1000,agtm)
835     title('Aceleración de la turbina con compensación')
836     xlabel('Tiempo[s]')
837     ylabel('[rad/s^2]')
838
839     figure (5)
840     plot(t/1000,Pmgm)
841     title('Potencia mecánica con compensación')
842     xlabel('Tiempo[s]')
843     ylabel('[W]')
```

```
844
845     figure (6)
846     scatter(wgt,Pmgm)
847     title('Potencia mecánica frente a velocidad de la turbina con
            compensación')
848     xlabel('Velocidad angular[rad/s]')
849     ylabel('Potencia[W]')
850
851     figure (7)
852     plot(tp/1000,wgrpmt1,tp/1000,wgrpmt2,tp/1000,wgrpmt3,tp/1000,wgrpmt4,tp/1000
853     title('Velocidad de cada pala')
854     xlabel('Tiempo[s]')
855     ylabel('[rpm]')
856     legend('Pala 1','Pala 2','Pala 3','Pala 4','Pala 5')
857
858     figure (8)
859     plot(tp/1000,agt1,tp/1000,agt2,tp/1000,agt3,tp/1000,agt4,tp/1000,agt5)
860     title('Aceleración de cada pala')
861     xlabel('Tiempo[s]')
862     ylabel('[rad/s^2]')
863     legend('Pala 1','Pala 2','Pala 3','Pala 4','Pala 5')
864
865     figure (9)
866     plot(tp/1000,Pmg1,tp/1000,Pmg2,tp/1000,Pmg3,tp/1000,Pmg4,tp/1000,Pmg5)
867     title('Potencia mecánica según cada pala')
868     xlabel('Tiempo[s]')
869     ylabel('[W]')
870     legend('Pala 1','Pala 2','Pala 3','Pala 4','Pala 5')
871
872     figure(10)
873     subplot(2,1,1);
874     yyaxis left
875     plot(t/1000,wgrpmtm);
876     title('Velocidades')
877     xlabel('Tiempo[s]')
878     ylabel('Velocidad de la turbina con un imán[rpm]')
879     yyaxis right
880     plot(t/1000,vvmt);
881     ylabel('Velocidad del viento[m/s]')
882     subplot(2,1,2);
883     plot(t/1000,agtm);
884     title('Aceleraciones')
885     xlabel('Tiempo[s]')
886     ylabel('Aceleración de la turbina con un imán[rad/s^2]')
887 end
888
889 if(modo==3)
890     figure (1)
891     plot(t/1000,vvmt)
892     title('Velocidad del viento')
893     xlabel('Tiempo[s]')
894     ylabel('[m/s]')
```

```
895
896     figure (2)
897     plot(t/1000,wgrpmt)
898     title('Velocidad de la turbina con un imán')
899     xlabel('Tiempo[s]')
900     ylabel('[rpm]')
901
902     figure (3)
903     plot(t/1000,avmt)
904     title('Aceleración del viento')
905     xlabel('Tiempo[s]')
906     ylabel('[m/s^2]')
907
908     figure (4)
909     plot(t/1000,agt)
910     title('Aceleración de la turbina con un imán')
911     xlabel('Tiempo[s]')
912     ylabel('[rad/s^2]')
913
914     figure (5)
915     plot(t/1000,Pmg)
916     title('Potencia mecánica con un imán')
917     xlabel('Tiempo[s]')
918     ylabel('[W]')
919
920     figure(6)
921     scatter(wgt,Pmg)
922     title('Potencia mecánica frente a velocidad de la turbina con un
          imán')
923     xlabel('Velocidad angular[rad/s]')
924     ylabel('Potencia[W]')
925
926     figure(7)
927     subplot(2,1,1);
928     yyaxis left
929     plot(t/1000,wgrpmt);
930     title('Velocidades')
931     xlabel('Tiempo[s]')
932     ylabel('Velocidad de la turbina con un imán[rpm]')
933     yyaxis right
934     plot(t/1000,vvmt);
935     ylabel('Velocidad del viento[m/s]')
936     subplot(2,1,2);
937     plot(t/1000,agt);
938     title('Aceleraciones')
939     xlabel('Tiempo[s]')
940     ylabel('Aceleración de la turbina con un imán[rad/s^2]')
941
942     figure(8)
943     subplot(2,1,1);
944     plot(t/1000,wgrpmt);
945     title('Velocidad de la turbina con un imán')
```

```
946     xlabel('Tiempo[s]')
947     ylabel('[rpm]')
948     subplot(2,1,2);
949     plot(t/1000,agt);
950     title('Aceleración de la turbina con un imán')
951     xlabel('Tiempo[s]')
952     ylabel('[rad/s^2]')
953 end
954
955 %% Almacenamiento de variables
956
957 N = [vmvm;Pmgmax;wgt(nmax)]; %Almacena las variables velocidad media
       del viento, potencia mecánica máxima y velocidad angular de la
       turbina para esa potencia en una matriz
958 M = [M, N]; % La matriz se va ampliando por columnas a medida que se
       realizan más ensayos
959 ne = ne+1; % Aumenta el contador de ensayos realizados para
       distintas velocidades del viento
```

2.5. Dibujo de la curva de seguimiento óptimo de la turbina

```
1  %% Opciones del programa
2
3  r = 1e-3; % Resolución para la velocidad angular
4
5  %% Ajuste de resolución de la velocidad
6
7  n = 1;
8  while(n<ne)
9      Pm(n) = M(2,n);
10     w(n) = M(3,n);
11     n = n+1;
12 end
13
14 wmax = max(w); % Velocidad angular máxima en [rad/s]
15
16 wi = 0:r:wmax; % Genera un vector de velocidades de la resolución
       "r" dada
17 Pmi = interp1(w,Pm,wi,'pchip'); % Interpola valores de potencia para
       cada velocidad
18
19 %% Gráfica
20
21 figure(11)
22 plot(wi,Pmi)
23 title('Curva de seguimiento óptimo de la turbina')
24 xlabel('Velocidad angular[rad/s]')
25 ylabel('Potencia[W]')
```

2.6. Estimación de las pérdidas mecánicas de la turbina

```
%% Opciones del programa

r = 3;  % Número de coeficientes a calcular

%% Selección de los puntos de muestreo

R = numel(t);  % Contabilización del número de puntos de tiempo del
   ensayo

%Obtención de los puntos de muestreo (situados a la misma distancia
   entre ellos)
n = 1;
while(n<=r)
    if(n==1)
        T(n) = 1;
    end
    if(n>1)
        T(n) = round(n*R/r);  % Aproximación al número entero más
           cercano del punto de tiempo seleccionado
    end
    if(n==r)
        T(n) = R;
    end
    n = n+1;
end

%% Resolución del sistema de ecuaciones lineales

% Obtención de la matriz A
c = 1;
while(c<=r)
   f = 1;
   while(f<=r)
       A(f,c) = wgt(T(f))^(c-1);
       f = f+1;
   end
   c = c+1;
end

% Obtención del vector columna b
f = 1;
while(f<=r)
   b(f,1) = -Jg*agt(T(f));
   f = f+1;
end

