UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DISEÑO DE UN INSTRUMENTO CON INTERFAZ MÓVIL PARA LA MEDICIÓN DE MAGNITUDES ELÉCTRICAS Y MECÁNICAS EN MÁQUINAS ASÍNCRONAS DE 1.1KW

AUTOR: **ALEJANDRO ROMAGUERA ASENSIO**

TUTOR: **JAVIER ANDRÉS MARTÍNEZ ROMÁN**

Curso Académico: **2020-21**

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

ÍNDICE DE DOCUMENTOS

MEMORIA

PRESUPUESTO

ANEXOS

"A mi abuela, ella tendría que haber visto esto.
A Javier, por todo su apoyo y consejos.
A mi familia."

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

DISEÑO DE UN INSTRUMENTO CON INTERFAZ MÓVIL PARA LA MEDICIÓN DE MAGNITUDES ELÉCTRICAS Y MECÁNICAS EN MÁQUINAS ASÍNCRONAS DE 1.1KW

MEMORIA

TRABAJO DE FINAL DE GRADO

ALEJANDRO ROMAGUERA ASENSIO

SEPTIEMBRE 2021

RESUMEN

Hoy en día, el uso de máquinas asíncronas en el entorno industrial es algo habitual y generalizado. Debido a ello, se abre una oportunidad para el desarrollo de sistemas de supervisión de las mismas. Ante esta necesidad, se ha propuesto en el presente Trabajo de Final de grado una solución para la medida de magnitudes eléctricas y mecánicas.

Dicha solución consta de un instrumento con entorno gráfico que permite la lectura de tensión, corriente, velocidad del eje y potencia. Y el consiguiente cálculo del par y rendimiento.

El instrumento consta de un módulo de medida basado en un microcontrolador ESP32, responsable de la lectura, cálculo y envío de datos al entorno gráfico. La representación gráfica es mediante la pantalla de una Tablet que recibe los datos a través de Bluetooth, por lo que no se requiere ningún cable adicional al montaje del módulo de medida en la máquina.

La Tablet posee una aplicación Android que permite la lectura instantánea e histórica de dichas medidas, lo que facilita un análisis rápido del funcionamiento de la máquina.

Palabras clave: Microprocesador, ESP32, Android, Arduino IDE, máquina asíncrona, Tablet.

RESUM

Avui dia, l'ús de màquines asíncrones en l'entorn industrial és una cosa habitual i generalitzat. A causa d'això, s'obre una oportunitat per al desenvolupament de sistemes de supervisió de les mateixes. Davant d'aquesta necessitat, s'ha proposat en el present Treball de Fi de grau una solució per a la mesura de magnituds elèctriques i mecàniques.

Aquesta solució consta d'un instrument amb entorn gràfic que permet la lectura de tensió, corrent, velocitat de l'eix i potència. I el consegüent càlcul de el parell i rendiment.

L'instrument es contitueix d'un mòdul de mesura basat en un microcontrolador ESP32, responsable de la lectura, càlcul i enviament de dades a l'entorn gràfic. La representació gràfica és mitjançant la pantalla d'una Tablet que reb les dades a través de Bluetooth, per la qual cosa no es requereix cap cable addicional a el muntatge del mòdul de mesura en la màquina.

La Tablet posseeix una aplicació Android que permet la lectura instantània i històrica d'aquestes mesures, el que facilita una anàlisi ràpida de el funcionament de la màquina.

Paraules clau: Microprocessador, ESP32, Android, Arduino IDE, màquina asíncrona, Tablet.

ABSTRACT

Today, the use of asynchronous machines in the industrial environment is common and widespread. Due to this, an opportunity opens up for the development of supervision systems for them. Given this need, a solution for the measurement of electrical and mechanical magnitudes has been proposed in this degree Final Project.

This solution consists of an instrument with a graphical environment that allows the reading of voltage, current, shaft speed and power. And the consequent calculation of torque and performance.

The instrument consists of a measurement module based on an ESP32 microcontroller, responsible for reading, calculating and sending data to the graphical environment. The graphic representation is by means of the screen of a Tablet that receives the data through Bluetooth, reason why no additional cable is required other than the mounted module of measurement in the machine.

The Tablet has an Android application that allows instant and historical reading of these measurements, which facilitates a quick analysis of the operation of the machine.

Key words: Microprocessor, ESP32, Android, Arduino IDE, asynchronous machine, Tablet.

Índice

Índice de ecuaciones

Índice de figuras

1. Introducción

El predecesor de los dispositivos inteligentes vio la luz mercantil el 13 de marzo de 1983 desarrollado por la empresa Motorola alcanzó las 300 mil unidades vendidas el primer año, a un precio de 4000 dólares. Ajustando al valor del Índice de Precios al Consumidor, esto sería, actualmente, casi 11000 dólares por un DynaTac.

La aparición del primer smartphone que revolucionaría el mundo vino de la mano de Apple, con el iPhone, en el año 2007. La competencia era feroz, pues en el 2008 se lanzaba el primer dispositivo de sistema operativo Android.

La visión de Google del proyecto Android hizo posible el diseño, creación y distribución de dispositivos a nivel global. Sólo en España se vendieron 1412 millones de smartphones en el año 2019. (Peñalba, 2020)

Las cifras hablan por sí solas, pero cabe destacar que esto permite una conectividad de crecimiento exponencial, pues un smartphone de precio muy asequible, se puede interconectar con numerosos dispositivos inteligentes.

Desde lavadoras hasta televisores, pasando por aplicaciones domóticas, el Internet of Things escala en su uso de manera exponencial. El nicho de mercado abierto para solventar problemas de la sociedad conectada no tiene fronteras visibles.

De semejante panorama, nace el proyecto de diseño de un instrumento con interfaz móvil para la medición de magnitudes eléctricas y mecánicas en máquinas asíncronas. Una herramienta capaz de llevar al bolsillo de cualquier técnico, sala de control o estudiante de prácticas, una medida clara, efectiva y útil de las magnitudes más importantes en dichas máquinas a un precio contenido.

La presente memoria detalla las actividades realizadas para el desarrollo del sistema citado anteriormente.

1.1. Objeto del trabajo

El desarrollo del trabajo de fin de grado tiene como fin la aplicación de los conocimientos teóricos y prácticos adquiridos durante el grado, que se ven amplificados y reforzados debido a la experimentación y aplicación requerida durante el mismo.

De este modo, se asientan conceptos, métodos de trabajo del ingeniero gracias a la excelente labor docente del director de este Trabajo de Fin de Grado, de ahora en adelante TFG. Debido a esto se ponen de manifiesto carencias a solventar, puntos fuertes y favorece el conocimiento del futuro ingeniero.

Es el primer contacto con el mundo real fuera de la universidad, donde se ponen a prueba la adaptabilidad y valía como ingeniero. Para el presente TFG, por ejemplo, ha sido necesaria la profundización en lenguajes de programación de los cuales no se tenía conocimiento alguno.

1.2. Objetivos

El proyecto consiste en sensar diversas magnitudes de funcionamiento una máquina asíncrona y acoplarle un módulo de medidas conectado inalámbricamente a un dispositivo móvil para una representación de las medidas requeridas.

El módulo de medidas está basado en un ESP32, un microcontrolador con tecnología Bluetooth. Éste, tras obtener las medidas programadas y realizar diferentes cálculos, mediante una interfaz de conexión remota inalámbrica, enviará los datos a un dispositivo móvil.

Estas medidas son las magnitudes básicas de una máquina de inducción, tensión, corriente, velocidad, par, potencia absorbida y rendimiento. Dichas medidas permiten hacer un análisis rápido del buen funcionamiento de la máquina. También de sus características de funcionamiento momentáneas, ya que el protocolo de conexión establece un bucle cerrado entre el dispositivo móvil y el módulo de medidas para que estén continuamente enviado/recibiendo datos y mostrándolos al usuario acompañados de su historial reciente.

La interfaz móvil consta de una Tablet de 7 pulgadas, con sistema operativo Android. Pese a que el proyecto tenga un enfoque global, la aplicación móvil está optimizada para dispositivos de 7 pulgadas disponibles en el laboratorio de Máquinas y Tecnología Eléctrica del Departamento de Ingeniería Eléctrica (DIE), de la Universitat Politècnica de València (UPV).

Por tanto, asumiendo todo el proceso como un único proyecto, el objeto del mismo es implementar una mejora en el área práctica tanto del Grado de Ingeniería en Tecnologías Industriales como del Máster de Ingeniería Industrial de la UPV. En concreto, una práctica de toma de medidas de una máquina de inducción, desarrollada en las asignaturas de grado y máster con diferente complejidad, que posee un montaje aparatoso en comparación con la proyección que ocupa a esta memoria.

1.3. Estructura de la memoria

Se puede admitir que el designio se compone de tres apartados principales, el diseño, desarrollo y programación del instrumento de medida, el desarrollo de una aplicación Android y la descripción de los ensayos realizados y de sus principales resultados. Además, la memoria se completa con un capítulo de planteamiento del cálculo en el que se discuten las operaciones a realizar sobre las medidas directas de los sensores con el objetivo de conseguir los valores a mostrar al usuario y, por último, un capítulo de conclusiones en el que se describen los logros alcanzados.

Para cada una de las tres partes principales, se inicia el proceso conociendo la máquina a sensar, así como sus atributos básicos aportados por el fabricante, de modo que se pueda ajustar el diseño del instrumento de medida a rangos aceptables y en los que éste sea útil.

- Módulo de medida: El desarrollo del instrumento consta de varias subsecciones en las que se divide el diseño.
    - Análisis de alternativas del microcontrolador: Este es el elemento más importante del instrumento, por lo que es importante hacer un buen análisis de tecnologías competitivas, en los que se tienen en cuenta precio y capacidades de emisión de datos primordialmente. Ya que las magnitudes a medir se pueden transformar con la electrónica adecuada.
    - Características del microcontrolador escogido: Ventajas e inconvenientes de la tecnología escogida. Para este proyecto se trata de un ESP32

> Elección de sensores:
  → Análisis de alternativas: Tecnologías competitivas y mejor adaptabilidad a las señales que se pretende medir. Para ello se destina un apartado para cada magnitud, dejando el siguiente esquema:
    + Sensores de tensión: Medidas de urs y ust. Para este proyecto se trata de un divisor de tensiones.
    + Sensores de corriente: Miden corrientes de las fases r y t, mediante un transformador de corriente Talema AX-0500.
    + Sensores de velocidad: Toma medidas de velocidad del eje, en este caso, las medidas las toma un *encoder* óptico incremental Omron E6A2-CS5C.

> Diseño electrónico de las entradas y salidas para monitoreado. Una vez escogidos los sensores, es necesario crear un circuito electrónico de adaptación de señal a las características del microcontrolador. Además de tener la precaución de evitar medidas que puedan saturar y, por tanto, dar medidas erróneas.
  → Calibrado de sensores: Cada sensor es único, por lo que es necesario crear un protocolo de cálculo que nos permita tarar el sensor a cero para poder transformar las medidas directas en indirectas y realizar el correspondiente reporte de datos.

> Montaje del diseño de entradas y salidas para monitoreado: El paso más sencillo, pues es seguir el diseño creado de entradas y salidas, en una placa perforada para circuitos.

> Programación del microcontrolador: Cada microcontrolador posee un lenguaje de programación característico, aunque es cierto que son de acceso libre y de *open source*, por lo que es fácil encontrar un lenguaje de programación que se acople al microcontrolador escogido, en este caso Arduino IDE.
  → Captura de información: Las medidas emitidas por los sensores se han de convertir en magnitudes digitales para su reconocimiento y posterior tratado.
  → Tratado de información: La información capturada en el paso anterior introduce el tratado para convertirlo en datos que luego interpretará la aplicación móvil.
  → Envío de datos vía Bluetooth: La aplicación móvil recibe los datos calculados por el microcontrolador mediante la tecnología Bluetooth integrada en el chip.

- Módulo de interfaz móvil: Análogamente al instrumento, posee diversas secciones;

> Justificación del tipo de dispositivo: Necesidades básicas del dispositivo y motivos de su elección. Siendo de poca importancia las necesidades debido a la imposibilidad actual de tener un dispositivo móvil inteligente sin dichas necesidades. Más bien es una introducción a la necesidad y conveniencia de implementar duplas de este estilo con el objetivo de medir magnitudes en máquinas de inducción.

> Programación de la aplicación: En la interfaz móvil no hay ningún diseño de hardware, por lo que consta únicamente de programación. En este caso, el lenguaje está obligado por el propio terminal, se realizará en Android Studio IDE.
  - → Parte lógica: Cálculos y protocolos de comunicación Bluetooth para la recepción de datos en valores digitales y su transformación a valores de ingeniería. Permitirá mostrar con el siguiente apartado, dichos valores al operario del instrumento.
  - → Parte gráfica: Diseño de interfaz amigable, útil y sencilla para la comprensión e identificación de las medidas tomadas por el módulo de medida. Permite un ágil reconocimiento de las magnitudes mostradas, expresadas en valores de ingeniería, en indicadores de aguja en % del valor nominal y también en una gráfica.

- Ensayos y resultados: Describe minuciosamente la mecánica seguida durante el proyecto para realizar ensayos de validación del instrumento y resumir las principales conclusiones respecto al conjunto de su funcionamiento.

1.4. Antecedentes

A fecha anterior al nacimiento de este TFG, el desarrollo de las prácticas y curso del Grado de Ingeniería en Tecnologías Industriales así como del Máster de Ingeniería Industrial impartido por la Universitat Politècnica de València, constaba del siguiente banco de trabajo:

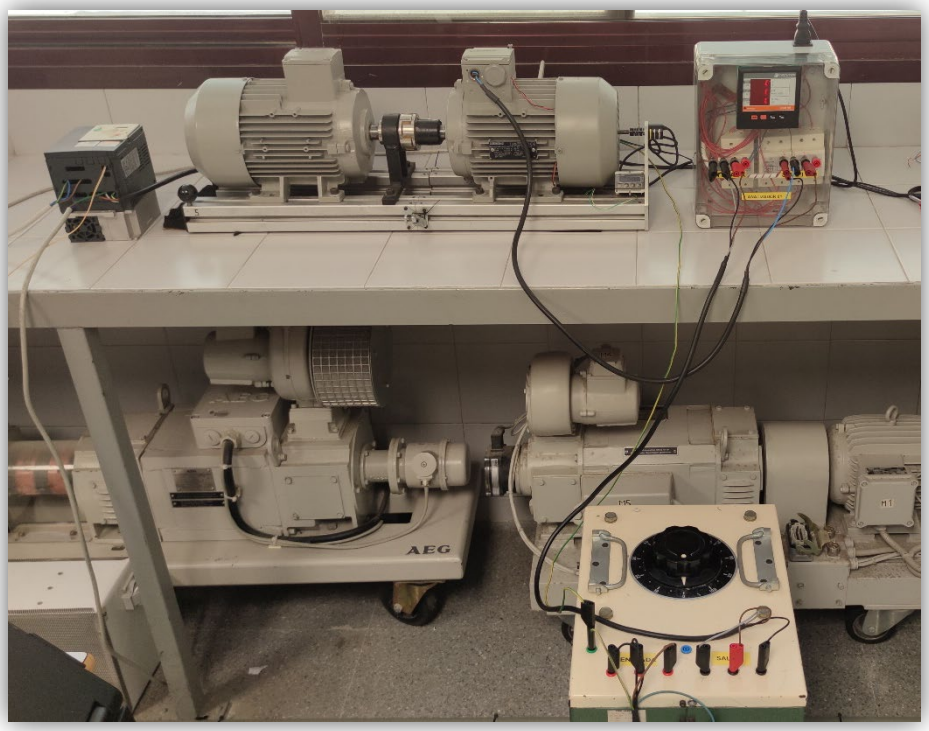


Fig. 1: Banco de ensayos previo. Fuente: Alejandro RA

Éste consta de una máquina de inducción de ensayo, acoplada a una máquina auxiliar alimentada por un variador de frecuencia. Dicha máquina recibe alimentación del autotransformador, pasando simplemente por el analizador de redes.

2. Planteamiento del cálculo

A lo largo del presente TFG, se usará como variable incremental de canal la letra X, de modo que la variable chX, se refiere al canal numerado con la letra X.

Las medidas que se tomarán se pueden diferenciar entre directas e indirectas. Las variables directas son las que pueden ser medidas y se calcularán mediante las ecuaciones que se describen en este capítulo. A raíz de las medidas directas, se obtienen también las indirectas.

Para ello vamos a utilizar fasores espaciales, ya que permiten caracterizar el régimen transitorio de la máquina utilizando una única constante compleja para cada magnitud, en lugar de las tres (para el estator, para el rotor serían tantas como fases o barras del rotor de jaula de ardilla) magnitudes de fase para cada concepto. En régimen permanente equilibrado sin armónicos espaciales significativos, los fasores espaciales tienen una amplitud de $2^{1/2}$ del valor eficaz de las magnitudes de fase. Por ende, las potencias han de dividirse entre 2 cuando se calculan a partir de fasores espaciales de tensiones y corrientes.

Fasor espacial de corrientes:

$$\bar{\imath}_p = \frac{2}{m}\sum_{k=1}^{m} i_k(t) * e^{j(k-1)\gamma} \qquad \text{Ecuación 1}$$

Siendo m el número de fases y el ángulo de desfase entre las fases. Para el caso actual, 2 /m.

El fasor espacial de corriente permite describir la distribución de corriente promedio en el entrehierro en los conductores que forman parte del devanado considerado. Para calcular el fasor espacial, sólo hace falta conocer las corrientes instantáneas en las fases del devanado, evitando utilizar parámetros constructivos. En sistemas de corrientes polifásicas equilibrados, el fasor espacial de corriente presenta una ventaja adicional y es que su proyección sobre el eje de una fase coincide con la corriente instantánea en una fase. (Martínez Román, 2020)

Podemos admitir que, transformando la componente fundamental de ecuación de la capa de corriente, al escoger como sistema referencia fijo (ligado al estator) aquel en el que la fase R produce capa de corriente según el eje real, la ecuación del fasor espacial de corriente queda de la siguiente manera:

$$\bar{\imath} = \frac{2}{3}(i_r + i_s * e^{j120} + i_t * e^{j240}) \qquad \textit{Ecuación 2}$$

En la sección 3.1.3. se detalla que las corrientes de las fases R y T se miden directamente, por lo que la corriente de la fase restante se calculará como:

$$i_s = -i_r - i_t \qquad \textit{Ecuación 3}$$

Para poder utilizar la ecuación fasorial, y las siguientes, será necesario hacer una separación entre parte real e imaginaria, ya que Arduino no permite el cálculo complejo, como se tratará en el siguiente capítulo.