% Resolución del sistema de ecuaciones lineales
x = A\b;
```

2.7. Muestreo de variables en tiempo real (desarrollado por Javier Colinas Cano excepto en las partes en las que se indique lo contrario)

```
clear all

tic % Función de matlab permite contar el tiempo que pasa desde que se inicia la función. Cada vez que se llame a la función toc se guardará el tiempo que ha pasado.
bt = Bluetooth('HC-05',1); %Se guarda la dirección de bluetooth en la variable bt para poder establecer conexión con el módulo bluetooth
fopen(bt); % Se abre el canal de conexión que permite la transferencia de datos de arduino al matlab. Además, cambiará la velocidad a de parpadeo del módulo cuando se conecten este y el matlab tras ejecutar esta línea de código.
pause(2); % Espera 2 segundos a que se establezca conexión para no comenzar con el código

%% Variables auxiliares

cont1 = 1; % Un contador para ta
cont2 = 1; % Un contador para th

%% Variables auxiliares que reciben los datos del bluetooth

a = 0;
b = 0;
c = 0;

%% Vectores en los que se va a guardar los tiempos que se reciben a través de bluetooth

ta = []; % Tiempo del anémometro
th = []; % Tiempo del sensor de efecto Hall

%% Vectores de tiempo para ordenar los datos según los recibe matlab

tiempo1 = [];
tiempo2 = [];

%% Vector que guarda las 2 gráficas

%s = [];

%% Variables auxiliares para diferenciar a que pala pertenece el dato.

m = 1; % Contador para ir guardando el dato en las diferentes palas.

% Contadores para cada pala
```

```
38 n1 = 1;
39 n2 = 1;
40 n3 = 1;
41 n4 = 1;
42 n5 = 1;
43
44 %% Datos
45
46 Jg = 0.147761; %2.2342;
47 Ra = 70e-3;
48 ca = 21*pi/200;
49 wa(1)=0;
50 aa(1)=0;
51 coma=2;
52
53 %% Bucle infinito hasta que se superen los 4 millones de datos
       recibidos de ta
54
55 while (cont1<4000000)
56     %% Recepción de los datos
57
58     c = fscanf(bt,'%c',1); % Se recibe solo 1 dato y se lee en
          lenguaje C, es decir, según el código ASCII. Sólo recibirá un
          65 o un 72 que en lenguaje C se traduce a una A o a una H
          respectivamente. Sirve para diferenciar a cual pertenece el
          dato que se va a recibir.
59     a = fread(bt,3); % Esta función pide que le envie 3 datos, que
          corresponden a 24 bits. Esto se debe a que el módulo
          bluetooth solo puede enviar datos que ocupen un máximo de 8
          bits. Por ello, se ha preparado el código para enviar los
          datos divididos en 3.
60     b = a(1)*256*256+a(2)*256+a(3); % Esta línea vuelve a juntar los
          3 datos recibidos para dar lugar al número que se tenía que
          enviar
61
62     %% Organización de los datos recibidos
63
64     if (c == 'A') % Identifica si el dato leído pertenece a ta o a th
65         ta(cont1) = b; % Guarda el dato recibido en el vector ta en
              la posición cont1
66         tiempo1(cont1) = toc; % Guarda el tiempo, contabilizado
              según matlab
67
68         %% Parte desarrollada por Antonio Serda Mena
69         % Esta parte traduce los tiempos en velocidades lineales,
              velocidades de rotación, aceleraciones lineales y de
              rotación
70
71 %         if cont1>1 % Esto se debe a que no se puede calcular una
       velocidad si no se tiene una diferencia de dos tiempos
72 %
73 %             wa(cont1) = (2*pi*1000/coma)/(ta(cont1)-ta(cont1-1));
```

```
74  %             vvm(cont1) = ca*Ra*wa(cont1);
75  %              %vvkm(cont1) = 3.6*ca*Ra*wa(cont1);
76  %              %aa(cont1) =
    (wa(cont1)-wa(cont1-1))/(ta(cont1)-ta(cont1-1));
77  %              %avm(cont1) = ca*Ra*aa(cont1);
78  %              %avkm(cont1) = 3.6*ca*Ra*aa(cont1);
79  %
80  %       % Fin de la parte desarrollada por Antonio Serda Mena
81  %
82  %              % A continuación se dibuja en una gráfica a la
    velocidad a la que gira el anemómetro en rev/s
83  %              s(1) = subplot(2,1,1); % Sirve para crear una matriz
    de figuras dividida en este caso en una matriz de 2 filas y una
    columna
84  %              plot(tiempo1, vvm, '-'); % Sirve para dibujar en la
    figura últimamente mencionada, es decir, que depende de la última
    figura que se haya abierto
85  %              hold on % Sirve para mantener la figura a la vista,
    mientras corre el código
86  %              grid % Esta función sirve para dividir la figura en
    celdas
87  %              pause(0.001); % Esta función permite que la gráfica se
    vaya modificando en tiempo real
88  %
89  %         end
90
91          cont1 = cont1+1; % Se añade 1 al contador para guardar el
               próximo que se reciba en la siguiente posición del vector
               ta
92
93      else
94          if (c == 'H') % Identifica si el dato recibido es del sensor
               de efecto Hall
95                th(cont2) = b; % Guarda el dato en el vector th en
                     la posición cont2
96                tiempo2(cont2) = toc; % Guarda el tiempo,
                     contabilizado según matlab
97
98                %% Parte desarrollada por Antonio Serda Mena
99                % Esta parte traduce los tiempos en velocidades
                     lineales, velocidades de rotación, aceleraciones
                     lineales y de rotación. Además ordena después las
                     velocidades recibidas según la pala a la que ha
                     dado ese dato.
100
101               if (cont2>1) % Esto se debe a que no se puede
                     calcular una velocidad si no se tiene una
                     diferencia de dos tiempos
102                    % Se calculan las velocidades y aceleraciones
                          de rotación y el par mecánico que se genera
103                  wgHz(cont2) = (1000)/(th(cont2)-th(cont2-1));
104                  wg(cont2) = 2*pi*wgHz(cont2);
```