A continuación, será tratado el fasor espacial de tensiones, para ello, se opta por la expresión análoga al de corrientes;

$$\overline{u}_{dev} = \frac{2}{m}\sum_{k=1}^{m} u_k(t) \cdot e^{j(k-1)\gamma} \qquad \textit{Ecuación 4}$$

Esta ecuación se transforma, para poder operar con medidas directas de dos tensiones de línea, en:

$$\bar{U} = \frac{1}{3}(2U_{rs} + U_{st}) + j\frac{U_{st}}{\sqrt{3}}$$ *Ecuación 5*

El fasor espacial de tensiones representa la distribución espacial de tensiones en el conductor medio del devanado que corresponda (en el caso de estudio, el estator) y es la última medida directa. Posteriormente, para el cálculo del par, es necesario conocer, como se observa en la siguiente ecuación, el fasor espacial de enlaces de flujo:

$$T = \frac{3}{2}p * Im[\bar{\imath} \times \bar{\Psi}]$$ *Ecuación 6*

El valor del Par así obtenido no tiene en cuenta ni las pérdidas en el hierro ni las pérdidas mecánicas de la máquina. Ambos aspectos dan lugar a un par equivalente de frenado que es justo el que aparece en el funcionamiento en vacío y que se determinará en esas condiciones para detraerlo del así calculado con el fin de estimar el par en el eje.

Siendo la fem inducida E la que nos permite calcular el citado fasor espacial de enlaces de flujo, utilizando la ecuación eléctrica dinámica del estator (es importante tener en cuenta que, puesto que trabajamos con las corrientes de línea como si fueran de fase, esto equivale a trabajar con el modelo equivalente en Y de la máquina, por lo que la resistencia de estator a utilizar es una tercera parte de la resistencia por fase de la máquina para su conexión real, que es triángulo en nuestro banco de ensayos):

$$\bar{E} = \bar{U} - R_{est} \cdot \bar{\imath}$$ *Ecuación 7*

Y $\bar{\Psi}$ se obtiene por su relación con E en régimen permanente:

$$\bar{\Psi} = \frac{\bar{E}}{-2\pi f_{est} j}$$ *Ecuación 8*

Una vez conocido el par, es posible calcular la potencia útil. Para este cálculo, cabe destacar que se calcula con el par útil, que se consigue, como se acaba de comentar, tras hallar el par de pérdidas ensayando en vacío la máquina. Dicho par aúna las pérdidas en el hierro y las pérdidas mecánicas. Está en torno a 1'1Nm para la máquina con la que se trabaja durante el presente TFG y se restará del par estimado en cada régimen, para dar el útil.

$$P_{ut} = T_{ut} * n * \frac{2\pi}{60}$$ *Ecuación 9*

La obtención de la potencia activa consumida por la máquina dará paso al cálculo del rendimiento.

$$P_{act} = \frac{3}{2}Re[\bar{U} * \bar{I}^*]$$ *Ecuación 10*

$$\eta = \frac{P_{ut}}{P_{act}}$$ *Ecuación 11*

Finalmente, cabe destacar que las corrientes ya están en su valor de línea, por lo que en el cálculo de su valor eficaz no requiere el factor $3^{1/2}$ que si es necesario para calcular el valor de fase de las tensiones.

$$U_{rms} = \frac{\sqrt{3}}{\sqrt{2}}|\bar{U}|$$ *Ecuación 12*

$$I_{rms} = \frac{1}{\sqrt{2}}|\bar{I}|$$ *Ecuación 13*

3. Diseño del Instrumento

En el siguiente capítulo, se desarrollará cuidadosamente cada una de las partes del montaje sobre las que se ha trabajado en el presente TFG. Dividido en dos grandes bloques, el de medida y el de representación, se presenta a continuación el cerebro de la operación y toma de medidas.

Tal y como se ha proyectado la conexión del módulo de medida y el de representación, se utilizará la tecnología Bluetooth.

Bluetooth es una tecnología inalámbrica que permite el intercambio de información entre dos dispositivos a una distancia reducida, de unos 10m. Bluetooth es un estándar a nivel mundial que está implementado en todos los dispositivos con sistema operativo Android.

Nacida en 1994 como sustituta del cable, la conexión permitía una velocidad de 720Kbs en su versión 1. La última versión, la 5.2, permite hasta 50MbS, lo que hace efectiva y conveniente su uso. Además de la velocidad, se puede contar con protocolos de seguridad para afianzar las transmisiones y evitar la pérdida de información.

La seguridad en dispositivos con Bluetooth va de la mano con el aumento del uso de los mismos. A mayor número de usuarios, potencialmente, mayor número de datos vulnerables. El crecimiento exponencial que se intuye en la figura 2 es debido a la creciente conectividad de diferentes dispositivos sobre los que ejercer un control.

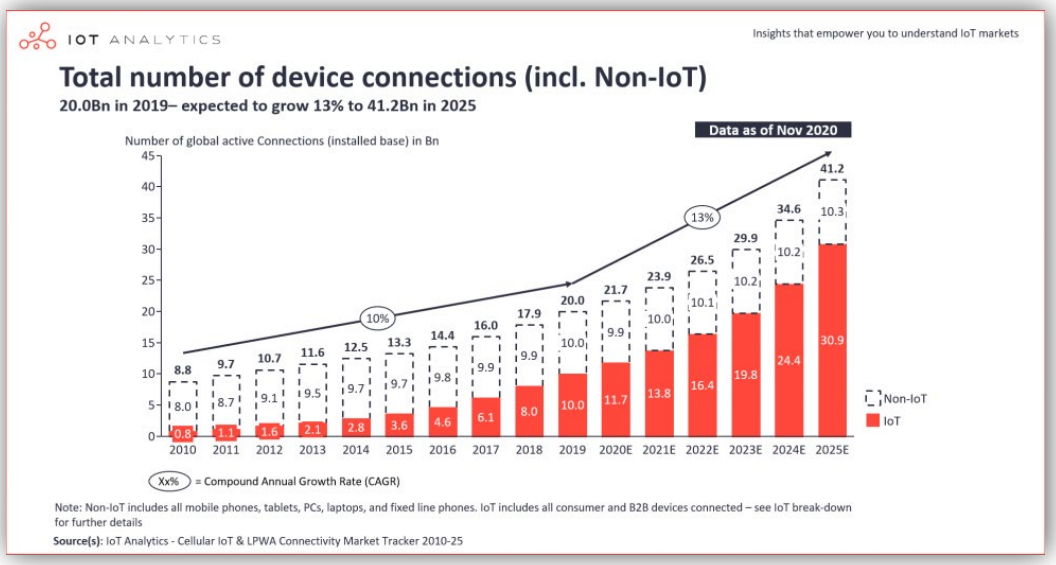


Fig. 2: Previsión de crecimiento de conexiones IoT 2010-2025. Fuente: Segarra Ingenieros.

Actualmente, el desarrollo de las nuevas versiones y protocolos asociados la lleva a cabo la empresa fundada en 1998 por cinco grandes empresas de telefonía, Bluetooth Special Interest Group.

3.1. Módulo de medida

3.1.1. Características del ESP32

Desarrollado por Espressif, una empresa multinacional con sede en Shanghái, con un modelo empresarial sin fábrica. Es la responsable también del conocido ESP8266, ampliamente utilizado hasta la llegada del ESP32. Es un SoC compatible con Arduino, lenguaje con el que se realizará la programación. Un SoC, *System on a Chip*, es una agrupación de herramientas que dotan a un microprocesador de competencias.

Su uso desde noviembre de 2016 ha demostrado que es un dispositivo robusto, fiable y versátil en distintas aplicaciones. Además posee una elevada cantidad de funcionalidades.

El ESP32 WROOM-32 DEVKIT V1 es un chip que combina wifi y Bluetooth 2.4 GHz de 16mW de potencia (versión clásica 4.2 y de bajo consumo BLE). Se basa en un microprocesador *Tensilica Xtensa LX6* de hasta 240MHz si se realiza un *overclocking*. Posee un *dual core* de 32 bit, 448KB de ROM y 520KB de SRAM. El microprocesador es la unidad de procesamiento central, CPU. Es el responsable de las operaciones, procesos y cálculos. Su parámetro más importante, junto al ancho de banda, es la velocidad de procesado. Estos ofrecen rapidez en el cálculo y operar con números reales eficentemente.

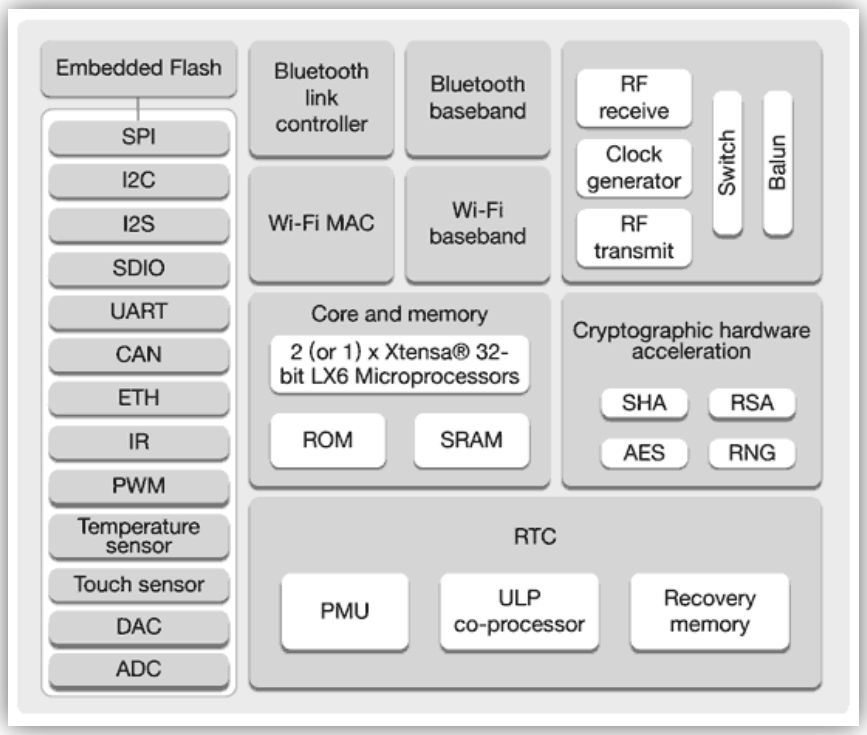


Fig. 3: Estructura del SoC ESP32. Fuente: Randomnerdtutorials.

Como se observa en la figura 3, el SoC incluye una gran variedad de funcionalidades, lo que lo hacen extremadamente polivalente.

Dispone de 36 *General Purpose Input/Output*, GPIO, programables, sobre los que se pueden utilizar *pull_down*, como se detallará en el código, y así usar pulsadores, etc. Estos son cada uno de los pines sobre los que se producirán las conexiones. Dos *Digital-Analog Converter*, DAC, de 8 bit y un ADC de 12 bit y 18 canales. Así como varias SPI avanzadas como se observa en la figura 4.

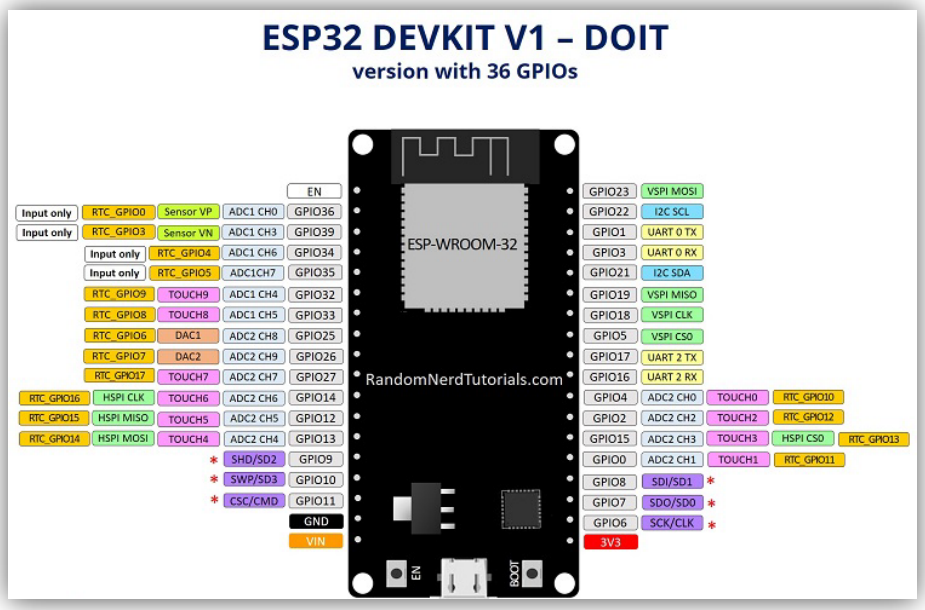


Fig. 4: Pin-out de desarrollo del ESP32. Fuente: Randomnerdtutorials.

Permite desarrollar proyectos complejos. Con un consumo energético bajo, es utilizado en aplicaciones no conectadas a la red eléctrica así como en *Internet of Things*, IoT.

Presenta también un sensor de efecto hall. Requiere una tensión de alimentación de 5V que estabiliza internamente a 3'3V de salida, con un uso mínimo de corriente de 500mA.

El ESP32 será el responsable de medir las magnitudes eléctricas y mecánicas de la máquina asíncrona, hacer los cálculos pertinentes y enviarlos a la app que será descrita más adelante.

3.1.2. Justificación de la elección del ESP32

Existen numerosas alternativas en el mundo del control para implementar proyectos similares al que se trata en el presente TFG, como alternativas al ESP32, es posible contar con su antecesor, el ESP8266 y el Arduino UNO R3.

El ESP8266 presenta un microprocesador de 32 bit *Tensilica L160* de hasta 160MHz en *overclocking*. 16 GPIO. Un ADC de 10 bit y no dispone de DAC.

En estándares de comunicación, cabe destacar la presencia de Wifi así como la ausencia de Bluetooth. Para alimentarlo se requieren 5V y 3'3V a la salida. Es compatible con Arduino IDE, igual que el ESP32.

Es un chip ampliamente usado, ya que rápidamente se convirtió en un básico para los programadores de Arduino.

Como alternativa, se alza el Arduino UNO R3. Esta placa pertenece a la empresa Arduino. La cual nace de la necesidad del acercamiento de placas de desarrollo para su uso masivo, pese a ser la más cara de las tres alternativas.

Actualmente también da nombre a una gran comunidad de software libre. Su funcionalidad está basada en la adición de componentes llamados *shields*, que permiten implementar tecnologías o conexiones no incluidas por defecto. Es una placa altamente compatible, sencilla y robusta.

El microcontrolador alojado en la placa es un *ATmega328P,* 16MHz. Apresta un total de 14 pines digitales y 6 analógicos. Una memoria flash de 32KB y 2KB de SRAM. Un ADC de 10 bit. Salida de 3.3V.

En resumen, se ha escogido el ESP32 debido a su bajo coste y elevado poder computacional a la par que funcionalidades integradas necesarias para el proyecto.

	Placa	ESP32	ESP8266	Arduino UNO
Característica				
Núcleos		2 de 32 bit	1 de 32 bit	1 dde 32 Byte
Velocidad		240MHz	160MHz	16MHz
GPIO		36	16	20
Wifi		Sí	Sí	Shield
Bluetooth		Clásico y BLE	No	Shield
RAM		SRAM 520Kb	160Kb	32KB + SRAM 2KB
ADC		12 bit 18 canales	10 bit	10 bit
DAC		2 de 8 bit	No	Shield
Entrada		5V	5V	5V
Salida		3,3V	3,3V	3,3V
Precio		9 €	7 €	20 €

Fig. 5: Tabla comparativa de chips. Fuente: Alejandro RA.

3.1.3. Diseño de los sensores y adaptación de señal

Durante el presente capítulo, será desarrollado en profundidad cada uno de los sensores escogidos así como una valoración de tecnologías alternativas que el SoC puede gestionar. Para ejemplificar rápidamente la elección de cada uno de los sensores, las figuras siguientes responden a las diferencias entre tecnologías. La electrónica de la placa sobre la que está instalada el ESP32 tiene un diseño debido a la naturaleza de las medidas que se van a tomar. Por lo que es imprescindible que cumplan una serie de requisitos técnicos.

3.1.3.1. Sensores de tensión

Debido al gran abanico de sensores de tensión de tecnologías muy similares o basadas en la misma con pequeñas variaciones, el siguiente capítulo destinará dos subapartados a dos tecnologías muy diferentes y de amplia aplicación en entornos industriales.

3.1.3.1.1. Transformador de tensión

El transformador de tensión es una máquina eléctrica estática cuya función es relacionar dos tensiones según la relación de espiras del mismo. El principio se basa en la variación de flujo creado por la corriente alterna del devanado primario que induce una corriente en el bobinado secundario. La ecuación que rige la relación de tensiones es lineal y es la siguiente:

$$\frac{V_p}{V_s} \cong \frac{N_p}{N_s}$$ *Ecuación 14*

Es una construcción robusta y de elevado rendimiento.

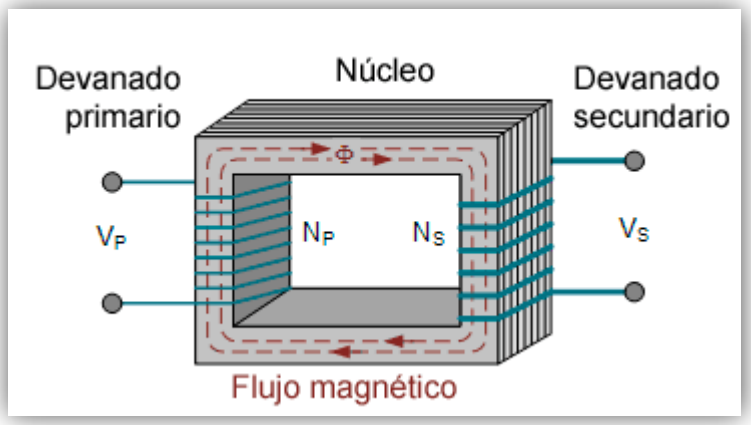


Fig. 6: Esquema de un transformador de tensión. Fuente: Arismex.