```
105 %                     wgrpm(cont2) = 60*wgHz(cont2);
106 %                     agHz(cont2) =
    (wgHz(cont2)-wgHz(cont2-1))/(th(cont2)-th(cont2-1));
107 %                     ag(cont2) = 2*pi*agHz(cont2);
108 %                     agrpm(cont2) = 60*agHz(cont2);
109 %                     Pmg(cont2) = Jg*(wg(cont2)*ag(cont2));
110
111                   % Fin de la parte desarrollada por Antonio Serda Mena
112
113                       % Se dibuja la velocidad de rotación en rev/s
                             en función de los tiempos del sensor de
                             efecto Hall
114                       plot(th, wg, '-');
115                       hold on % Se mantiene la figura mientras se
                             ejecuta el código
116                       grid % Pone celdas a la figura
117                       pause(0.001); % Permite la graficación de datos
                             en directo
118
119                       %% Organización de las velocidades según su pala
120                       % Parte desarrollada por Antonio Serda Mena
121
122                       % Pala 1
123 %                     if(m==1)
124 %                         th1(n1) = th(cont2);
125 %                         if n1>1
126 %                             wgHz1(n1) = 1000/(th1(n1)-th1(n1-1));
127 %                             wg1(n1) = 2*pi*wgHz1(n1);
128 %                              %wgrpm1(n1) = 60*wgHz1(n1);
129 %                              %agHz1(n1) =
    (wgHz1(n1)-wgHz1(n1-1))/(th1(n1)-th1(n1-1));
130 %                              %ag1(n1) = 2*pi*agHz1(n1);
131 %                              %agrpm1(n1) = 60*agHz1(n1);
132 %                              %Pmg1(n1) = Jg*(wg1(n1)*ag1(n1));
133 %                              %s(2) = subplot(2,1,2);
134 %                             plot(th1, wg1, '-');
135 %                             hold on
136 %                             grid
137 %                             pause(0.001);
138 %                         end
139 %                         n1 = n1+1; % Contador de la pala 1
140 %                     end
141
142 % %                     % Pala 2
143 % %                     if(m==2)
144 % %                         th2(n2) = th(cont2);
145 % %                         wgHz2(n2) = 1000/(th2(n2)-th2(n2-1));
146 % %                         wg2(n2) = 2*pi*wgHz2(n2);
147 % %                         wgrpm2(n2) = 60*wgHz2(n2);
148 % %                         agHz2(n2) =
    (wgHz2(n2)-wgHz2(n2-1))/(th2(n2)-th2(n2-1));
149 % %                         ag2(n2) = 2*pi*agHz2(n2);
```

```
150 % %                         agrpm2(n2) = 60*agHz2(n2);
151 % %                         Pmg2(n2) = Jg*(wg2(n2)*ag2(n2));
152 % %                         n2 = n2+1;  % Contador de la pala 2
153 % %                     end
154 
155 % %                     % Pala 3
156 % %                     if(m==3)
157 % %                         th3(n3) = th(cont2);
158 % %                         wgHz3(n3) = 1000/(th3(n3)-th3(n3-1));
159 % %                         wg3(n3) = 2*pi*wgHz3(n3);
160 % %                         wgrpm3(n3) = 60*wgHz3(n3);
161 % %                         agHz3(n3) =
    (wgHz3(n3)-wgHz3(n3-1))/(th3(n3)-th3(n3-1));
162 % %                         ag3(n3) = 2*pi*agHz3(n3);
163 % %                         agrpm3(n3) = 60*agHz3(n3);
164 % %                         Pmg3(n3) = Jg*(wg3(n3)*ag3(n3));
165 % %                         n3 = n3+1;  % Contador de la pala 3
166 % %                     end
167 
168 % %                     % Pala 4
169 % %                     if(m==4)
170 % %                         th4(n4) = th(cont2);
171 % %                         wgHz4(n4) = 1000/(th4(n4)-th4(n4-1));
172 % %                         wg4(n4) = 2*pi*wgHz4(n4);
173 % %                         wgrpm4(n4) = 60*wgHz4(n4);
174 % %                         agHz4(n4) =
    (wgHz4(n4)-wgHz4(n4-1))/(th4(n4)-th4(n4-1));
175 % %                         ag4(n4) = 2*pi*agHz4(n4);
176 % %                         agrpm4(n4) = 60*agHz4(n4);
177 % %                         Pmg4(n4) = Jg*(wg4(n4)*ag4(n4));
178 % %                         n4 = n4+1;  % Contador de la pala 4
179 % %                     end
180 
181 % %                     % Pala 5
182 % %                     if(m==5)
183 % %                         th5(n5) = th(cont2);
184 % %                         wgHz5(n5) = 1000/(th5(n5)-th5(n5-1));
185 % %                         wg5(n5) = 2*pi*wgHz5(n5);
186 % %                         wgrpm5(n5) = 60*wgHz5(n5);
187 % %                         agHz5(n5) =
    (wgHz5(n5)-wgHz5(n5-1))/(th5(n5)-th5(n5-1));
188 % %                         ag5(n5) = 2*pi*agHz5(n5);
189 % %                         agrpm5(n5) = 60*agHz5(n5);
190 % %                         Pmg5(n5) = Jg*(wg5(n5)*ag5(n5));
191 % %                         n5 = n5+1;  % Contador de la pala 5
192 %                          m = 0;
193 %                      end
194 
195 %                  m = m+1;  % Contador para ir guardando los datos
    en diferentes palas
196 
```