3.1.3.1.2. Divisor de tensiones

Un puente divisor de tensiones es un método de medida de tensión lineal. Es de implementación muy sencilla y barata, basada en la ley de Ohm.

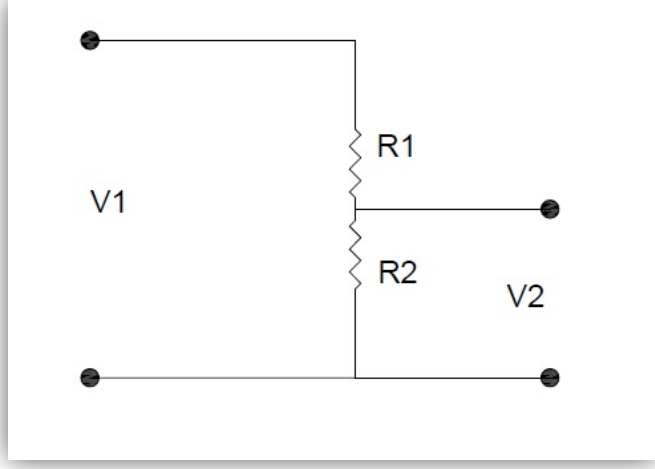


Fig. 7: Divisor de tensiones. Fuente: Alejandro RA.

Se puede calcular fácilmente V2 en función de V1:

$$V_2 = \frac{R2}{R1 + R2} V_1$$ *Ecuación 15*

El divisor de tensiones es el método utilizado en el proyecto actual.

3.1.3.2.Sensores de corriente

Para tratar el sensado de corriente se estudiarán diversas alternativas de tecnología muy madura. Basadas en diferentes principios físicos. Entre ellos está el efecto Hall, la ley de Lenz, etc.

La tabla comparativa no arroja mucha luz sobre las diferentes alternativas. Debido a la similitud entre los diferentes dispositivos de sensado, se ha optado por la opción más económica tras descartar la resistencia *shunt* debido a sus problemas inherentes a esta tecnología para valores bajos y especialmente debido a que no ofrece aislamiento galvánico entre el circuito principal y el de medida.

T. corriente	T. Hall	R. *shunt*	B. Rogowski
Sin pérdidas en el conductor principal	Sin pérdidas en el conductor principal	Pérdidas en el conductor principal	Sin pérdidas en el conductor principal
Fácil instalación	Fácil instalación		
Seguridad	Seguridad		Seguridad
Elevada precisión	Elevada precisión	No excesivamente precisas	Elevada precisión
Amplio rango		Amplio rango	Amplio rango
Hasta 1kHz	Hasta 50kHz		Media a alta frecuencia
Lineal	Lineal	Lineal	Lineal
4 €	5 €	0'3 €	6 €

Fig. 8: Tabla comparativa sensores de corriente. Fuente: Alejandro RA.

Fig. 9: Transformador de corriente escogido para el proyecto, Talema AX0500. Fuente: Digikey.

3.1.3.2.1. Transformador Hall

El principio de funcionamiento está basado en el efecto Hall. Una partícula cargada con una velocidad determinada, experimentará una fuerza de valor;

$$F_m = q(v\ x\ B)$$ *Ecuación 16*

De modo que, al aplicar una diferencia de potencial entre los bornes de una placa metálica, existirá un desvío de cargas por la acción de dicho campo magnético. Esto genera una tensión perpendicular a la corriente aplicada llamada voltaje Hall.

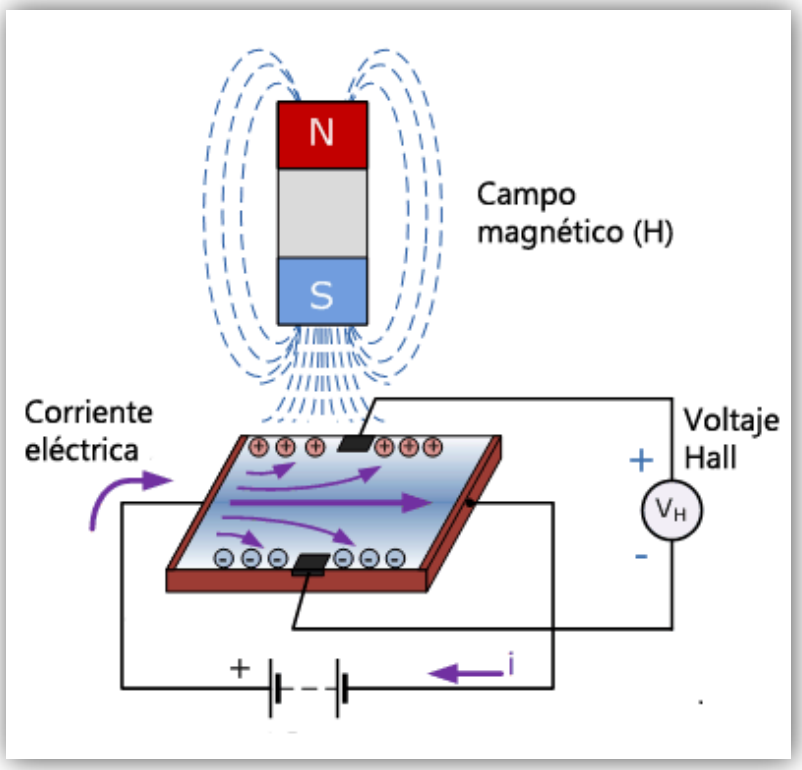


Fig. 10: Efecto Hall. Fuente: Wikipedia.

La ecuación que rige la tensión es la que sigue:

$$V_H = v * B * d$$ *Ecuación 17*

Siendo v la velocidad de las cargas a través de la placa. Y d la anchura de la placa. B la intensidad del campo magnético.

En los sensores de efecto Hall existe una subdivisión, de lazo abierto y cerrado. Operan bajo el mismo principio pero los de lazo cerrado aumenta las prestaciones, incrementando el precio.

La placa que contiene el voltímetro Hall se halla en un núcleo ferromagnético arrollado por dentro del cual pasa el conductor sobre el que se desea realizar la medida. Un ejemplo de esos sensores son las pinzas amperimétricas de efecto Hall.

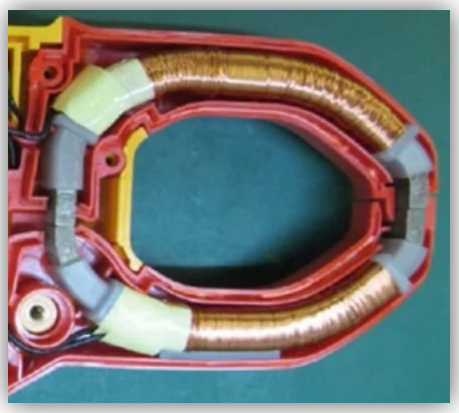


Fig. 11: Pinza amperimétrica de efecto Hall. Fuente: Fluke, modificada.

Los sensores de lazo cerrado poseen una bobina extra encargada de eliminar la medida de la fuerza de Lorentz para acondicionar la señal. La bobina extra se alimenta mediante un amplificador operacional de elevada ganancia.

3.1.3.2.2. Resistencia Shunt

Las resistencias *shunt* son simplemente resistencias de conocido valor óhmico por las que pasa la corriente a medir. Sobre esta resistencia, se pueden realizar medidas de tensión para conocer, mediante la ley de Ohm, la corriente que circula por la misma. Se instalan en la línea principal, lo que conlleva una interrupción de la línea y la pérdida de aislamiento entre el circuito de potencia y el de medida. Presentan problemas de ruidos si son de muy bajo valor. Por tanto, es necesario encontrar una solución óptima para el circuito sobre el que se implementará.

Fig. 12: Resistencia shunt. Fuente: Alibaba.

A la hora de emplearlo en el ESP32 se puede admitir que supone una aparamenta extra para la medida de la tensión y así poder aplicar la ley de Ohm. Todo ello duplicado, al medir dos corrientes.

3.1.3.2.3. Transformador de corriente

Como tercera alternativa, se disponen los transformadores de corriente, basados en la tecnología de un transformador de tensión convencional. Es decir, una corriente variable en el tiempo, en este caso senoidal, genera un campo magnético variable a su alrededor de ecuación:

$$B(t) = \frac{\mu I(t)}{2\pi r} \qquad \textit{Ecuación 18}$$

Siendo r la distancia al conducto y μ la permeabilidad magnética. Se inducirá una corriente variable de características proporcionales a la primaria en la línea secundaria.

Esta vez añadiendo una resistencia entre los bornes para seguir el mismo proceso que en las resistencias *shunt* para el cálculo de la intensidad admisible por la placa. En el proyecto actual se realiza este cálculo en el 3.1.3., para el caso de evitar la saturación.

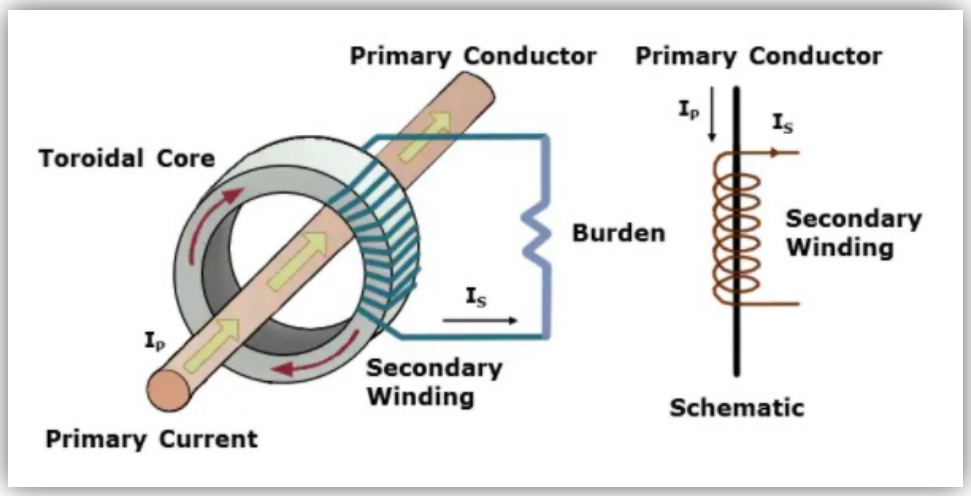


Fig. 13: Transformador de corriente toroidal. Fuente: Acotron.

Igualmente al transformador de tensión, en base a la relación de espiras y corriente primaria, aparecerá una corriente calculable en el secundario, en este caso, nuestro sensor.

Son fácilmente instalables, no introducen pérdidas en el conductor principal y se pueden ajustar a rangos muy dispares. Poseen una carcasa de protección, pero debido a su principio de funcionamiento, son inherentemente seguros.

3.1.3.2.4. Bobina de Rogowski

La bobina de Rogowski es un arrollamiento de conductores sobre un núcleo diamagnético. Se dispone con un orificio por el que pasará el conductor. La bobina, al formar un circuito cerrado

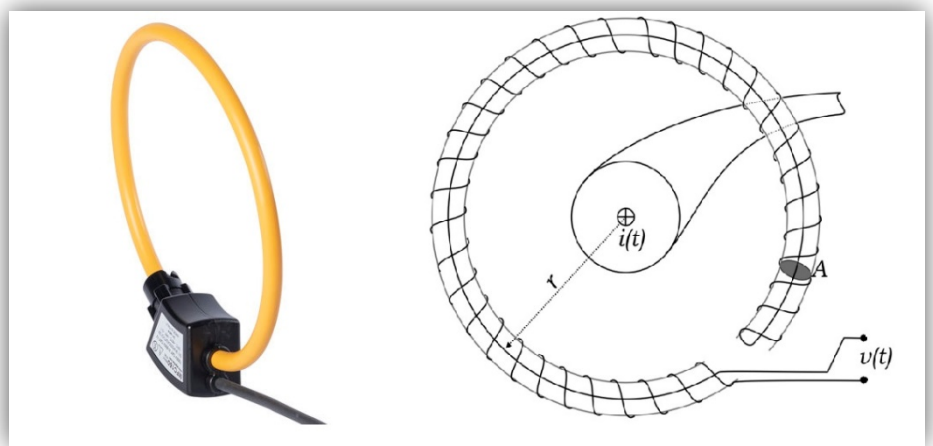


Fig. 14: Bobina de Rogowski. Fuente: Wikipedia y Farnell, modificadas.

alrededor del conductor principal, no le afecta la forma ni la posición del mismo dentro de la bobina.

Se basa en la ecuación siguiente:

$$V(t) = \mu * N * A$$ *Ecuación 19*

Siendo N el número de vueltas y A la sección del núcleo. La teoría en la que se basa es en que la inductancia propia y mutua de la bobina así como la capacitancia están distribuidas a lo largo de la bobina de manera uniforme.

En caso de no disponer de un elemento integrador para conocer la corriente dentro del propio transductor, se ha de recurrir a un puente externo.

3.1.3.3. Sensor de velocidad

Para la medida de la velocidad del eje de la máquina de inducción se dispone de varias alternativas, entre las que se estudiarán; la dinamo tacométrica, *encoder* óptico, de efecto hall y magnético.

Es crucial escoger bien el sensor, así como programar acorde a sus características para poder calcular acertadamente la velocidad de giro. Para ello conviene realizar un análisis de alternativas tecnológicas y escoger el que más se ajuste al propósito

Tacodinamo	Ópt. incremental	Ópt. Absoluto	Mag. incremental	Mag. absoluto
Lineal	Lineal	Lineal	Lineal	Lineal
Robusta	Robusto	Robusto	Robusto	Robusto
Estable	Estable	Estable	Estable	Estable
Amplio rango	Amplio rango	Amplio rango	Amplio rango	Amplio rango
	Sensibilidad a la suciedad	Sensibilidad a la suciedad		
Sentido de giro		Sentido de giro		Sentido de giro
		Posicionamiento de eje		Posicionamiento de eje
320 €	160 €	492 €	385 €	415 €

Fig. 15: Tabla comparativa de encoder. Fuente: Alejandro RA.

Se puede apreciar que este tipo de sensor mantiene unas características constructivas básicas de fiabilidad. Pero el salto de precio entre el escogido para el proyecto y el siguiente más barato es el doble de precio. Para esta aplicación no es necesario conocer la posición del eje en todo momento, tampoco el sentido de giro. Además, como se detalla en el capítulo 4, ya estaba instalado en el montaje original. Por tanto, la elección, basada en criterios técnicos y económicos es la tecnología óptica incremental.

3.1.3.3.1. Dinamo tacométrica

Las dinamos tacométricas son máquinas eléctricas rotativas que, acopladas a un eje, generan una salida proporcional a la velocidad de giro.

El principio de funcionamiento de una dinamo es el movimiento de un estátor sobre el que se inducen corrientes debido a la variación de flujo entrante a las espiras del mismo. Por tanto, al girar el estátor en un campo magnético fijo, en el colector del estator se apreciarán tensiones de forma pulsante.

Fig. 16:Tacodinamo. Fuente: Alibaba.

La onda pulsante se puede rectificar mediante un filtro resistencia-condensador, que dará una salida rizada según la agresividad del filtrado. La señal de salida será proporcional a la velocidad de giro. Son transductores altamente lineales, que gozan de un gran rango en el cual son precisas y estables. Además dan información sobre el sentido de giro del eje.

La señal en azul sería la tensión en bornes de la tacodinamo sin que hubiese un filtro de rectificado, una onda pulsatoria de frecuencia proporcional a la velocidad de giro. Sería una tecnología viable. En naranja, tras aplicarle un filtro RC, aparece una tensión casi contínua, de rizado ajustable según precisión del transductor para la aplicación determinada. En este caso, se necesitaría también un amplificador de tensión.

Fig. 17: Tensión de salida de una tacodinamo con rectificación y sin. Fuente: Alejandro RA

3.1.3.3.2. Encoder óptico

Los *encoder* ópticos son constituidos por uno o varios discos tallados por láser que presentan unos huecos o ventanas uniformes y equidistantes que, al interrumpir un haz de luz generan una onda de pulso cuadrada cuando el eje gira a velocidad constante. Se acoplan al eje de manera directa. La resolución de los mismos depende del número de entallas que posea el disco, se mide en pulsos por revolución.

Fig. 18: Encoder rotativo Omron E6A2-CS5C, 200 pulsos por revolución. Fuente: Omron.

3.1.3.3.2.1. Encoder incremental

El *encoder* incremental es un tipo de *encoder* óptico simple. No lleva codificación ninguna, es barato, sencillo de instalar y no presenta problemas a diferentes regímenes de velocidad. No poseen información de la dirección de rotación, tampoco de la posición del eje.

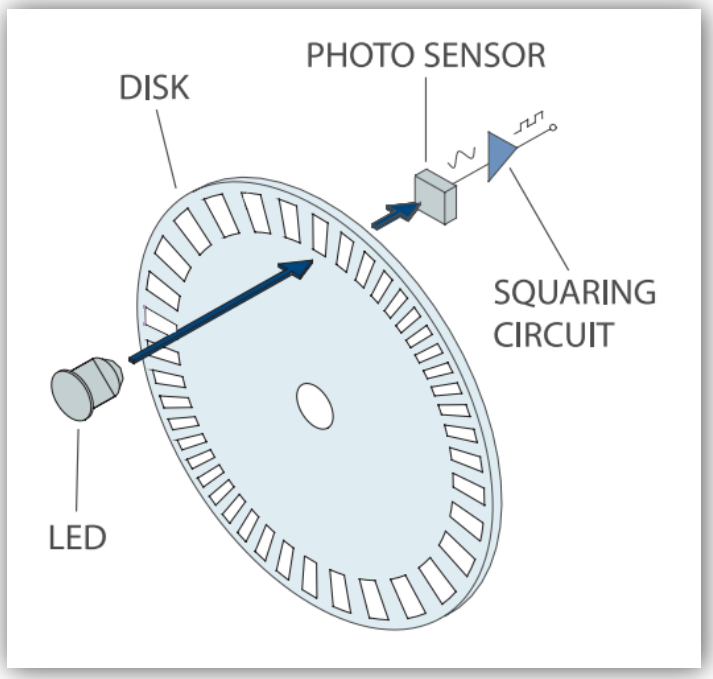


Fig. 19: Funcionamiento encoder incremental. Fuente: Logicbus.