```
197                     % Fin de la parte desarrollada por Antonio
                           Serda Mena
198
199                 end
200
201
202                 cont2 = cont2+1; % Se añade 1 al contador para
                       guardar el próximo que se reciba en la siguiente
                       posición del vector th
203             end
204         end
205         a = 0; % Se iguala a 0 para asegurarse de que no se repite un
                dato
206 end
207
208 %% Dar nombre a los datos de las figuras
209 % Estas líneas de código solo se ejecutan si se llega a salir del
       bucle infinito, es decir, cuando se llegen a los 4 millones de
       datos del anemómetro (esto se puede modificar)
210
211 %xlabel(s(1),'tiempo');
212 xlabel('tiempo');
213 %ylabel(s(1),'metros por segundo del anemometro');
214 ylabel('radianes por segundo del sensor hall');
215 %title(s(1),'grafica de registro de velocidades anemometro');
216 title('grafica de registro de velocidades sensor hall');
217
218 fclose(bt); % Esta última línea de código desconecta la conexión
       entre bluetooth y matlab. Visualmente cambia el parpadeo del
       módulo bluetooth a mayor velocidad. Es necesario para que no
       surjan problemas. También se puede escribir este comando en el
       command window si se para el código antes de finalizar. Además se
       puede desconectar y volver a conectar (para que cambie el
       parpadeo) y borrar todos los datos del matlab con la línea de
       código clear all.
```

PARTE IV

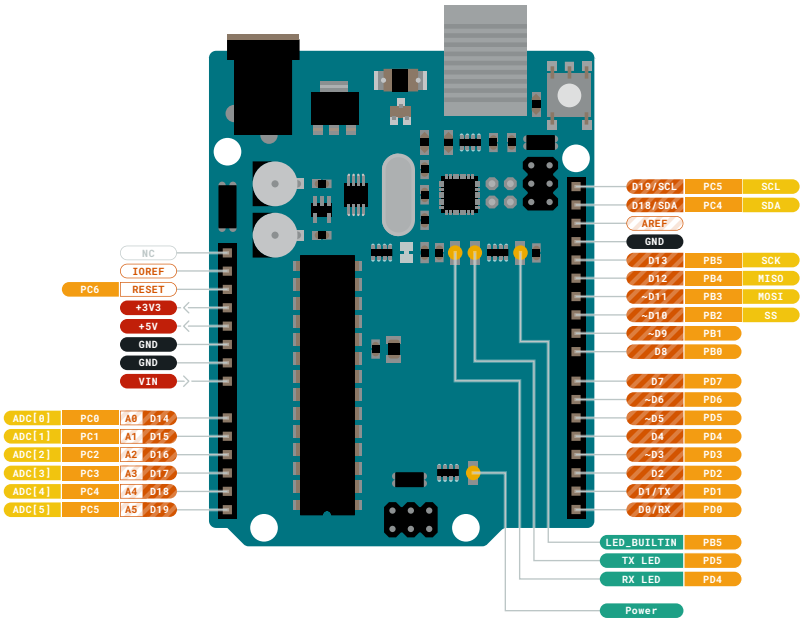
HOJAS DE CARACTERÍSTICAS

Arduino Uno Rev3 (Elegido por Juan Romeo Granados)

ARDUINO

ARDUINO UNO REV3

STORE.ARDUINO.CC/UNO-REV3



- Ground
- Power
- LED
- Internal Pin
- SWD Pin
- Digital Pin
- Analog Pin
- Other Pin
- Microcontroller's Port
- Default

MAXIMUM current per I/O pin is 20mA

MAXIMUM current per +3.3V pin is 50mA

VIN 6-20 V input to the board.

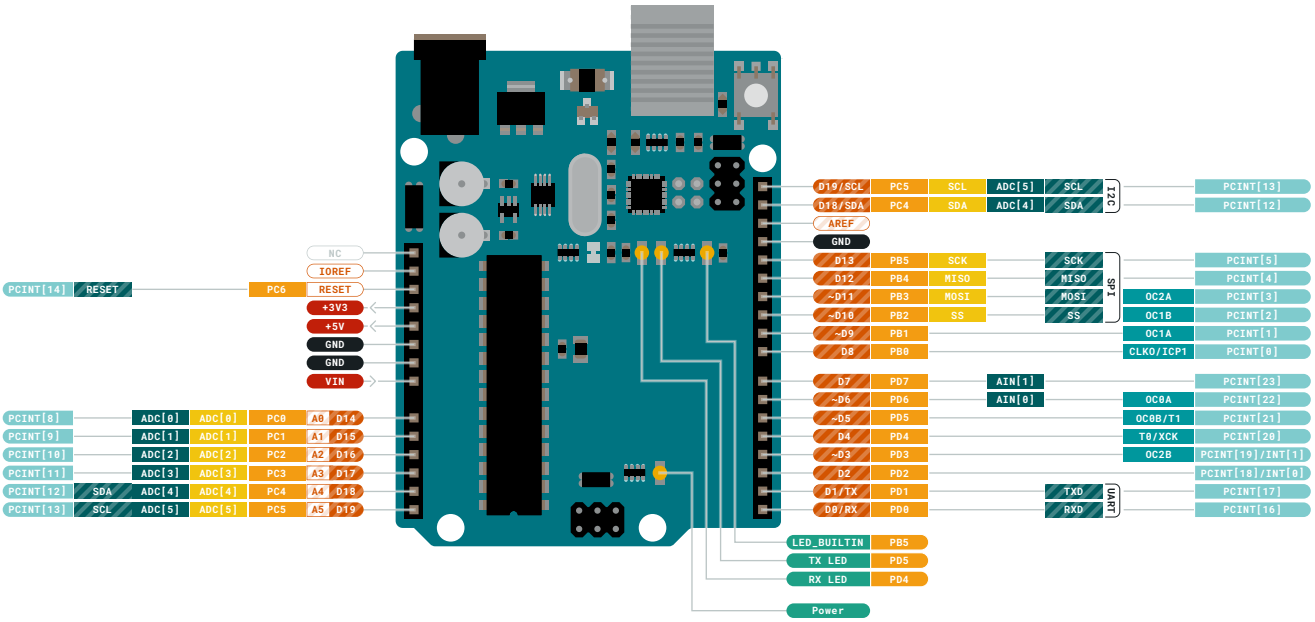
ARDUINO.CC

Last update: 23/03/2020

ARDUINO

ARDUINO
UNO REV3

STORE.ARDUINO.CC/UNO-REV3



Ground
Power
LED
Internal Pin
SWD Pin

Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default

Analog
Communication
Timer
Interrupt
Sercom

MAXIMUM current per I/O pin is 20mA

MAXIMUM current per +3.3V pin is 50mA

VIN 6-20 V input to the board.

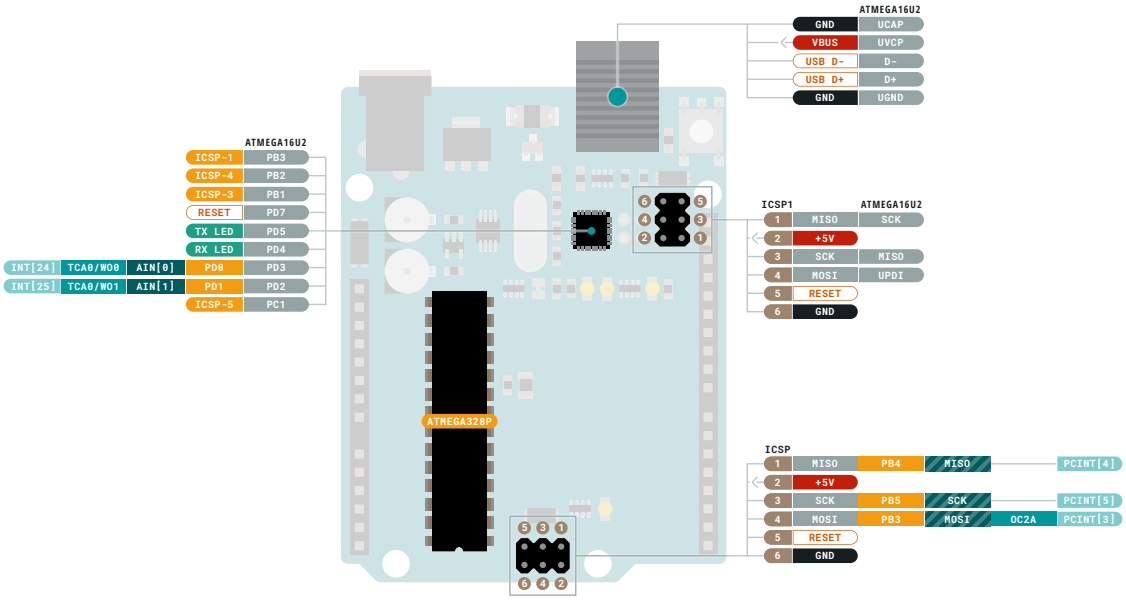
ARDUINO.CC

Last update: 23/03/2020

ARDUINO

ARDUINO UNO REV3

STORE.ARDUINO.CC/UNO-REV3



Ground
Power
LED
Internal Pin
SWD Pin

Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default

Analog
Communication
Timer
Interrupt
Sercom

MAXIMUM current per I/O pin is 20mA

MAXIMUM current per +3.3V pin is 50mA

VIN 6-20 V input to the board.

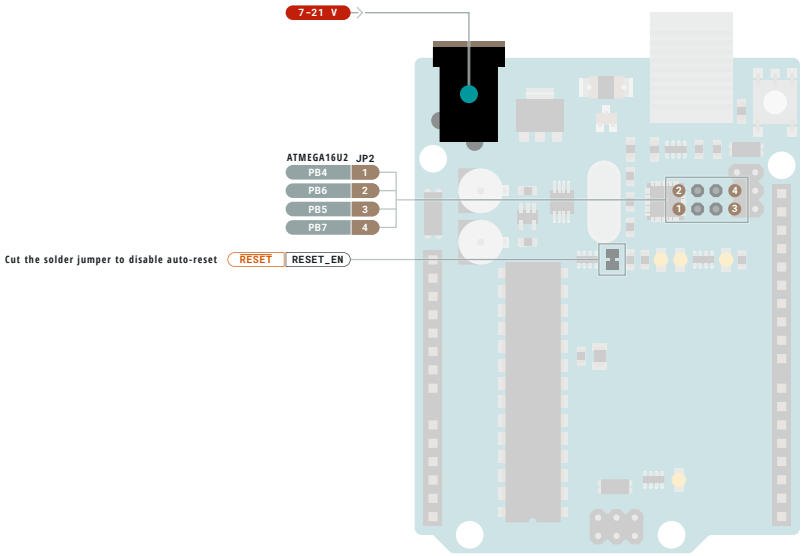
ARDUINO.CC

Last update: 23/03/2020

ARDUINO

ARDUINO
UNO REV3

STORE.ARDUINO.CC/UNO-REV3



- Ground
- Power
- LED
- Internal Pin
- SWD Pin
- Digital Pin
- Analog Pin
- Other Pin
- Microcontroller's Port
- Default

SJ Pin
Making a short circuit using the solder jumper allows only the function in the SJ Pin cells.

MAXIMUM current per I/O pin is 20mA

MAXIMUM current per +3.3V pin is 50mA

VIN 6-20 V input to the board.

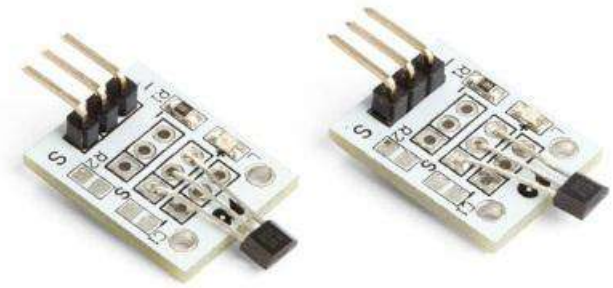
ARDUINO.CC

Last update: 23/03/2020

SENSOR MAGNÉTICO DE EFECTO HALL (HOLZER) - VMA313 (Elegido por Juan Romeo Granados)

VMA313

ARDUINO® COMPATIBLE HALL (HOLZER) MAGNETIC SWITCH MODULE (2 PCS)



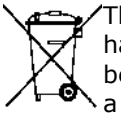
USER MANUAL

USER MANUAL

1. Introduction

To all residents of the European Union

Important environmental information about this product



This symbol on the device or the package indicates that disposal of the device after its lifecycle could harm the environment. Do not dispose of the unit (or batteries) as unsorted municipal waste; it should be taken to a specialized company for recycling. This device should be returned to your distributor or to a local recycling service. Respect the local environmental rules.

If in doubt, contact your local waste disposal authorities.

Thank you for choosing Velleman®! Please read the manual thoroughly before bringing this device into service. If the device was damaged in transit, do not install or use it and contact your dealer.

2. Safety Instructions

- This device can be used by children aged from 8 years and above, and persons with reduced physical, sensory or mental capabilities or lack of experience and knowledge if they have been given supervision or instruction concerning the use of the device in a safe way and understand the hazards involved. Children shall not play with the device. Cleaning and user maintenance shall not be made by children without supervision.

- Indoor use only.
  Keep away from rain, moisture, splashing and dripping liquids.

3. General Guidelines

- Refer to the Velleman® Service and Quality Warranty on the last pages of this manual.
- Familiarise yourself with the functions of the device before actually using it.
- All modifications of the device are forbidden for safety reasons. Damage caused by user modifications to the device is not covered by the warranty.
- Only use the device for its intended purpose. Using the device in an unauthorised way will void the warranty.
- Damage caused by disregard of certain guidelines in this manual is not covered by the warranty and the dealer will not accept responsibility for any ensuing defects or problems.
- Nor Velleman nv nor its dealers can be held responsible for any damage (extraordinary, incidental or indirect) – of any nature (financial, physical…) arising from the possession, use or failure of this product.
- Due to constant product improvements, the actual product appearance might differ from the shown images.
- Product images are for illustrative purposes only.
- Do not switch the device on immediately after it has been exposed to changes in temperature. Protect the device against damage by leaving it switched off until it has reached room temperature.
- Keep this manual for future reference.

4. What is Arduino®

Arduino® is an open-source prototyping platform based in easy-to-use hardware and software. Arduino® boards are able to read inputs – light-on sensor, a finger on a button or a Twitter message – and turn it into an output – activating of a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so, you use the Arduino programming language (based on Wiring) and the Arduino® software IDE (based on Processing).

Surf to www.arduino.cc and www.arduino.org for more information.

5. Overview

The device includes an on-chip Hall voltage generator for magnetic sensing, an amplifier that amplifies the Hall voltage, a Schmitt trigger to provide switching hysteresis for noise rejection, and an open-collector output.

Arduino®		VMA313
A10	▶	S
+5 V	▶	+
GND	▶	-

voltage ..... 5 VDC
connection ..... 3 pins, + (middle pin), - (ground) and S (signal)
output (S) ..... Schmitt Trigger, Active Low
activation ..... 30 gauss
deactivation ..... 10 gauss
LED indicator ..... on when activated
dimensions ..... 25 x 15 mm
weight ..... 2 g

Use this device with original accessories only. Velleman nv cannot be held responsible in the event of damage or injury resulting from (incorrect) use of this device. For more info concerning this product and the latest version of this manual, please visit our website www.velleman.eu. The information in this manual is subject to change without prior notice.

Velleman® Service and Quality Warranty

Since its foundation in 1972, Velleman® acquired extensive experience in the electronics world and currently distributes its products in over 85 countries.

All our products fulfil strict quality requirements and legal stipulations in the EU. In order to ensure the quality, our products regularly go through an extra quality check, both by an internal quality department and by specialized external organisations. If, all precautionary measures notwithstanding, problems should occur, please make appeal to our warranty (see guarantee conditions).

General Warranty Conditions Concerning Consumer Products (for EU):

• All consumer products are subject to a 24-month warranty on production flaws and defective material as from the original date of purchase.

• Velleman® can decide to replace an article with an equivalent article, or to refund the retail value totally or partially when the complaint is valid and a free repair or replacement of the article is impossible, or if the expenses are out of proportion.

You will be delivered a replacing article or a refund at the value of 100% of the purchase price in case of a flaw occurred in the first year after the date of purchase and delivery, or a replacing article at 50% of the purchase price or a refund at the value of 50% of the retail value in case of a flaw occurred in the second year after the date of purchase and delivery.

• Not covered by warranty:

- all direct or indirect damage caused after delivery to the article (e.g. by oxidation, shocks, falls, dust, dirt, humidity...), and by the article, as well as its contents (e.g. data loss), compensation for loss of profits;

- consumable goods, parts or accessories that are subject to an aging process during normal use, such as batteries (rechargeable, non-rechargeable, built-in or replaceable), lamps, rubber parts, drive belts... (unlimited list);

- flaws resulting from fire, water damage, lightning, accident, natural disaster, etc.…;

- flaws caused deliberately, negligently or resulting from improper handling, negligent maintenance, abusive use or use contrary to the manufacturer's instructions;

- damage caused by a commercial, professional or collective use of the article (the warranty validity will be reduced to six (6) months when the article is used professionally);