3.1.3.3.2.2. Encoder absoluto

Este tipo de *encoder*, presenta un disco más complejo que el anterior y una conjunción de fotosensores. Esta tecnología cuenta con el conocimiento contínuo de la posición del eje, por lo que sería fácil averiguar el sentido de giro.

La complejidad del disco radica en la presencia de una codificación en el mismo. Éste posee diferentes entallas para cada anillo que corresponde con un fotorreceptor diferente. Dicha codificación se puede realizar mediante diversos métodos para facilitar la implementación del indicador de pulsos. Una mecánica de codificación es el código Gray, ampliamente utilizado en electrónica.

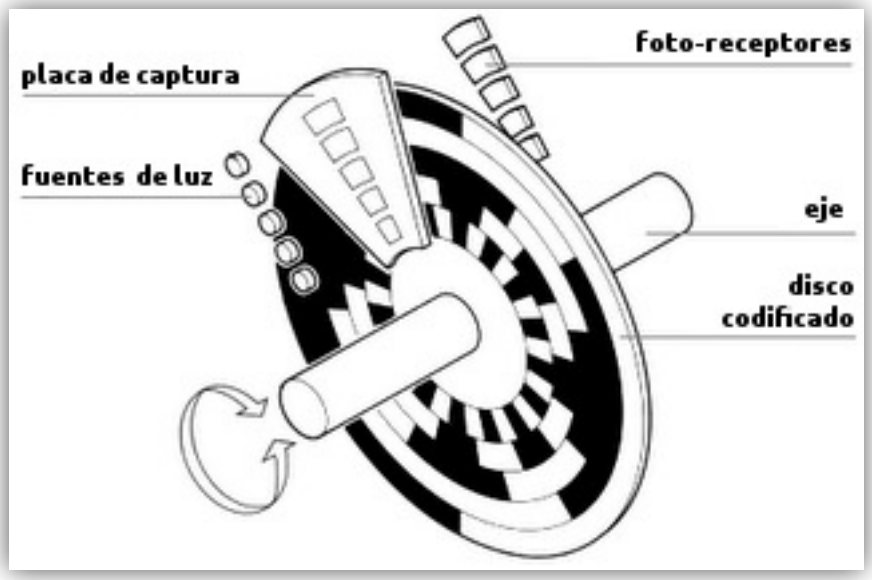


Fig. 20: Funcionamiento de un encoder absoluto de 5 bit. Fuente: Researchgate.

Este tipo de sensores, denominados absolutos no pierden información sobre el estado del eje en caso de una falta de alimentación del sensor.

El código Gray permite la ordenación de los valores binarios, de modo que se puede extrapolar cada uno de los bits a cada uno de los anillos. De este modo, se puede conocer la velocidad, la posición y el sentido de giro del eje. Consiste en que dos dígitos consecutivos solamente disten entre si un bit. No como ocurre en binario, que entre el 3 y el 4 decimal, hay una diferencia de dos bit.

Decimal	Binario 4 bit				Grey 4 bit			
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	0	1	1	0	1
10	1	0	0	1	1	1	1	1
11	1	0	1	0	1	1	1	0
12	1	0	1	1	1	0	1	0
13	1	1	0	0	1	0	1	1
14	1	1	0	1	1	0	0	1
15	1	1	1	0	1	0	0	0

Fig. 21: Código binario y Gray de 4 bit. Fuente: Alejandro RA.

A simple vista destaca que el anillo más exterior corresponde con el primer bit, pues medio disco posee una ventana y el resto es macizo.

3.1.3.3.3. Encoder magnético

El funcionamiento este tipo de transductores se basa en el mismo principio de conteo de pulsos. Pero en vez de recibir un haz de luz, el detector detecta una polarización magnética. Igualmente que los discos de los *encoder* ópticos, poseen un disco con una codificación magnética, a menudo se les llama *encoder* Hall.

La tensión Hall es la que detectará el cabezal del *encoder* cuando el campo magnético generado por el anillo proporcione ese desvío de cargas. La medida será proporcional a la velocidad de giro.

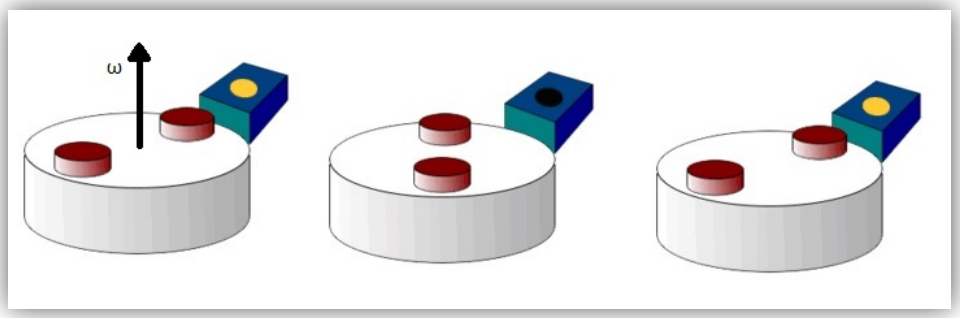


Fig. 22: Detección magnética de la velocidad. Fuente: Wikipedia, modificada.

Como se aprecia en la figura superior, la codificación del anillo permite la detección, para el disco de la imagen sólo hay dos bit.

Los *encoder* magnéticos no tienen desgaste, ya que no poseen elementos rozantes, son sensores de elevada aplicabilidad industrial. No presentan fallos de medida ante entornos sucios y permiten una resolución muy elevada.

3.1.3.3.3.1. Encoder incremental

Análogamente al *encoder* óptico incremental, el magnético presenta las mismas ventajas e inconvenientes prácticamente.

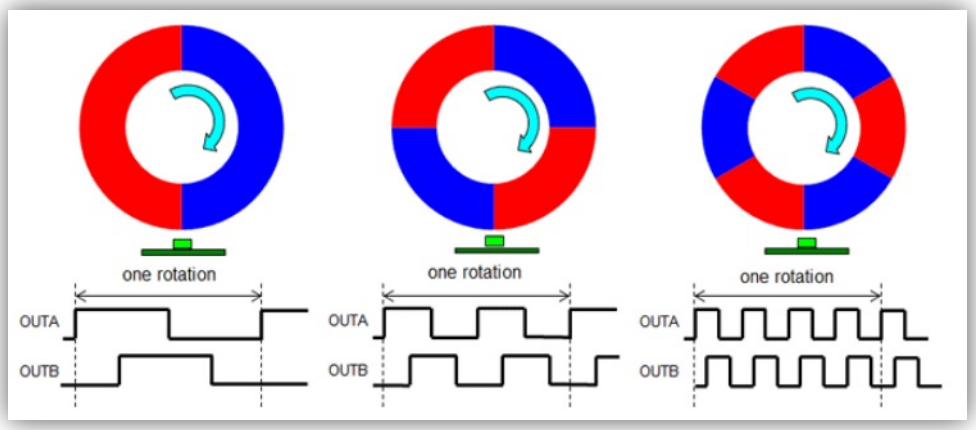


Fig. 23: Encoder Hall incremental. Fuente: Akm.

3.1.3.3.3.2. Encoder absoluto

Para un *encoder* de estas características, la tecnología presenta un doble sensor, que permite conocer el sentido de giro, ya que los pulsos entre sensores llevarán un desfase calculado mediante la codificación del anillo en cuestión. Los magnéticos absolutos también poseen información de la posición del eje.

3.1.4. Descripción de las entradas y salidas

Pese a que el ESP32 posee 36 pines de entrada/salida para el control, únicamente se hará uso de siete. Las magnitudes a medir son las corrientes de las tres fases y las tensiones de dos líneas calculadas a partir de las tensiones de tres fases respecto de la referencia externa (la tercera corriente se calcula derivada de las otras dos al tratarse de un sistema trifásico sin conexión de neutro).

Estos pines serán referenciados en el programa en Arduino que es tratado en el punto 3.1.6.

Pin	Entrada	Salida	Magnitud
D23	X		Pulsos encoder
32	X		it
35	X		ir
34	X		ut
36	X		ur
39	X		us
3,3V		X	
GND			

Fig. 24: Pines de entrada y salida ESP32. Fuente: Alejandro RA.

La siguiente figura muestra la disposición del montaje del ESP32, así como sus conexiones, que serán detalladas para una visión completa de la placa diseñada. Asimismo, serán comentadas brevemente las magnitudes proporcionadas por los sensores, sin entrar en la discusión de elección de los mismos, pues el subcapítulo 3.1.4. está destinado a ello.

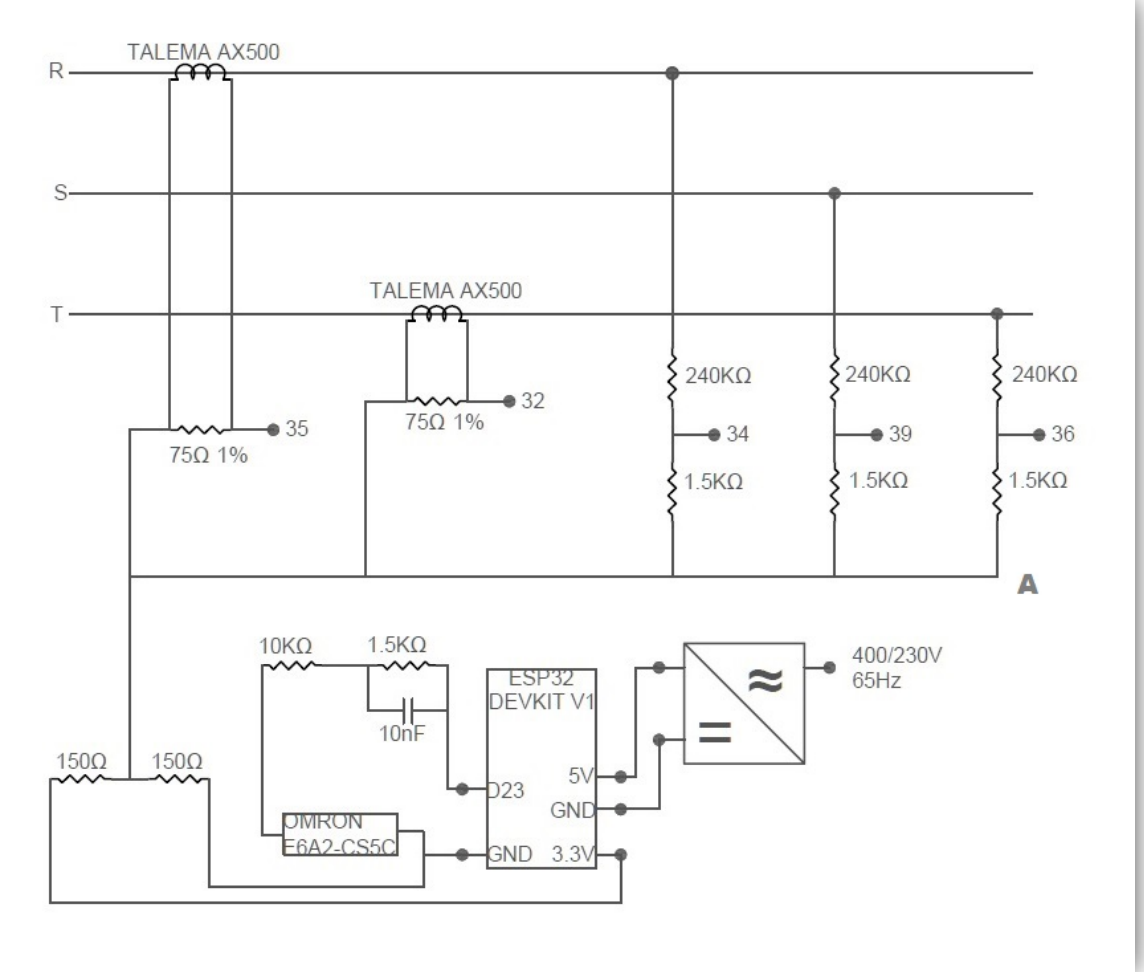


Fig. 25: Esquema conexiones de sensores al ESP32. Fuente: Alejandro RA.

El ESP32, como se reflejó anteriormente, tenía una tensión de alimentación de 5V en contínua, que en el montaje proporciona una fuente de alimentación conmutada de uso general. Si se subdivide el esquema en tres partes, sean estas;

- Velocidad: Para la medida de la velocidad se ha dispuesto un *encoder* óptico rotativo incremental que da una señal de onda cuadrada. Tal y como se ve en la figura 26. Los pulsos son pasados por un filtro de paso bajo de primer orden, esto facilita no dar falsas lecturas. Para la implementación del filtro, se ha optado por un condensador de 10nF, que establece una frecuencia de corte de 10'6KHz, suficiente para no distorsionar en exceso la onda pulsante.

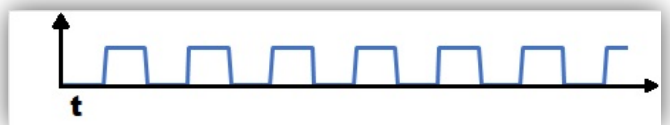


Fig. 26: Onda cuadrada. Fuente: Alejandro RA.

- Corriente: Mediante un sensor de corriente Talema AX500, se realiza la medida directa de las corrientes de las fases R y T, es un transformador de corriente de orificio pasante. Tiene una relación de corrientes primario secundario 500:1, por lo que, si la máquina asíncrona consume alrededor de 4'4A en régimen nominal, presentará picos de 6'22A. Apoyados en la gráfica aportada por el fabricante;

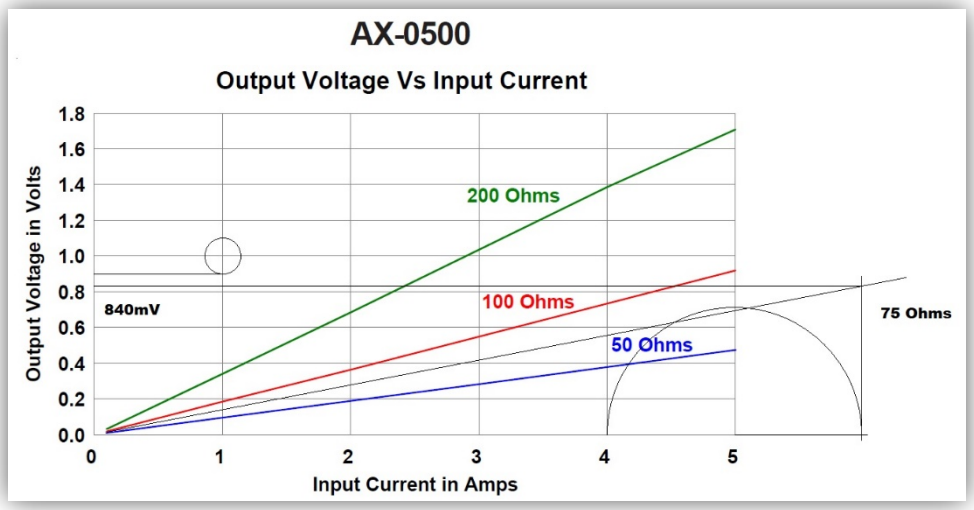


Fig. 27: Relación corriente de entrada y tensión de salida transformador de corriente. Fuente: Digikey, modificada.

Disponiendo la resistencia de 75 , la tensión en bornes del transformador es de unos 840mV, lejos de la saturación, como se verá más adelante. Finalmente, el ESP32 podrá leer la corriente por los pines 32 y 35.

- Tensión: Destinados a la lectura de la tensión, se dispone un divisor de tensiones en cada una de las fases. Atendiendo a la ecuación del mismo en el punto A, se puede concluir que:

$$V_A = \frac{230 * \sqrt{2}}{\sqrt{3}(240 + 1'5)E3} * 1'5E3 = 1'1664V \qquad \textit{Ecuación 20}$$

Ésta será la tensión de pico que se indica por parte de dicho divisor. La elección de las resistencias es debido a que se ha de reducir la posibilidad de saturación del ADC. La tensión será medida por el SoC mediante los pines 34, 36 y 39. Se medirán tensiones entre fases y referencia para calcular posteriormente dos tensiones de línea, en concreto urs y ust.

Como se puede observar, dispuesto entre la salida de 3'3V y tierra del ESP32, se encuentra un divisor de tensiones, que proporcionará:

$$V_{divsalida} = \frac{3.3}{2 * 150} * 150 = 1'65V \qquad \textit{Ecuación 21}$$

Debido a que el SoC trabaja a 3'3V, los convertidores no son ajenos a ello, si se admite que, al ser tensiones senoidales trifásicas equilibradas, al aparecer 1'1664V de tensión de pico como máxima, esto será para valores negativos también. En consecuencia:

$$V_{sat} = 3'3V \gg 1'65 + 1'1664 = 2'816V \qquad \textit{Ecuación 22}$$

Evitando un error de lectura por saturación del ADC en el caso más desfavorable por un amplio margen.

Conviene destacar también que el comportamiento de los pines del ADC no es lineal. Por lo que podría ser difícil reconstruir valores en los extremos de rango. La gráfica ilustrativa es la siguiente:

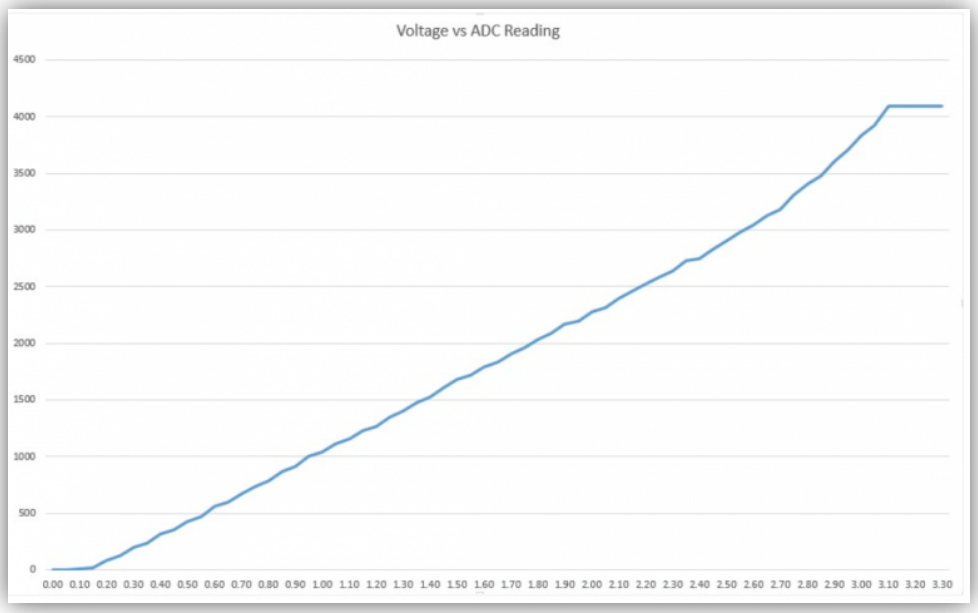


Fig. 28: Comportamiento no lineal de un ADC de 12bit. Fuente: docs.iotaap.

Por lo tanto, será una cuestión a solventar para poder trabajar de la forma más lineal posible en la parte de calibrado de sensores.

3.1.5. Calibrado de los sensores

Los canales de adquisición de datos del microcontrolador poseen un valor por defecto que corresponde a una tensión, provienen del transductor utilizado. Ésta se suma al valor de lectura, alterando enormemente la misma. Para ello, se realiza un ensayo en el cual la máquina asíncrona no está alimentada, estando únicamente el ESP32 a tensión. Siguiendo, se le obliga a reportar vía Bluetooth los valores que está leyendo por los canales destinados a la toma de muestras.

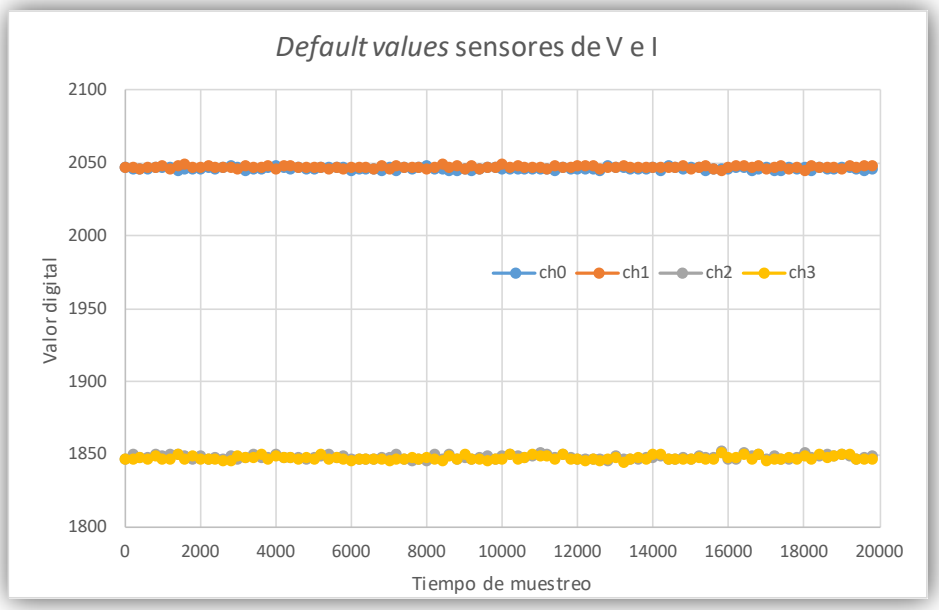


Fig. 29: Gráfica de valores por defecto de los canales de medida. Fuente: Alejandro RA.

La tabla incluida en el anexo, permite graficar los valores por defecto para utilizarlos en el cálculo posteriormente. Los dos primeros canales son los que corresponden a las tensiones y los otros dos a las corrientes. Realizando un promedio de los cien valores muestreados, se asignan esos valores como valores medios del canal.

A continuación, mediante un muestreo en vacío, se verifica el correcto funcionamiento de los sensores. La tabla incluida en el anexo tiene como resultado la gráfica que se muestra a continuación.

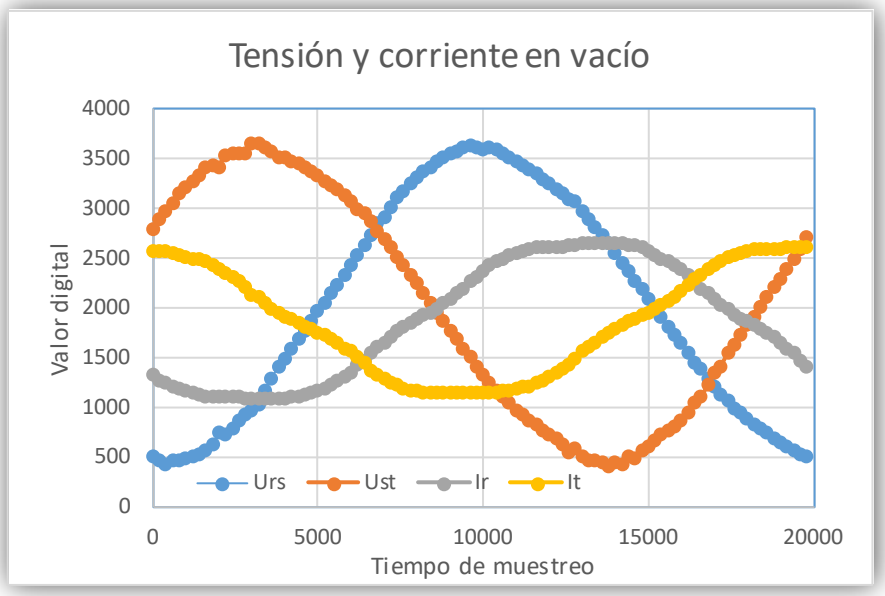


Fig. 30: Gráfica de valores digitales de tensión y corriente en vacío. Fuente: Alejandro RA.

Como se observa en la gráfica, las ondas son relativamente senoidales y a priori, coherentes con lo que cabría esperar; se observan dos parejas de tensiones y corrientes de similar amplitud y desfase cercano a 120º como corresponde a un sistema trifásico equilibrado. Para estos valores, es conveniente calcular su valor eficaz y así poder deducir el factor de escala a utilizar en cada canal. El factor de escala responde al valor por el que se debe multiplicar una medida digital para conseguir el equivalente analógico.

$$Fe = \frac{Medida_{canal\ (externa)}(eficaz)}{Medida_{canal\ (digital)}(eficaz)} \qquad \textit{Ecuación 23}$$

Una vez calculado el factor de escala, se realiza un cálculo de las magnitudes que más tarde implementará el programa de Arduino en unidades de ingeniería. Tras revisar la coherencia de las mismas, comparando con los datos ofrecidos por el fabricante incluidos en el anexo, se concluye la validez del calibrado de los sensores.

3.1.6. Programación

En el presente subcapítulo, se tratará la programática del microcontrolador para la toma de muestras, cálculo y envío de datos.

3.1.6.1. Arduino IDE

La programación del cerebro del proyecto se ha llevado a cabo mediante el entorno de programación Arduino IDE, un *software* de programación que admite programación en C y C++ con algún cambio en la sintaxis concreta para el control. Es un entorno de *open source*, por tanto, gratuito y con una gran comunidad de programadores que lo respalda.

El portal que permite descargarlo es https://www.arduino.cc/en/software

Las siglas IDE responden a Entorno de Desarrollo Integrado, es decir, que el propio entorno posee herramientas para facilitar la programación. Entre ellas se encuentran las librerías o la posibilidad de compilar, montar y transferir el programa al micro y recibir las salidas desde él, por ejemplo mediante la conexión serie que se utiliza para el volcado.

Las librerías o bibliotecas informáticas son un conjunto de métodos o implementaciones de operaciones comunes a las que se puede recurrir para ahorrar programar una funcionalidad cada vez que lo precisas o que no se disponen de forma nativa en el entorno de programación. Tanto es así que al crear una actividad, automáticamente Arduino IDE incluye librerías por defecto.

3.1.6.2. Instalación de la placa

Para poder trabajar sobre una placa, es necesario indicarle al entorno sobre qué placa se está trabajando. En el presente caso, ESP32 DEVKITV1, como se ha comentado con anterioridad.

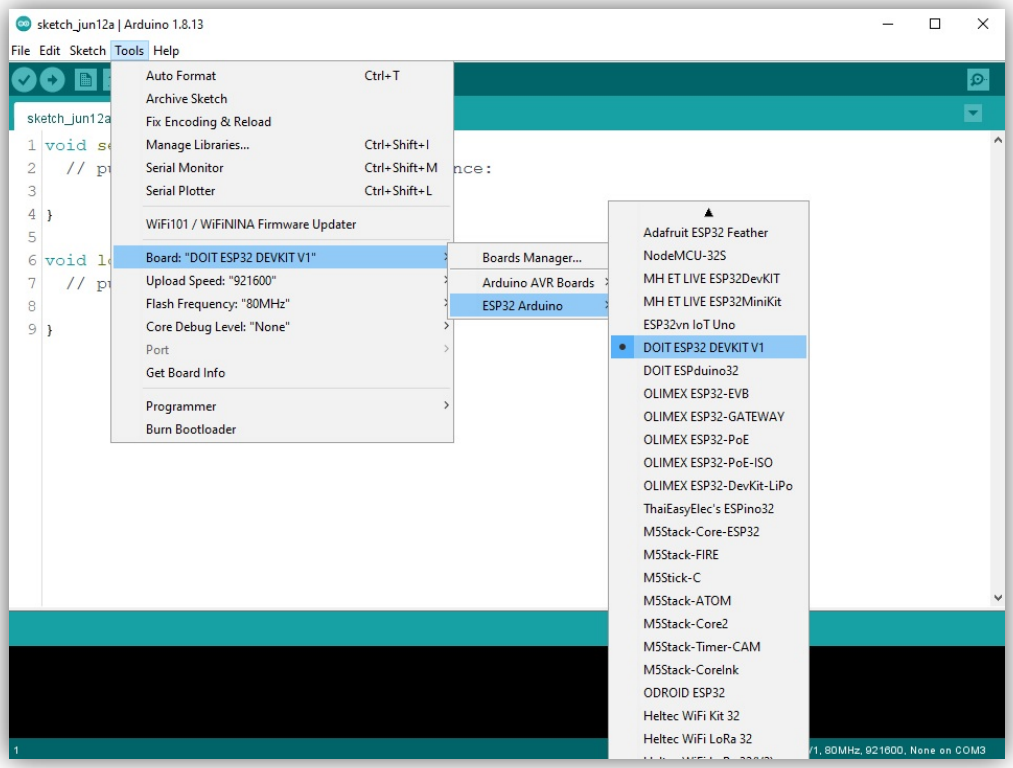


Fig. 31: Arduino IDE. Fuente: Alejandro RA.

Esto facilitará la solución de las compatibilidades características de la placa así como disponer de funcionalidades concretas para sacarle el máximo rendimiento a la misma.

3.1.6.3.Estructura de la aplicación Arduino

Con la intención de realizar una explicación pormenorizada del programa instalado en la placa, cabe analizar la estructura básica de un programa .ino.

De la figura 32, se observa una parte del código, *setup,* donde se escribe la inicialización del programa, que se ejecuta al inicio y una única vez. Seguidamente, el *loop* o bucle, se ejecutará de manera indefinida. Cabría destacar que las dos funciones van precedidas de la palabra void, que indica que no producen una variable de retorno.

Mi_primer_programa

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

Fig. 32: Estructura básica de un programa Arduino. Fuente: Alejandro RA.

El ESP32 del que se dispone, posee dos núcleos, lo cual permitiría desarrollar tareas de manera simultánea, en este caso de captura y emisión de datos, pero debido a la naturaleza del proceso a implementar, no es necesario hacer uso del *dual core*. El bucle *loop()* va a recibir, tratar y enviar datos de forma secuencial *ad infinitum* o hasta que se desconecte la app Android.

Tras una segmentación clara en *setup* y *loop*, cabe proceder con todo aquello que se necesita para poner a punto el código, aquí se encuentra el método para el cálculo de la velocidad, que se ha debido programar para ser usado en el *loop*. También los métodos de captura de información, cálculo y emisión de datos.

3.1.6.3.1. Método de cálculo de n

Mediante una tabla, se ilustran las diferentes variables que Arduino soporta para su declaración. Se asigna según uso de cada variable declarada la naturaleza adecuada.

Tipos de Datos	Memoria que ocupa	Rango de valores
boolean	1 byte	0 o 1 (True o False)
byte / unsigned char	1 byte	0 – 255
char	1 byte	-128 – 127
int	2 bytes	-32.768 – 32.767
word / unsigned int	2 bytes	0 – 65.535
long	2 bytes	-2.147.483.648 – 2.147.483.647
unsigned long	4 bytes	0 – 4.294.967.295
float / double	4 bytes	-3,4028235E+38 - 3,4028235E+38
string	1 byte + x	Array de caracteres
array	1 byte + x	Colección de variables

Fig. 33: Variables declarables Arduino. Fuente: Aprendiendoarduino.

Una vez se tiene una visión global de las variables que se utilizarán, el método creado para el cálculo de la velocidad se presenta como en la figura 34.

```
7  #define SAMPLECOUNT 100
8  #define alpha 0.01
9  short samples = SAMPLECOUNT;
10 short ch0[SAMPLECOUNT], ch2[SAMPLECOUNT], ch1[SAMPLECOUNT], ch3[SAMPLECOUNT], epr[SAMPLECOUNT];
11 float n, ccf;
12 int encdt[SAMPLECOUNT];
13 short cont = 0;
14 int bc, cc;
15
16 void IRAM_ATTR count(){
17   int nc = ESP.getCycleCount();
18   cc = nc-bc;
19   if(cc > 45000){
20     encdt[cont]=cc;
21     bc = nc;
22     ccf = (1-alpha)*ccf+alpha*cc;
23     n = 240e6/ccf/200*60;
24     cont++;
25     if(cont>=samples) cont=0;
26     }
27 }
```

Fig. 34: Método de acceso a la RAM y cálculo de la velocidad. Fuente: Alejandro RA.

La definición de la línea 7, SAMPLECOUNT, se trata de un número, en concreto el número de muestras por dato con el que se realizarán los cálculos y muestreo. Al usar cien valores por muestra enviada a la app, se evitan en gran parte las variabilidades que puedan presentarse en una medida instantánea.

Tras la creación de los canales de tensión, ch0 y ch1. Análogamente los de corriente, ch2 y ch3. Finalmente, epr guarda el conteo de pulsos.

Para el cálculo de la velocidad, conviene declarar variables intermedias para desarrollar con precisión la misma, se realizará mediante una consulta y operación con los ciclos de reloj del ESP32.

- encdt[SAMPLECOUNT]: Almacenará el conteo de ciclos entre flancos de subida del encoder.
- anaread[SAMPLECOUNT]: Custodiará el valor de la diferencia de tiempos entre muestras tomadas.
- bc: *Before count* guarda el ciclo en el que se encuentra el microprocesador antes del conteo, servirá de origen de tiempos.
- cc: *Cycle count* conservará el número de ciclos que han pasado.
- ccf: C*ycle count final* preservará el número de ciclo tras la consulta.
- nc: *Now count* indica el ciclo en el que se encuentra el microprocesador actualmente.
- cont: Será un simple contador de repeticiones, índice de encdt[SAMPLECOUNT].

Según la documentación, el ESP32 está basado en una arquitectura Harvard, esto conlleva que dispone de un canal de datos y otro de instrucciones, los cuales pueden leer la RAM del microprocesador. Mediante el método IRAM_ATTR se implementa una lectura de datos de RAM mediante el bus de datos, ya que se van a alojar las funciones que sean consideradas en la RAM. De este modo, pese a perder memoria, es posible asegurar que el acceso a esos datos será constante. (Kolban, 2018)

Esto es crucial para las consultas requeridas de ciclos de reloj. Como se verá más adelante, en el 3.1.4.3., el *encoder* tiene un paso de 200 pulsos por revolución.

La consulta se realiza de la siguiente manera, nc consulta mediante el método *ESP.getCycleCount()*. El resultado de la diferencia entre el número de ciclos actual y anterior a la consulta es el elemento condicional de la estructura que le sigue. Si esta diferencia es mayor a 45000, que es un poco más bajo que el número de ciclos mínimos que se dan a velocidad de sincronismo, es decir 1500rpm, se realiza el cálculo. Si no lo es se descarta la llamada a interrupción ya que se entiende que se ha producido por ruido en la entrada asociada y no por un flanco del encoder.

$$\Delta Ciclos_{min} = \frac{f_{\mu}}{Pulsos_{rps} * rps} = \frac{240E6 * 60}{200 * 1500} = 48000 ciclos \qquad \textit{Ecuación 24}$$

Por tanto se descartarán llamadas antes de 45000 de ciclos de la CPU. Éste constituirá el primer filtro digital del montaje. En caso que hubiese una interrupción como la de la figura 35, no se realizaría la consulta

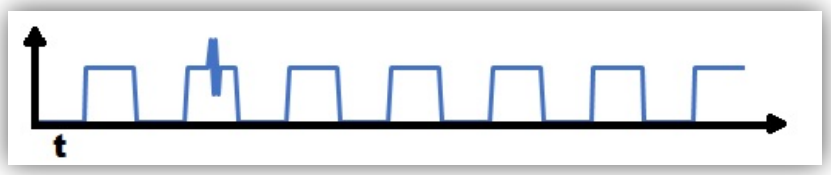


Fig. 35: Onda cuadrada con interferencia. Fuente: Alejandro RA.

Una vez dentro de la estructura *if* es posible ilustrar mediante dos trazas el código de la figura 36.