- damage resulting from an inappropriate packing and shipping of the article;

- all damage caused by modification, repair or alteration performed by a third party without written permission by Velleman®.

• Articles to be repaired must be delivered to your Velleman® dealer, solidly packed (preferably in the original packaging), and be completed with the original receipt of purchase and a clear flaw description.

• Hint: In order to save on cost and time, please reread the manual and check if the flaw is caused by obvious causes prior to presenting the article for repair. Note that returning a non-defective article can also involve handling costs.

• Repairs occurring after warranty expiration are subject to shipping costs.

• The above conditions are without prejudice to all commercial warranties.

The above enumeration is subject to modification according to the article (see article's manual).

Made in PRC
Imported by Velleman nv
Legen Heirweg 33, 9890 Gavere, Belgium
www.velleman.eu

Anemómetro Sensor de Velocidad Viento con RJ11 para Estación Meteorológica N25FR WH1080 (Elegido por Juan Romeo Granados)

Weather Sensor Assembly p/n 80422

Imported by Argent Data Systems

Usage Notes

This kit includes a wind vane, cup anemometer, and tipping bucket rain gauge, with associated mounting hardware. These sensors contain no active electronics, instead using sealed magnetic reed switches and magnets to take measurements. A voltage must be supplied to each instrument to produce an output.

Assembly

The wind sensor arm mounts on top of the two-piece metal mast and supports the wind vane and anemometer. A short cable connects the two wind sensors. Plastic clips on the underside of the arm hold this cable in place. Screws are provided to secure the sensors to the arm.

The rain gauge may be mounted lower on the mast using its own mounting arm and screw, or it may be mounted independently.

Rain Gauge

The rain gauge is a self-emptying tipping bucket type. Each 0.011" (0.2794 mm) of rain causes one momentary contact closure that can be recorded with a digital counter or microcontroller interrupt input. The gauge's switch is connected to the two center conductors of the attached RJ11-terminated cable.

Anemometer

The cup-type anemometer measures wind speed by closing a contact as a magnet moves past a switch. A wind speed of 1.492 MPH (2.4 km/h) causes the switch to close once per second.

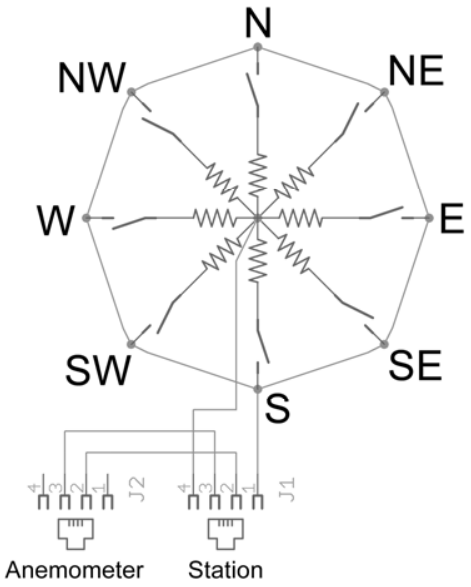
The anemometer switch is connected to the inner two conductors of the RJ11 cable shared by the anemometer and wind vane (pins 2 and 3.)

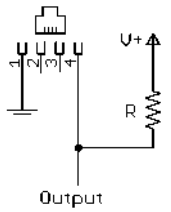
Wind Vane

The wind vane is the most complicated of the three sensors. It has eight switches, each connected to a different resistor. The vane's magnet may close two switches at once, allowing up to 16 different positions to be indicated. An external resistor can be used to form a voltage divider, producing a voltage output that can be measured with an analog to digital converter, as shown below.

The switch and resistor arrangement is shown in the diagram to the right. Resistance values for all 16 possible positions are given in the table.

Resistance values for positions between those shown in the diagram are the result of two adjacent resistors connected in parallel when the vane's magnet activates two switches simultaneously.

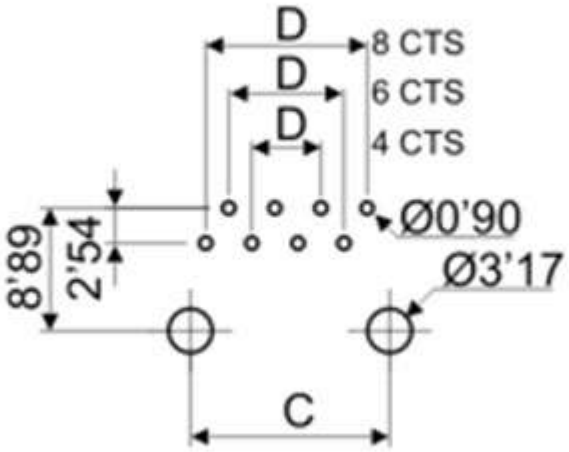


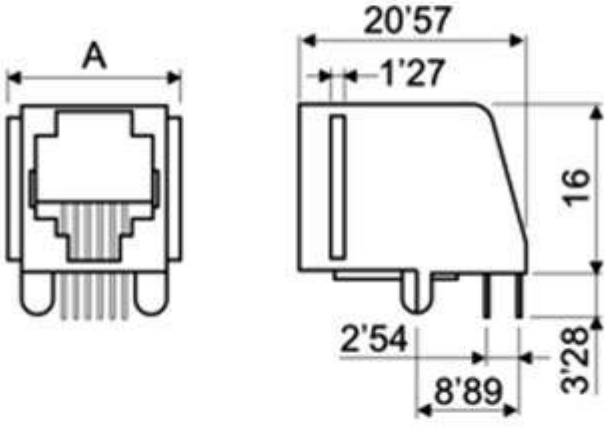


Example wind vane interface circuit. Voltage readings for a 5 volt supply and a resistor value of 10k ohms are given in the table.

Direction (Degrees)	Resistance (Ohms)	Voltage (V=5v, R=10k)
0	33k	3.84v
22.5	6.57k	1.98v
45	8.2k	2.25v
67.5	891	0.41v
90	1k	0.45v
112.5	688	0.32v
135	2.2k	0.90v
157.5	1.41k	0.62v
180	3.9k	1.40v
202.5	3.14k	1.19v
225	16k	3.08v
247.5	14.12k	2.93v
270	120k	4.62v
292.5	42.12k	4.04v
315	64.9k	4.33v
337.5	21.88k	3.43v

Base Hembra Circuito Impreso 6P4C - RJ11 - Horizontal - 39.700/6/4


D
8 CTS
D
6 CTS
D
4 CTS
Ø0'90
Ø3'17
8'89
2'54
C


A
20'57
1'27
16
2'54
3'28
8'89

Servomotor Alta Potencia - HD-1501MG

1. 使 用 環 境 條 件
Apply Environmental Condition：

No.	项目 item	规格 standard
1-1	保存溫度 Storage Temperature Range	-20℃～60℃
1-2	操作溫度 Operating Temperature Range	-10℃～50℃
1-3	操作電壓 Operating Voltage Range	4.8V~6.0V

2. 測 試 環 境
Standard Test Environment：

2-1	測試環境 Standard Test Environment	每一个检查必须是正常的温度和湿度进行测量，温度 25±5℃，相对湿度 65±10%,在按照本规范的标准测试条件下判断特征。 Every characteristic of the inspect must be normal temperature and humidity carry out the test , temperature 25±5℃ and relative humidity 65±10% of judgment made in accordance with this specification standard testing conditions.

3. 外 觀 檢 查
Appearance Inspection：

No.	项目 item	规格 standard
3-1	外觀尺寸 Outline Drawing	尺寸见附件 Dimension see the attachment