Línea	cont	bc	encdt[cont]	nc	cc	ccf	n	nactual-nantes
16	0	0	-	-	-	0	-	
17	0	0	-	48649	-	0	-	
18	0	0	-	48649	48649	0	-	
19	0		-	48649	48649	0	-	
20	0	0	48649	48649	48649	0	-	
21	0	48649	48649	48649	48649	0	-	
22	0	48649	48649	48649	48649	486,49	0	
23	0	48649	48649	48649	48649	486,49	147998,93	147998,93
24	1	48649	48649	48649	48649	486,49	147998,93	
16	499	24275851	48649	24275851	48649	48645,3128	1480,10149	
17	499	24275851	48649	24324500	48649	48645,3128	1480,10149	
18	499	24275851	48649	24324500	48649	48645,3128	1480,10149	
19	499	24275851	48649	24324500	48649	48645,3128	1480,10149	
20	499	24275851	48649	24324500	48649	48645,3128	1480,10149	
21	499	24324500	48649	24324500	48649	48645,3128	1480,10149	
22	499	24324500	48649	24324500	48649	48645,3497	1480,10149	
23	499	24324500	48649	24324500	48649	48645,3497	1480,10037	-0,001121884
24	500	24324500	48649	24324500	48649	48645,3497	1480,10037	

Fig. 36: Traza 1 y 500 del método de cálculo de la velocidad. Fuente: Alejandro RA.

Se incian los cálculos asumiendo que se parte desde cero en la línea temporal, esto sitúa el bc=0, ccf=0 y el cont=0, permitiendo a las demás variables tener asignado un número aleatorio, pues no afecta al cálculo de la velocidad. Partiendo también de que la máquina asíncrona tiene una velocidad de 1480 rpm, por lo que;

$$nc = \frac{240E6 * 60}{200 * 1480} = 48649 \qquad \textit{Ecuación 25}$$

Es posible observar que el proceso tal y como se muestra en el código anterior, induce a una diferencia en la primera traza entre la velocidad calculada y la real en un factor 100. Podría parecer un error, pero hay que tener en cuenta que es la primera aproximación que realiza el microprocesador, tras 500 iteraciones, se presenta una diferencia entre la velocidad real y la calculada de 0'1rpm. Cabe destacar que la resolución del indicador de velocidad en el montaje actual es de 1rpm. Además, pese a que 500 iteraciones puedan parecer cifras astronómicas de cálculo;

$$\frac{nc_{500}}{f_{microprocesador}} = \frac{24324500}{240E6} = 0'1014\ segundos \qquad \textit{Ecuación 26}$$

Es decir, que el ESP32 estabiliza el cálculo con un error del 0'0068% en algo más de 100milisegundos.

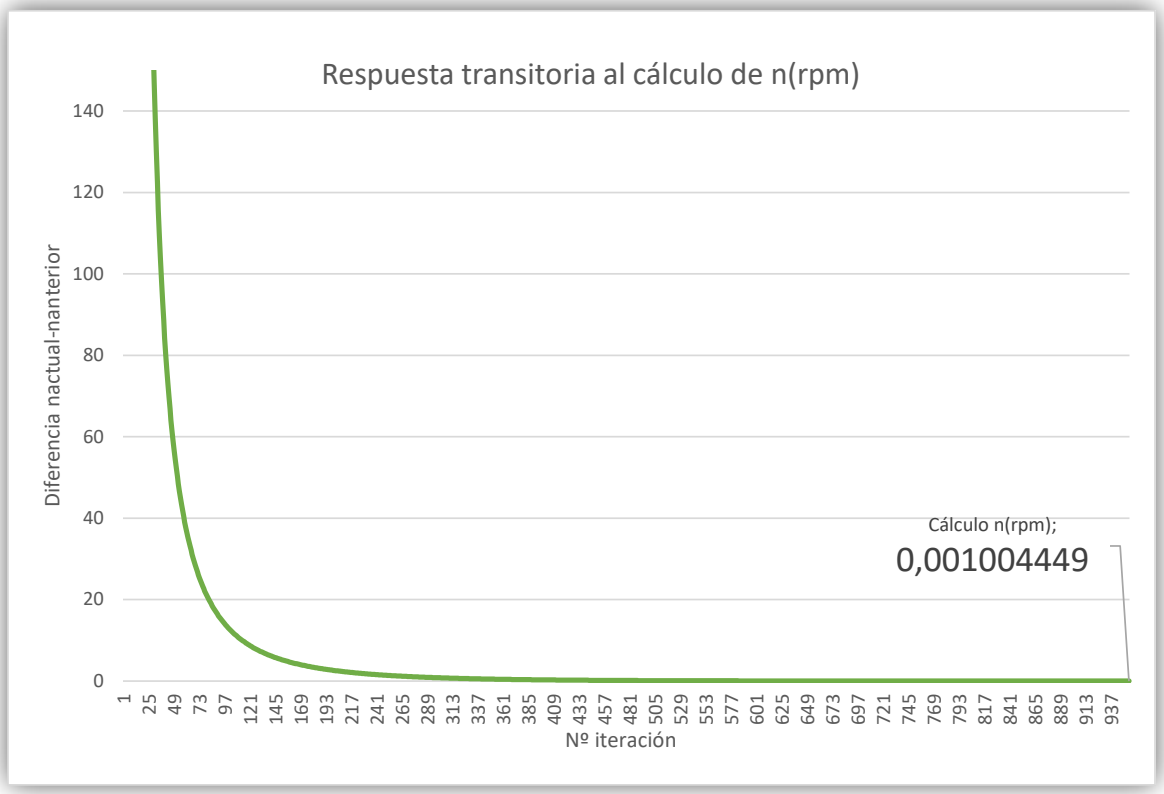


Fig. 37: Transitorio del cálculo de velocidad del ESP32. Fuente: Alejandro RA.

Atendiendo a la respuesta transitoria del cálculo en la gráfica anterior, se infiere que alrededor de las 1000 iteraciones, es decir 200milisegundos, la diferencia entre una medida y la siguiente tendría un error del 0'1%. Esta estructura de cálculo es similar a un filtrado paso bajo de tipo Respuesta Infinita a Impulso (IIR) de orden 1 con la diferencia de que ese tipo de filtrado se define para señales digitalizadas a tasa de muestreo constante. En el caso actual, la tasa de *operación* del filtro equivalente depende de la frecuencia de pulsos del *encoder* óptico y, por lo tanto, de la velocidad de giro de la máquina. Como esta, en todos los estados de interés, es muy cercana a 1500 r.p.m. se puede considerar que es prácticamente constante y de ahí la analogía con un IIR de orden 1.

3.1.6.3.2. Setup

```
#include <BluetoothSerial.h>
BluetoothSerial ESP_BT;

void setup() {
  //Serial.begin(9600);
  ESP_BT.begin("MaqAsinc_001");
  //Serial.println("Bluetooth Device is Ready to Pair");
  pinMode(23,INPUT);
  bc = ESP.getCycleCount();
  pinMode(23,INPUT_PULLDOWN);
  attachInterrupt(23,count,RISING);
  analogSetAttenuation(ADC_11db);
}
```

Fig. 38: Setup de la aplicación Arduino. Fuente: Alejandro RA.

La puesta a punto del programa se puede realizar una vez incluida la librería con las facilidades Bluetooth que permitirán establecer la conexión con la app. También así se define el puerto serie equivalente a la conexión Bluetooth, esto es, la instrucción que indicará al SoC que esa línea de código ha de enviarla mediante Bluetooth, en esta situación, ESP_BT. El dispositivo se denomina MaqAsinc_001, y así aparecerá para cualquier dispositivo equipado con radio Bluetooth. Para ello, se ha de iniciar la emisión mediante el método *begin*.

La primera de las entradas que serán empleadas, será la entrada correspondiente a la velocidad, la número 23. Para definir que es una entrada, en el método *pinMode*, hay que destacarlo, mediante la palabra reservada *INPUT*, entrada. Seguido de la línea bc, que almacenará el ciclo inicial del ESP32, para que cuando se produzca la consulta que lleva al cálculo de la velocidad, bc tenga un valor que dé sentido al código, pues en caso contario, erraría. Para adaptar con la electrónica del *encoder*, que según la bibliografía posee el colector abierto, se ha de disponer programáticamente la entrada 23 a un *PULLDOWN.* Esto es, simplemente, establecer el estado de reposo de la entrada como un cero lógico. Es posible hacer uso del método creado con anterioridad (Figura 38), gracias a la asociación del pin 23 con una interrupción, *attachInterrupt*, en el flanco de subida, esto es, cuando el *encoder*, dé un pulso. Finalmente, se introduce una atenuación estándar para entradas analógicas en converidores AD.

3.1.6.3.3. Captura de información

El método de captura es el encargado de leer las entradas del divisor de tensiones y el transformador de corriente. Para ello dispone de las variables anaXX, que serán los canales analógicos de lectura. Para una aclaración rápida, la parte declarada quedaría así:

- ana40: Canal analógico de lectura de la tensión de la fase r, pin 36.
- ana41: Canal analógico de lectura de la tensión de la fase s, pin 39.
- ana42: Canal analógico de lectura de la tensión de la fase t, pin 34.
- ana43: Canal analógico de lectura de la corriente de la fase r, pin 35.
- ana44: Canal analógico de lectura de la corriente de la fase t, pin 32.
- nowus: Destinado a almacenar el tiempo de reloj actual en microsegundos.
- nextus: Análogamente a nowus, para el siguiente instante de captura.
- endus: Análogamente a nowus, para el último instante de captura.

```
64 void capture(){
65         //Serial.println("sampling started");
66         unsigned int nowus, nextus;
67         //unsigned int endus;
68         unsigned short ana40, ana41, ana42, ana43, ana44;
69 
70         for(short i =0; i<samples;i++){
71           if(!i) {
72             anaread[0]=0;
73             nowus = micros();
74             nextus = nowus + deltat;
75           }
76           else anaread[i] = micros()-nowus;
77 
78           ana40 = analogRead(36);
79           ana41 = analogRead(39);
80           ana42 = analogRead(34);
81           ana43 = analogRead(35);
82           ana44 = analogRead(32);
83 
84           ana44 += analogRead(32);
85           ana43 += analogRead(35);
86           ana42 += analogRead(34);
87           ana41 += analogRead(39);
88           ana40 += analogRead(36);
89 
90           ana40 = (ana40-ana41)>>2;
91           ana41 = (ana41-ana42)>>2;
92           ana43 = (ana43+2)>>1;
93           ana44 = (ana44+2)>>1;
94 
95           ch0[i] = ana40+2048;
96           ch1[i] = ana41+2048;
97           ch2[i] = ana43;
98           ch3[i] = ana44;
99           epr[i] = round(10*n);
100          //endus = micros();
101 
102          while(nextus > micros()){}
103          nextus +=deltat;
104                }}
```

Fig. 39: Captura de información Arduino. Fuente: Alejandro RA.

Seguidamente a la declaración de las variables superiores, destaca un bucle de longitud igual a *SAMPLECOUNT*, habilitando la realización de esa captura centenaria. Para ello cabe inicializar el vector *anaread*[SAMPLECOUNT], que se encuentra declarado en la línea 62, con su posición inicial a cero. Asimismo, establecido el origen temporal de captura mediante el método integrado *micros()*, que devuelve el instante en microsegundos en el que la placa de Arduino se encuentra desde la inicialización del programa. De este modo, el siguiente periodo se calcula con tan solo sumar el incremento de tiempo anunciado en deltat, línea 61. Se puede recalcar que la primera parte del condicional sólo se ejecuta si i=0.

La lectura de la información proporcionada por los pines se realiza en secuencia directa e inversa de forma consecutiva, como se advierte en las líneas 78 a 82 primero y 83 a 87 después. Esta forma de leer los valores analógicos es especialmente importante al querer utilizar las medidas de tensión y de corriente para realizar cálculos de potencia, que son muy sensibles al desfase entre tensión y corriente.

Si se hiciera una sola lectura, existiría un desfase temporal entre la muestra de un canal y del siguiente asociado al tiempo de ejecución de la función *anaread().* Ese desfase daría lugar a errores en la valoración de la potencia si los canales consecutivos fueran de tensión y de corriente, ya que se asociaría una cierta tensión instantánea a una corriente que se ha leído, en realidad, un poco después.

Para evitar esto se hace una lectura en secuencia directa de los canales seguida de una en secuencia inversa: al tomar el valor medio de ambas lecturas de cada canal, se puede aceptar, dada la variación lineal aproximada de tensiones y corrientes en ese intervalo tan breve de tiempo, que las lecturas de todas la tensiones y corrientes se corresponden con el valor promedio en el instante central de la captura para todos los canales. Es decir, todos ellos coincidentes en el tiempo. Esta estrategia añade además un ligero filtrado de las medidas de todos los canales al trabajar con el valor medio de dos muestras. Los valores captados son valores de línea, cosa a recordar en el apartado de cálculos. Finalmente se adicionan los datos de la lectura directa e inversa.

Debido al acúmulo de las variables en una sola, se han de dividir los valores. Ésta operación se desempeñará mediante el desplazamiento de bits. El lenguaje de máquina no tiene interés para el presente TFG, por lo que no será objeto de profundización más que en las consecuencias. Un desplazamiento de un bit, constituye una división por 2, mientras que un desplazamiento de dos, por cuarto.

Si se admite la medida diferencial de ondas senoidales equilibradas en el origen, las medidas de tensión poseen el doble de rango, pues en el caso más desfavorable, el SoC estaría leyendo el máximo en las dos, cosa que situaría la medida muy por encima del valor del ADC, tal y como ilustra la figura a continuación.

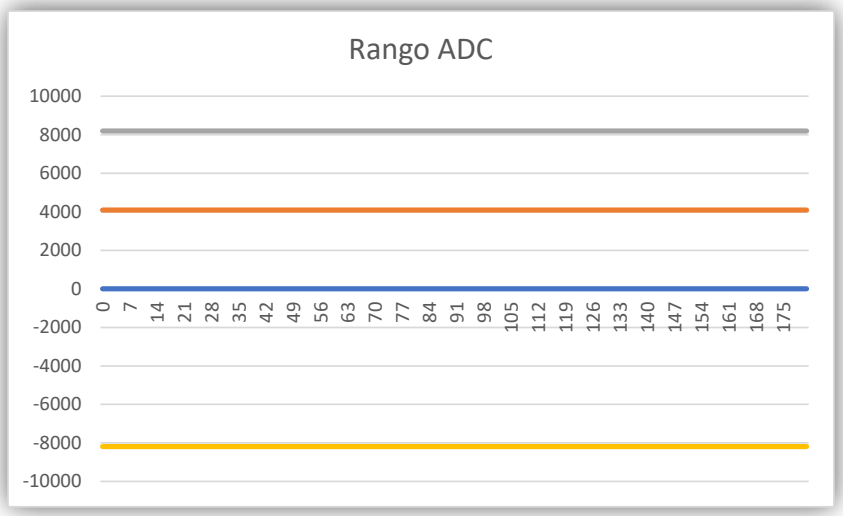


Fig. 40: Rango ADC de 12bit. Fuente: Alejandro RA.

Como se aprecia en la figura 40, el rango máximo del ADC son 4095, por lo que las medidas de tensión, que son de fase, hemos de desplazarlas dos bit.

Seguidamente, en las líneas 94 y 95, se sitúan los valores negativos del 0 al 2047 y los positivos del 2048 al 4095, así no se pierde ninguna parte de la senoide. No es el caso de las corrientes, ya que son medidas directas aportadas por el transformador de corriente.

Cerrando el condicional, se reserva el espacio correspondiente al número de bucle for dentro del vector epr[i], el valor de diez veces la velocidad calculada y es redondeado su valor mediante la función *round.*

Cabe resaltar las partes comentadas del código, referentes a endus y el *serial monitor*, se utilizan para mostrar en la consola de Arduino IDE la depuración para saber cuánto tiempo emplean los cálculos.

3.1.6.3.4. Tratado de información

Una vez hechas las medidas de la máquina de inducción, a través de las expresiones resaltadas en el apartado 2., se tratarán los fasores espaciales como parte real y parte compleja. Esto se debe a que Arduino no permite el cálculo con vectores espaciales. En consecuencia se expondrán constantes y variables globales, que actuarán durante toda la ejecución del programa, y otras locales, para actuar en el método *calculations.*

```
float Ure, Uim, ire, iim, Pabs, T, Put, Rend, Urms, Irms;
float urmss[SAMPLECOUNT],irmss[SAMPLECOUNT],pabss[SAMPLECOUNT],Ts[SAMPLECOUNT],rends[SAMPLECOUNT];
float urmsm=0, irmsm=0, nm=0, pabsm=0, Tm=0, rendm=0;
float urevec[SAMPLECOUNT], uimvec[SAMPLECOUNT], irevec[SAMPLECOUNT], iimvec[SAMPLECOUNT];

void calculations(){
  float nch0an, nch1an, nch2an, nch3an, is;
  const float VM0 = 2046.45, VM1 = 2045.19, VM2 = 1882.23, VM3 = 1873.3, pi = 3.14159265358979, FE0 = 0.20678279;
  const float FE1 = 0.20678279, FE2 = 0.00576808, FE3 = 0.00575158, expre = -0.5, exp120im = 0.8660254038;
  const float exp240im = -0.8660254038, Rest = 2.8666667;
  float  Ere, Eim, Psire, Psiim;

    for(short i =0; i<samples;i++){

      nch0an = ((float)ch0[i]-VM0)*FE0;
      nch1an = ((float)ch1[i]-VM1)*FE1;
      nch2an = ((float)ch2[i]-VM2)*FE2;
      nch3an = ((float)ch3[i]-VM3)*FE3;

      is = -nch2an - nch3an;
      Ure = (2*nch0an + nch1an)/3;
      Uim = nch1an/sqrt(3);
      ire = 2*(nch2an + is*expre + nch3an*expre)/3;
      iim = 2*(is*exp120im + nch3an*exp240im)/3;
      Pabs = -3*(Ure*ire + Uim*iim)/2;
      T =  3*(-((Uim + Rest*iim)/(100*pi))*iim - ((Ure + Rest*ire)/(100*pi))*ire);
      Put = (T-1.067663)*2*pi*epr[i]/600;
      Rend = Put/Pabs;
      Urms = sqrt(3)*sqrt(Ure*Ure+Uim*Uim)/sqrt(2);
      Irms = sqrt(ire*ire+iim*iim)/sqrt(2);
      urmss[i] = Urms;
      irmss[i] = Irms;
      Ts[i] = T;
      pabss[i] = Pabs;
      rends[i] = Rend;
      urevec[i]=Ure;
      uimvec[i]=Uim;
      irevec[i]=ire;
      iimvec[i]=iim;

    }

    urmsm = 0;
    irmsm = 0;
    nm = 0;
    pabsm = 0;
    Tm = 0;
    rendm = 0;

    for(short i =0; i<samples;i++){
      urmsm += urmss[i];
      irmsm += irmss[i];
      nm += epr[i];
      pabsm += pabss[i];
      Tm += Ts[i];
      rendm += rends[i];
    }

      urmsm /= 100.0;
      irmsm /= 100.0;
      nm /= 100.0;
      pabsm /= 100.0;
      Tm /= 100.0;
      rendm /= 100.0;
}
```

Fig. 41: Método calculations. Fuente: Alejandro RA.