3-2	外觀 Appearance	无损坏，不允许影响功能 No damage which affects functions allowed

4. 電 氣 特 性
Electrical Specification (Function of the Performance)：

No.	项目 item	4.8V	6.0V
4-1	空載轉速 Operating speed (at no load)	0.16 sec/60°	0.14 sec/60°
4-2	空載電流 Running current (at no load)	400 mA	500 mA
4-3	停止扭力 Stall torque (at locked)	15.5 kg-cm	17 kg-cm
4-4	停止電流 Stall current (at locked)	2300 mA	2500 mA
4-5	待機電流 Idle current (at stopped)	4 mA	5 mA

注：项目 4-2 定义平均值时，伺服器无负荷运行

Note: Item 4-2 definition is average value when the servo running with no load

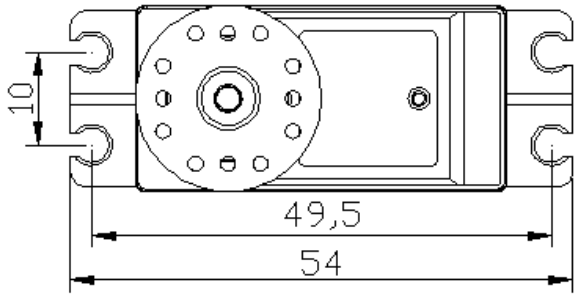
5. 機 械 特 性
Mechanical Specification：

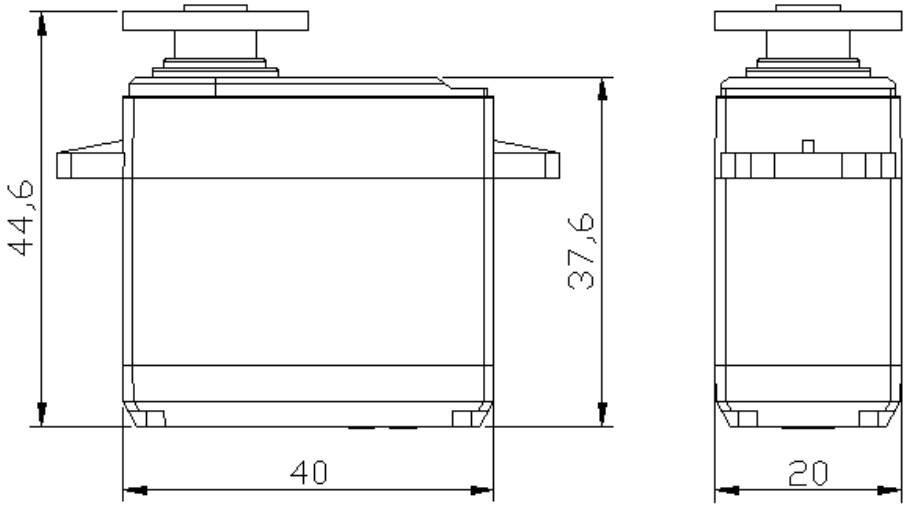
No.	项目 item	规格 standard
5-1	外觀尺寸 Overall Dimensions	见附件 See the drawing
5-2	機構極限角度 Limit angle	180°± 10°
5-3	重量 Weight	63 ± 1g
5-4	導線規格 Connector wire gauge	#28 PVC
5-5	導線長度 Connector wire length	300± 5 mm
5-6	舵片規格 Horn gear spline	25T/ ϕ 5.80
5-7	舵片種類 Horn type	条型.半臂舵板 Single, Double
5-8	減速比 Reduction ratio	1/298

6. 控 制 特 性

Control Specification：

No.	项目	规格
6-1	控制系統 Control system	改变脉冲宽度 Pulse Width Modification
6-2	放大器種類 Amplifier type	模拟控制器 Analog Controller
6-3	操作角度 Operating travel	90°（在 1000→2000 μsec）
6-4	中立位置 Neutral position	1500 μsec
6-5	脈波訊號虛位 Dead band width	2 μsec
6-6	旋轉方向 Rotating direction	顺时针（在 1500→2000 μsec） Counterclockwise (when1500→2000 μsec)
6-7	脈波寬度範圍 Pulse width range	800→2200 μsec
6-8	可作動角度範圍 Maximum travel	大约 165°(在 800→2200 μsec) Approx 165°(when800→2200 μsec)

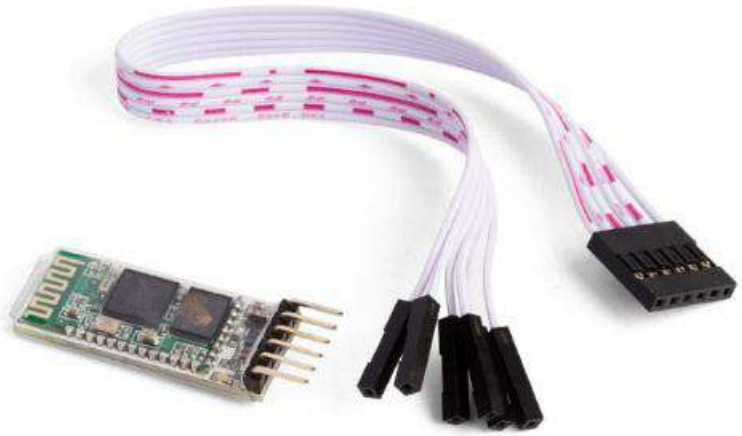




MÓDULO DE TRANMISIÓN HC-05 - VMA302 (Elegido por Javier Colinas Cano)

VMA302

BLUETOOTH® HC-05 TRANSMISSION MODULE



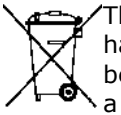
USER MANUAL

USER MANUAL

1. Introduction

To all residents of the European Union

Important environmental information about this product



This symbol on the device or the package indicates that disposal of the device after its lifecycle could harm the environment. Do not dispose of the unit (or batteries) as unsorted municipal waste; it should be taken to a specialized company for recycling. This device should be returned to your distributor or to a local recycling service. Respect the local environmental rules.

If in doubt, contact your local waste disposal authorities.

Thank you for choosing Velleman®! Please read the manual thoroughly before bringing this device into service. If the device was damaged in transit, do not install or use it and contact your dealer.

2. Safety Instructions

- This device can be used by children aged from 8 years and above, and persons with reduced physical, sensory or mental capabilities or lack of experience and knowledge if they have been given supervision or instruction concerning the use of the device in a safe way and understand the hazards involved. Children shall not play with the device. Cleaning and user maintenance shall not be made by children without supervision.

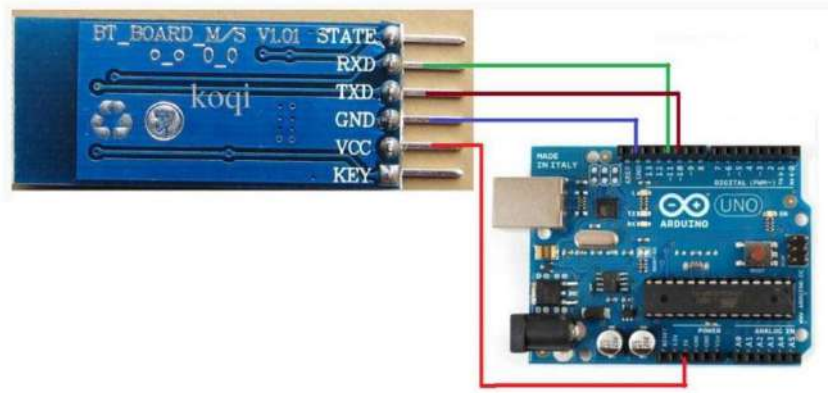
- Indoor use only.
  Keep away from rain, moisture, splashing and dripping liquids.

3. General Guidelines

- Refer to the Velleman® Service and Quality Warranty on the last pages of this manual.
- Familiarise yourself with the functions of the device before actually using it.
- All modifications of the device are forbidden for safety reasons. Damage caused by user modifications to the device is not covered by the warranty.
- Only use the device for its intended purpose. Using the device in an unauthorised way will void the warranty.
- Damage caused by disregard of certain guidelines in this manual is not covered by the warranty and the dealer will not accept responsibility for any ensuing defects or problems.
- Nor Velleman nv nor its dealers can be held responsible for any damage (extraordinary, incidental or indirect) – of any nature (financial, physical…) arising from the possession, use or failure of this product.
- Due to constant product improvements, the actual product appearance might differ from the shown images.
- Product images are for illustrative purposes only.
- Do not switch the device on immediately after it has been exposed to changes in temperature. Protect the device against damage by leaving it switched off until it has reached room temperature.
- Keep this manual for future reference.

4. Overview

This module allows you to integrate a microcontroller into a Bluetooth® network.



Arduino®		VMA302
5 V	▶	VCC
GND	▶	GND
D11	▶	Rx
D10	▶	Tx

Pin Layout

KEY	if brought high before power is applied, forces AT Command Setup Mode; blinks slowly (2 seconds)
VCC	power supply
GND	ground
TXD	transmit serial data
RXD	receive serial data
STATE	tells if connected or not

frequency ..... 2.45 GHz
asynchronous speed.....max. 2.1 Mbps
security ..... authentication
profile ..... Bluetooth Serial Port
power supply.....+3.3 VDC
working temperature..... max. 60 °C

5. Programming Code

Code begin:

```
// This program shown how to control arduino from PC Via Bluetooth
// Connect ...
// arduino>>bluetooth
// D11   >>>  Rx
// D10   >>>  Tx


// you will need arduino 1.0.1 or higher to run this sketch

#include <SoftwareSerial.h>// import the serial library

SoftwareSerial Genotronex(10, 11); // RX, TX
int ledpin=13; // led on D13 will show blink on / off
int BluetoothData; // the data given from Computer

void setup() {
  // put your setup code here, to run once:
  Genotronex.begin(9600);
  Genotronex.println("Bluetooth On please press 1 or 0 blink LED ..");
  pinMode(ledpin,OUTPUT);
}

void loop() {

  // put your main code here, to run repeatedly:
   if (Genotronex.available()){
BluetoothData=Genotronex.read();
   if(BluetoothData=='1'){   // if number 1 pressed ....
   digitalWrite(ledpin,1);
   Genotronex.println("LED   On D13 ON ! ");
   }
  if (BluetoothData=='0'){// if number 0 pressed ....
  digitalWrite(ledpin,0);
   Genotronex.println("LED   On D13 Off ! ");
  }
}
delay(100);// prepare for next data ...
}
```

Code end

RED Declaration of Conformity

Hereby, Velleman NV declares that the radio equipment type VMA302 is in compliance with Directive 2014/53/EU.

The full text of the EU declaration of conformity is available at the following internet address: www.velleman.eu.

Use this device with original accessories only. Velleman nv cannot be held responsible in the event of damage or injury resulting from (incorrect) use of this device. For more info concerning this product and the latest version of this manual, please visit our website www.velleman.eu. The information in this manual is subject to change without prior notice.

Velleman® Service and Quality Warranty

Since its foundation in 1972, Velleman® acquired extensive experience in the electronics world and currently distributes its products in over 85 countries.

All our products fulfil strict quality requirements and legal stipulations in the EU. In order to ensure the quality, our products regularly go through an extra quality check, both by an internal quality department and by specialized external organisations. If, all precautionary measures notwithstanding, problems should occur, please make appeal to our warranty (see guarantee conditions).

General Warranty Conditions Concerning Consumer Products (for EU):

• All consumer products are subject to a 24-month warranty on production flaws and defective material as from the original date of purchase.

• Velleman® can decide to replace an article with an equivalent article, or to refund the retail value totally or partially when the complaint is valid and a free repair or replacement of the article is impossible, or if the expenses are out of proportion.

You will be delivered a replacing article or a refund at the value of 100% of the purchase price in case of a flaw occurred in the first year after the date of purchase and delivery, or a replacing article at 50% of the purchase price or a refund at the value of 50% of the retail value in case of a flaw occurred in the second year after the date of purchase and delivery.

• Not covered by warranty:

- all direct or indirect damage caused after delivery to the article (e.g. by oxidation, shocks, falls, dust, dirt, humidity...), and by the article, as well as its contents (e.g. data loss), compensation for loss of profits;

- consumable goods, parts or accessories that are subject to an aging process during normal use, such as batteries (rechargeable, non-rechargeable, built-in or replaceable), lamps, rubber parts, drive belts... (unlimited list);

- flaws resulting from fire, water damage, lightning, accident, natural disaster, etc.…;

- flaws caused deliberately, negligently or resulting from improper handling, negligent maintenance, abusive use or use contrary to the manufacturer's instructions;

- damage caused by a commercial, professional or collective use of the article (the warranty validity will be reduced to six (6) months when the article is used professionally);

- damage resulting from an inappropriate packing and shipping of the article;

- all damage caused by modification, repair or alteration performed by a third party without written permission by Velleman®.

• Articles to be repaired must be delivered to your Velleman® dealer, solidly packed (preferably in the original packaging), and be completed with the original receipt of purchase and a clear flaw description.

• Hint: In order to save on cost and time, please reread the manual and check if the flaw is caused by obvious causes prior to presenting the article for repair. Note that returning a non-defective article can also involve handling costs.

• Repairs occurring after warranty expiration are subject to shipping costs.

• The above conditions are without prejudice to all commercial warranties.

The above enumeration is subject to modification according to the article (see article's manual).

Made in PRC
Imported by Velleman nv
Legen Heirweg 33, 9890 Gavere, Belgium
www.velleman.eu