Estableciendo unas directrices generales, es sencillo comprender la nomenclatura de las variables globales;

- Xre: Referida a la parte real de la variable compleja a la que representa.
- Xim: Referida a la parte imaginaria de la variable compleja a la que representa.
- Xrms: Realizada la raíz de la media cuadrática, representa el valor eficaz de la variable que representa.
- Xs: Permitirá guardar en un vector, los valores instantáneos de cálculo.
- Xm: Almacenará el valor promedio de las cien muestras captadas en el método *capture,* almacenadas en los vectores Xs[i].

- Xvec: Guarda en un vector, las variables que representa.

En cuanto a las locales del método *calculations;*

- nchXan: Será el valor analógico de la información recibida por el canal X.
- VMX: Como se trató durante la subsección 3.1.5., los canales tienen un valor de cero por defecto.
- FEX: Corresponde al factor de escala calculado en el apartado 3.1.5.

Finalmente, se declaran las constantes matemáticas que se emplearán, pues Arduino no las incluye de manera nativa, éstas son pi y e^120j, también e^240j. La constante constructiva de la máquina asíncrona sería la resistencia del estator, 8’6 en las tres fases.

Las operaciones se encuentran anidadas en un bucle *for*, de longitud equivalente al número de muestras, cien. Para cada nchXan, se le resta el VMX y se le aplica el FEX para realizar una conversión DA. Previamente, se le realiza un *casting* a la variable correspondiente al canal y de índice marcado por el bucle para pasarla a un tipo de dato de coma flotante.

Las líneas 125 a 135 nacen de las expresiones albergadas en el subcapítulo 2. Una vez realizadas las operaciones con los valores instantáneos, se almacenan en sus respectivos Xs y Xvec para aplicarles las transformaciones correspondientes.

Tras completar el cálculo de las cien cifras mediante el bucle, se inicializan las variables Xm con un valor de cero, ya que si no, almacenarán un valor aleatorio, que emponzoñaría el dato.

Las líneas 155 a 162 corresponden a un bucle de idénticas características al iniciado en 118, para el sumatorio de los valores reservados en los vectores Xs. Una vez finalizado el bucle, se presenta un cociente para el cómputo del valor medio, esto es, dividiendo entre el número de muestras.

Con esto, se da por concluido el método de cálculo, que da paso al de envío de datos

3.1.6.3.5. Envío de datos

Recuperando los valores promedios calculados, se realizará el método *report*, que adquiere dichos valores, les realiza una conversión AD y los envía por Blueetooth a la aplicación Android.

Para una correcta conversión, es necesario inicializar las variables Xd, equivalentes del valor en digital con un valor nulo. A continuación, se incluye la llamada al método de cálculo para admitir los valores promedio ahí calculados para obtenerlos en el método *report*. A cada variable se le aplica un factor de reducción del 10%, para asegurar una posible medida de hasta el 10% mayor a la nominal. Y cada variable es convertida según su valor nominal en unidades de ingeniería.

En concordancia con la aplicación, el ESP32 indica a la aplicación que se ha realizado el muestreo por parte del SoC y que se dispone a enviar el resto de datos a la orden de la aplicación. Para enviar esta baliza de presentación, recurriendo a la línea 183.

A continuación se envían cálculos de tensión, corriente, velocidad, par, potencia absorbida y rendimiento en digital.

```
172 void report(){
173   int Ud=0, Id=0, nd=0, Td=0, Pd=0, Rendd=0;
174 
175       calculations();
176       Ud = 4096*urmsm/230.0/1.1;
177       Id = 4096*irmsm/4.4/1.1;
178       nd = 4096*nm/10.0/1500.0/1.1;
179       Td = 4096*Tm/7.1/1.1;
180       Pd = 4096*pabsm/1260.0/1.1;
181       Rendd = 4096*rendm;
```

Fig. 42: Reporte de datos digitales Arduino. Fuente: Alejandro RA.

3.1.6.3.6. Loop

Hasta ahora, todos los métodos expuestos, no tienen implementación hasta que no aparece el bucle infinito. Éste se encarga de que, tras una orden del dispositivo Android, ejecute el programa.

```
43 int incoming;
44 
45 void loop() {
46 
47   if (ESP_BT.available()){
48 
49     incoming = ESP_BT.read(); /
50     Serial.print("Received:"); Serial.println(incoming);
51 
52     if (incoming == 51){
53 
54         capture();
55         ESP_BT.println("sampling finished");
56         report();//Llena el report
57         }    }
58   delay(20);
59 }
```

Fig. 43: Loop Arduino. Fuente: Alejandro RA.

Para poder recibir dicha señal, es declarada la variable *incoming*. Si el ESP32 tiene disponible la radio Bluetooth, éste se pone a la escucha de la radio mediante el método *read().* Mediante el puerto serie del monitor, imprime las palabras *Received* y el valor de la variable *incoming* para poder hacer debugging del contenido de la comunicación. Si la señal escuchada por el SoC es un 51, equivalente ASCII a un 3, se llama al método de captura. A continuación, se informa a la aplicación que el muestreo ha finalizado. Para concluir, una llamada al método de envío de datos por Bluetooth realizará su función programada.

El bucle se volverá a ejecutar tras una espera de 20milisegundos.

3.2. Módulo de interfaz de usuario

3.2.1. Justificación de la elección del tipo de dispositivo

En la actualidad, la capacidad de portar un dispositivo con sistema operativo Android está al alcance de una gran mayoría de la población mundial. Son dispositivos que parten de precios reducidos y con una alta compatibilidad con protocolos de comunicación implantados en dispositivos de código libre. Entre estos protocolos se encuentra Bluetooth, compatible con el ESP32, como se ha visto con anterioridad.

Además la facilidad para compartir la aplicación en la tienda de aplicaciones nativa de los dispositivos, Google Play, juega a favor en la universalidad de este tipo de aplicaciones destinadas a la toma de medidas *in situ* de una máquina de inducción.

Como último punto a favor, cabe comentar que el departamento ya disponía de dispositivos Tablet para algunas prácticas de asignaturas cursadas durante el grado. Por tanto se podía aprovechar el material para desarrollar esta nueva metódica.

3.2.2. Programación

3.2.2.1. Lenguaje de programación

Android Inc. fue una empresa formada por cuatro socios en 2003 con la intención de desarrollar un sistema operativo avanzado para cámaras digitales, que se pudiesen conectar a dispositivos con pantalla para mostrarlas de manera inalámbrica. Poco después se redigiría el foco para que germinasen “dispositivos móviles que estén al corriente de la ubicación y preferencias del usuario”.

Dos años después de la fundación de la empresa, Google invertiría la cuantía de 50M de euros, algo sin precedentes para un proyecto tan joven. Con el objetivo de inyectar en el mercado una alternativa viable al ya imperante IOS, Symbian y Windows Mobile. El primer terminal con la tecnología emergente data de septiembre de 2008. En 2010, la cuota de mercado de este sistema operativo sería del 23’3% sobre 305 millones de dispositivos frente al monopolio absoluto del 86’1% sobre 1.372 millones de terminales en 2019. (Mena Roa, 2020)

El año anterior al lanzamiento del sistema operativo, se creó la Open Handset Alliance, una adhesión de 34 empresas de telecomunicaciones, hardware y software del momento, actualmente pertenecen 84. El objetivo de la OHA es la potenciación de estándares abiertos para dispositivos móviles. La estandarización permite la familiarización del usuario con un concepto global, no adscrito a una marca en concreto. Así como la operatividad y comunicación interdispositivo referida en el punto anterior. Por ello, Google dispuso Android en el concepto ‘open source’, bajo la conocida como licencia Apache. De este modo, se presenta la posibilidad de todo un ejército de programadores en potencia, ya que existen foros, webs y clubs con un elevado tráfico de datos para la cooperación en el desarrollo de aplicaciones así como búsqueda de bugs, fallos de seguridad y demás. Uno de ellos es GitHub o Stack Overflow, a los que se ha recurrido durante la programación de la aplicación IMIEScope.

El entorno de programación que pone a disposición del público Google se conoce como Android Studio y es un IDE. Proporciona soluciones integradas en el propio software para facilitar la programación de las aplicaciones en mayor medida que Arduino IDE. A parte incluye emulador para virtualizar dispositivos, si las especificaciones del ordenador utilizado para la programación lo permiten. En el presente caso, se han hecho todos los cambios directamente en el terminal físico.

Se puede descargar desde: https://developer.android.com/studio

Android Studio permite la programación en distintos lenguajes; C++, Java o Kotlin. También posee un traductor de Java a Kotlin, al ser este un sublenguaje. Para la aplicación desarrollada se ha optado por Java, que posee una sintaxis derivada de C y C++, lenguajes estudiados durante el Grado de ITI.

Java es un lenguaje de programación orientado a objetos, es decir, a contenedores de código y datos, es decir, compartimentar lo variable de lo invariable. De este modo, no es menester cambiar el código cada vez que se desea cambiar los datos. Para desarrollar un problema de gran complejidad se opta por el desarrollo por separado de los subproblemas que se derivan del mismo, para luego ensamblarlos en el producto final. Además, como beneficio pasivo, se obtienen los objetos para luego poder reutilizarlos en diferentes áreas del código sin necesidad de volver a programarlos o incluso en otros problemas futuros. Cabe subrayar la maleabilidad de los objetos. Por otro lado brinda la capacidad de ejecutar los programas independientemente del hardware. Para utilizar Android Studio, previamente es menester instalar el Java Development Kit, o JDK. Es toda la documentación y herramientas que necesita para poder operar correctamente con Java.

Se descarga desde: https://www.oracle.com/java/technologies/javase-jdk16-downloads.html

Por tanto el entorno de Android Studio consta de:

- Intérprete: Ejecutor de los archivos java
- Compilador
- Visualizador de applets: Desde donde se puede programar, ya sea por código o por widgets, la interfaz gráfica de la aplicación. Permite la visualización en tiempo real de los cambios sucedidos en el entorno gráfico.
- Depurador: Es una herramienta muy potente para la localización de fallos de programación, errores de sintaxis, etc. Suele presentar una alternativa a errores sencillos en tiempo real.
- Generador de documentación

3.2.2.2. Estructura de la aplicación Android

La estructura de una aplicación Android presenta una gran complejidad en el funcionamiento por el elevado número de componentes que han de trabajar sincronizados. Por suerte, Android Studio se encarga de la mayoría de los trabajos en el plano de la compatibilidad de cada una de las partes de la aplicación. De este modo, el usuario no ha de preocuparse de la compatibilidad de estándares, como pueda ser el Bluetooth o el Wifi, este es el caso de estudio, ya que como se verá en puntos posteriores, se hará uso librerías o bibliotecas informáticas.

Como consecuencia de la ayuda prestada por Android Studio, es posible distinguir dos grandes componentes de la aplicación objeto de este apartado, el código lógico y la parte gráfica. Se tratarán por separado haciendo constar durante el desarrollo del código lógico la unión entre las dos partes.

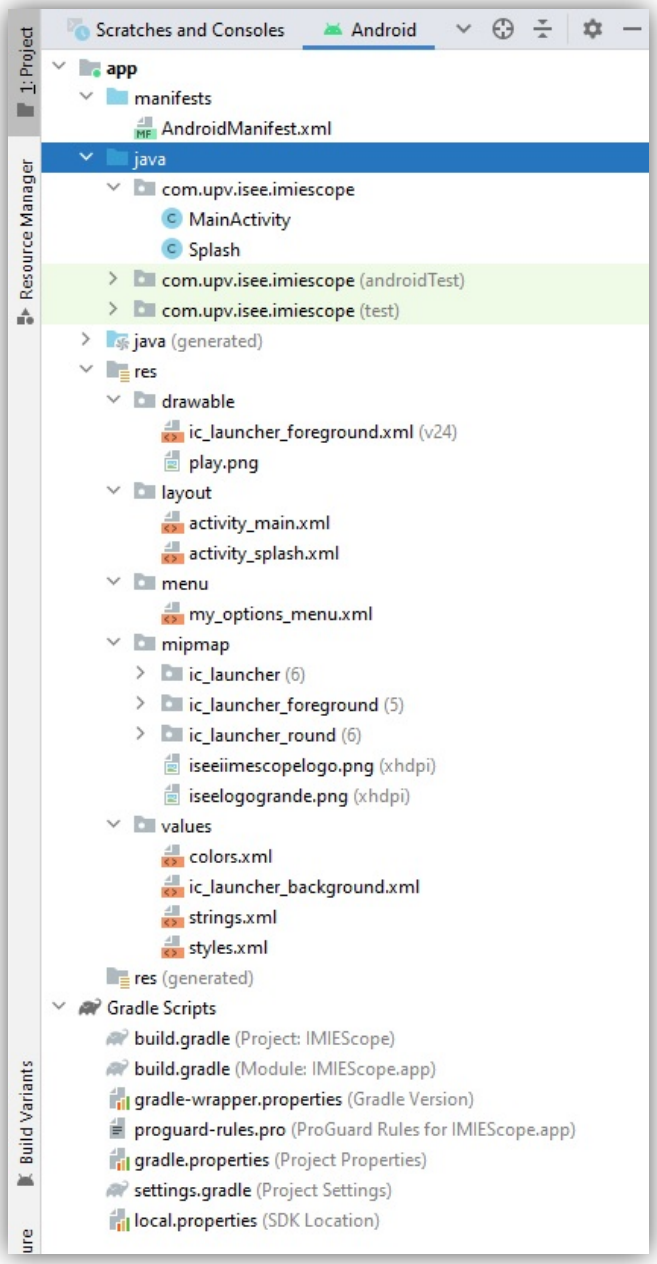


Fig. 44: Árbol de la aplicación. Fuente: Alejandro RA.

En la figura 44 se encuentra la estructura que posee la app Android. Ésta consta de los archivos que conforman la app en sí y luego de las herramientas de compatibilidad. Numerosas referencias se harán a él a la hora de identificar la localización de archivos y programas.

3.2.2.2.1. Código lógico

Este apartado de la aplicación responde a todo el protocolo de comunicación, cálculo, enlace y emisión de datos por pantalla. Comporta el 80% del tiempo empleado en el desarrollo de la misma. Debido a la longitud del código, más de 500 líneas y atendiendo al esqueleto habitual de un programa, se divide el cifrado en apartados. Por otro lado, se incluirán figuras con extracciones del código para su explicación. El código completo se encontrará en los anexos.

Esta parte está escrita en Java y su parte primordial es la *Main activity*, o actividad principal. Frecuentemente las aplicaciones Android poseen diversas actividades, es decir, programas completos que desarrollan una parte de esta. En este caso sólo existen dos, la *Main activity* y *Splash*, localizadas en la subcarpeta com.upv.isee.imiescope.

3.2.2.2.1.1. Actividades

Las actividades se ordenan en el apartado *AndroidManifest*, que permite indicar el orden de dichas actividades mediante un *intent filter.* Un *intent* es un mensajero para llamar a la acción a un elemento de la aplicación. Para este caso, permite iniciar la *Splash activity* como primera actividad a ejecutar por la aplicación, seguida por la *Main activity*.

```
24          <activity android:name=".Splash"
25              android:label="@string/app_name"
26              android:theme="@style/Theme.AppCompat.Light.NoActionBar"
27              >
28              <intent-filter>
29                  <action android:name="android.intent.action.MAIN" />
30
31                  <category android:name="android.intent.category.LAUNCHER" />
32              </intent-filter>
33          </activity>
34          <activity android:name=".MainActivity">
35          </activity>
```

Fig. 45: Android Manifest Intent Filter. Fuente: Alejandro RA.

La *Splash activity* es un programa sencillo que hace aparecer, durante tres segundos, el logo del equipo de investigación para el cual se ha desarrollado la aplicación al iniciarse la misma. La actividad se yergue sobre el código de la figura 46.

```
 8 public class Splash extends AppCompatActivity {
 9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_splash);
14
15         new Handler().postDelayed(new Runnable() {
16             @Override
17             public void run() {
18                 Intent intent = new Intent(Splash.this,MainActivity.class);
19                 startActivity(intent);
20                 finish();
21             }},3000); }}
```

Fig. 46: Splash screen activity. Fuente: Alejandro RA.

La unión lógico-gráfica se localiza en la línea 13 de la figura 46 mediante el método *setContentView*, que como argumento recibe la clase R, poseedora de todos los recursos de la aplicación, dentro de la que está *layout*, para definir que es un objeto de tipo gráfico y en sí el código .xml que se tratará en el punto 3.2.2.2.2.

Como se observa en la línea 18 de la figura 45, existe un salto desde la actividad *Splash* a la *MainActivity.class* mediante el método *Intent* con nombre *intent*. A renglón seguido, empieza la acción ejecutada por *intent* y termina, cosa que dirigirá la pantalla a la actividad principal con un retraso de 3000ms.

Class es una palabra reservada de Java para definir un objeto de tipo *class*, es decir, un programa. Se puede apreciar también que aparece la palabra reservada *this*, es utilizada para reclamar variables localizadas en otras clases e indicar que se le hará un trato con el propio nombre que tiene en la clase con la que se interactúa. De este modo, no se permite la existencia de ambivalencias entre variables de la clase de la que se extrae dicha variable y la clase que opera con ésta si existiese una variable de igual nombre.

3.2.2.2.1.2. Librerías

Para el caso de la aplicación de Android, se dispondrán numerosas librerías, pero para optimizar, es cuestión explicar las que tienen una funcionalidad clave en la misma.

Se importan desde la actividad en la que se van a aprovechar.

```
 3 import android.bluetooth.BluetoothAdapter;
 4 import android.bluetooth.BluetoothDevice;
 5 import android.bluetooth.BluetoothSocket;
22 import org.achartengine.ChartFactory;
23 import org.achartengine.GraphicalView;
24 import org.achartengine.model.XYMultipleSeriesDataset;
25 import org.achartengine.model.XYSeries;
26 import org.achartengine.renderer.XYMultipleSeriesRenderer;
27 import org.achartengine.renderer.XYSeriesRenderer;
36 import de.nitri.gauge.Gauge;
```

Fig. 47: Librerías Android Studio. Fuente: Alejandro RA.

La primera librería, de la cual se van a hacer uso en tres métodos, va a permitir la comunicación con el ESP32.

Por otro lado, se dispone la *achartengine* que se trata de una librería de gráficos varios, que consiente imprimir por pantalla los datos que se precisen, como cabrá explicar más adelante y como se puede intuir por el nombre de los métodos exportados, se hará uso de los gráficos XY. Se trata de una librería que se apoya en la parte lógica para su personalización, por lo que se le dedicará el apartado 3.2.2.2.2.

Finalmente se precisa de la librería *nitri.gauge,* encargada de plasmar los medidores de aguja sobre los que serán hechas modificaciones en el diseño de la interfaz gráfica.

Para que la aplicación cuente con las librerías *achartengine* y *nitri.gauge,* se han de apuntar en la sección *build.gradle(Project: IMIEScope)* y *build.gradle(Project: IMIEScope.app)* para incluir las líneas destacadas en azul en la figura 48. Al no ser librerías muy utilizadas globalmente, conviene asegurarse que Android Studio sepa tratar las compatibilidades y sincronizar objetos en la app.

MainActivity.java | activity_main.xml | build.gradle (:app) | build.gradle (IMIEScope) | strings.xml

You can use the Project Structure dialog to view and edit your project configuration — Open (Ctrl+Alt+Mayús+S) — Hide notification

```
// Top-level build file where you can add configuration options common to all sub-projects/m
buildscript {

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:4.2.0'


        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()

        maven{ url 'https://www.jitpack.io' }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

MainActivity.java | activity_main.xml | build.gradle (:app) | build.gradle (IMIEScope) | strin — Attach Debugger to Android Process

You can use the Project Structure dialog to view and edit your project configuration

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 30
    defaultConfig {
        applicationId "com.upv.isee.imiescope"
        minSdkVersion 16
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    //noinspection GradleCompatible
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'


    implementation files('libs/achartengine-1.1.0.jar')
    implementation 'com.github.Pygmalion69:Gauge:1.5.2'
}
```

Fig. 48: Modificación gradle para librerías. Fuente: Alejandro RA.

3.2.2.2.1.3. Conexión vía Bluetooth con el ESP32 y recepción de datos

Como se anunció en el apartado 3.1.2., el ESP32 se conecta vía Bluetooth a la Tablet de la que dispone el departamento, para ello, Android Studio dispone de una librería ampliamente utilizada. De esta librería, se importarán simplemente tres métodos, sobre los que serán tratados muy someramente, ya que la descripción del protocolo de comunicación Bluetooth es materia poco útil para el TFG. Sin embargo, para una comprensión básica del protocolo Bluetooth de la aplicación, conviene definir diversos conceptos básicos.

- *Socket* o conector: Permite enviar o recibir datos ininterrumpidamente mediante *threads.* Es el equivalente a un bus o canal.
- *Threads* o hilos: Los hilos permiten la comunicación, son cadenas de datos con información. En conexiones de tipo Bluetooth existe el hilo de servidor, cliente y de conexión.

El Bluetooth es un protocolo de comunicación inalámbrica mediante ondas de radio en banda de 2.4GHz. Es el responsable de la comunicación, como se ha visto, del microcontrolador y la Tablet. El proceso de comunicación con la app requiere que los dispositivos se encuentren emparejados previamente. Para esto, la ruta a seguir en la Tablet a Ajustes>Bluetooth>Buscar dispositivos. En ese momento, la Tablet empezará a buscar la emisión del ESP32. Tras pulsar sobre el nombre del microcontrolador, MaqAsinc_001, se sincronizará para su posterior uso por parte de la app, con la que se conectará automáticamente.

Esa automatización está programada en la app de modo que la Tablet es el cliente y MaqAsinc_001, el servidor. La comunicación tiene el siguiente diagrama de flujo;

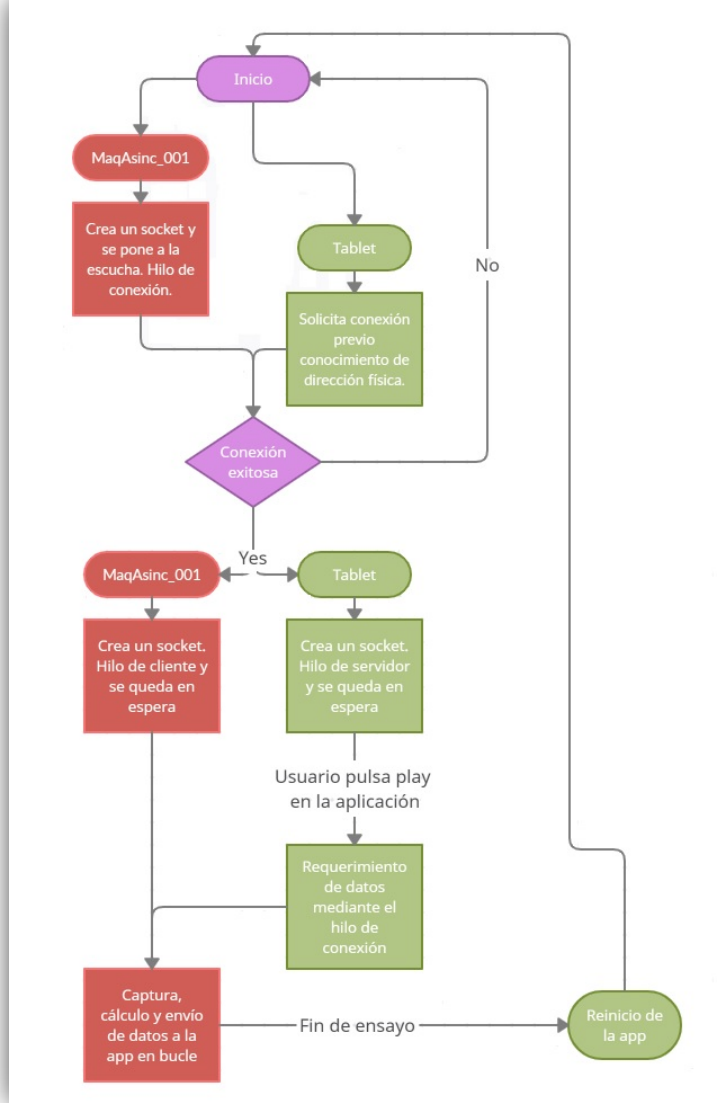


Fig. 49: Diagrama de flujo conexión Bluetooth. Fuente: Alejandro RA.

Cuando la app recibe los datos hace un nuevo requerimiento de datos, los recibe, trata y grafica.

Para poder realizar todas estas tareas previas a la ejecución y envío de datos, se importaron las librerías comentadas en el punto anterior. En concreto la primera permitirá establecer el adaptador, responsable del control de la radio de la app. Permite saber si se tiene acceso y disponibilidad del Bluetooth. *Device* y *socket* permitirán distinguir dispositivos sincronizados y establecer el socket por parte de la app.

Finalmente cabría destacar que el hilo de conexión será el de tráfico de datos, que habrá que tratar mediante un *handler*, o manipulador de hilos.

Como se ha comentado, la conexión, al ser un protocolo estandarizado, tiene todos los métodos a implementar ya integrados en Android Studio, lo que permite la autocompleción de éstos, y dejan de ser reseñables para el desarrollo del TFG.

3.2.2.2.1.4. Tratado de datos

En el presente subcapítulo, se detallarán la recepción de los datos enviados por el ESP32. Dentro de un *handler*, se encuentran las instrucciones de lectura de las líneas de datos enviados por Bluetooth. Un *handler* es un manipulador de *threads* o hilos secuenciales de instrucciones. El *handler* permite enviar y procesar objetos asociados a la cola de mensajes de un hilo y comunicar dos o más subprocesos, ciñéndose estrictamente a la asignación de los datos recibidos a variables de la aplicación.

De ahora en adelante, los procesos en bucle serán representados como el primero y el último, dando a entender que los tres siguientes al primero siguen su misma estructura pero con la variable correspondiente. Debido a la extensión del código, se mostrarán saltos de línea denotados con una línea gruesa negra en la columna del número de línea. Como consecuencia de dichos saltos, podrían no aparecer cierres de llaves que para la programación son clave pero para el objeto de estas secciones de la memoria, que son la explicación del código, no lo son.

Como se trató en el apartado 3.1.6.3.5., los datos emitidos por el ESP32 tienen la siguiente estructura; Ud; Id; nd; Td; Pd; Rendd. Es preciso comentar la aparición de *sampling finished*, que será un punto importante a continuación.

Fig. 50: Muestra de datos enviados por Bluetooth. Fuente: Alejandro RA

```
204         mHandler = new Handler(){
205             public void handleMessage(Message msg){
206                 if(msg.what == MESSAGE_READ){
207                     String readMessage = null;
208                     try {
209                         //Lectura del mensaje del esp32, codificación UTF-8
210                         readMessage = new String((byte[]) msg.obj,0,msg.arg1,"UTF-8");
211                     } catch (UnsupportedEncodingException e) {
212                         e.printStackTrace();
213                     }
214                     int last = readMessage.indexOf("\r\n");
215                     String header = readMessage.substring(0,last);
216                     String remains = readMessage.substring(last+2);
217 
218                     if(header.equals("sampling finished")){
219 
220                         boolean scontinue = false;
223                         do{
224                             int nextlf = remains.indexOf("\r\n");
225                             String line = remains.substring(0,nextlf);
226                             remains = remains.substring(nextlf+2);
227                             scontinue = (remains.indexOf("\r\n")>0);
228 
229                             if(line.contains("Relay"))
230                                 break;
231 
232                             line = line.substring(1);
233                             int nextsc = line.indexOf(";");
234 
235                             //Recibimos y almacenamos hasta 100 valores
236                             datarec = datarec +1;
237                             if(datarec >100)
238                                 datasto = 100;
239                             else
240                                 datasto = datarec;
241 
242                              //Leemos valor a valor lo enviado por el esp32
243                             double timems =0.0;
244                             double ch0m = Double.valueOf(line.substring(0,nextsc));
245                             line = line.substring(nextsc+1);
246                             nextsc = line.indexOf(";");
 |
254                             double ch5m = Double.valueOf(line);
 |
307                         }while(scontinue);
```

Fig. 51: Lectura datos app. Fuente: Alejandro RA.

Como inicio es declarada la variable *readMessage*, encargada de recibir en codificación UTF-8 los datos de la figura 50, mediante una simple estructura de separación de cadenas, es posible obviar el 3, ya que está comentado en el punto 3.2.2.2.1.3 mediante el comando de la línea 214 para colocar el punto de lectura en el inicio de la siguiente línea. Para continuar y asignar el presente *header* y los *remains*, estas dos variables representarán el inicio de tratado de datos y los datos en sí.

El *header,* cuando sea equivalente a *sampling finished* se inicia la estructura condicional de la línea 218, continente de la lectura del código. Se declara una variable que será responsable de mantener el bucle de tipo *do while* en uso, llamada *scontinue*. En dicho bucle se declara la variable *nextlf*, encargada de situar la lectura en el principio y en la siguiente línea, como el carro porta cabezal de una máquina de escribir. Junto con la asignación de la variable *line* como línea de datos de los que se realizará la extracción sus valores y la *remains* como resto de datos.

Por tanto, los *remains* están en cola para su tratado. Como última herramienta de lectura, es definida *nextsc*, como indicador de dónde están el punto y coma que separa dentro de la línea un valor de otro.

Para una correcta implementación de la aplicación, se ha optado por trabajar con cien líneas de valores, ya que de este modo, se puede realizar un histórico impreso por pantalla como se tratará más adelante. Debido a esta característica, se ponen en uso las variables *datarec* para los datos recibidos y *datasto* para los datos guardados. Huelga decir que van a servir únicamente como indicadores de posición dentro de vectores. Estas variables son globales de la aplicación, por lo que no se ve la declaración en esta sección.

Culminando la lectura, entran las líneas 244 a 254, en las que se crean las variables de tipo *double*, en coma flotante, de *ch0m* a *ch5m*. Asigna el valor numérico en formato digital y salta al siguiente valor del punto y coma para repetir el proceso con las seis variables.

El problema de representación de datos en formato vectorial mediante *XYSeries* de la librería *achartengine* será resuelto en siguientes apartados.

3.2.2.2.2. Diseño de la interfaz gráfica

Hasta el momento se ha hecho una exposición de la metodología seguida para el tratado de la información, pero la aplicación posee una parte gráfica cuidadosamente diseñada para hacerla atractiva, intuitiva y útil a la hora de tener un orden de magnitud de las medidas representadas o indicadas numéricamente.

Android Studio pone a disposición del programador una potente herramienta de diseño. Para acceder a ella, siguiendo la ruta a la subcarpeta *layout*, como se observa en la figura 44, en la que están alojados dos archivos de tipo .xml, uno dedicado al *Splash screen* y otra para la *Main activity*.

XML no es exactamente un lenguaje de programación, sino un arquitecto de lenguajes de marcado, que permite crear etiquetas y estructurar datos para almacenar y representar. Posee una sintaxis concreta que no es necesario conocer debido a la funcionalidad diseño de Android Studio. Ésta permite codificar la parte gráfica mediante código .xml o directamente mediante objetos visuales. También permite cambiar sus atributos para su personalización rápida y eficazmente.

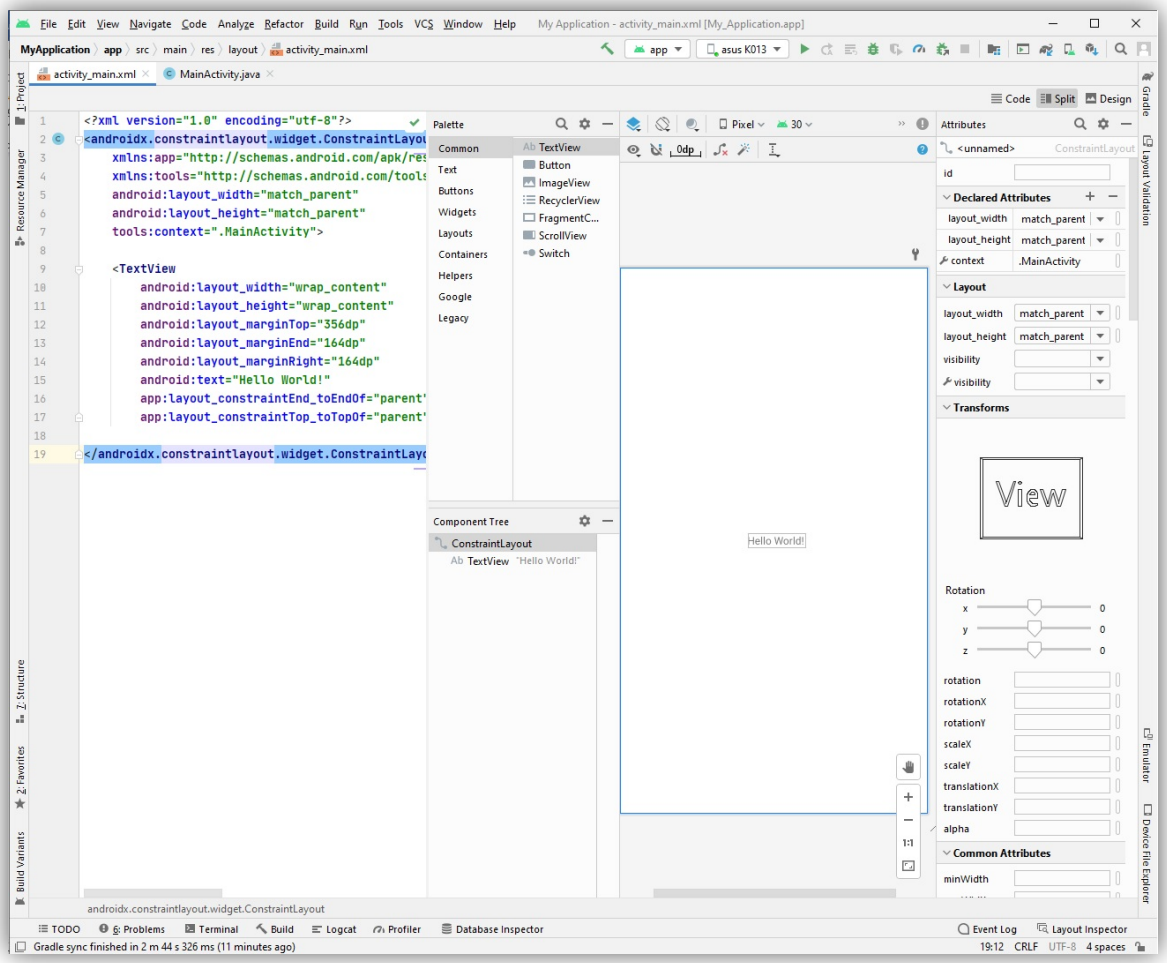


Fig. 52: Entorno diseño gráfico. Fuente: Alejandro RA.

Como se aprecia en la figura 52, se tiene la parte izquierda, en lenguaje XML, en el centro superior la paleta de objetos, en el centro inferior el árbol de objetos. Se observa la pantalla del dispositivo que se escoja, ya que Android Studio posee una biblioteca de dispositivos virtuales con sus resoluciones y características para correr el emulador. Y finalmente la pestaña modificadora de atributos a la derecha.

3.2.2.2.2.1. Diseño

El diseño de la interfaz de la aplicación es el mostrado a continuación, del que destacan tres elementos, la gráfica, los indicadores de aguja y las medidas en unidades de ingeniería. Diseccionando los tres elementos en tres subíndices diferentes en los que se detallará el código obrado para su programación en cada caso. Se puede resaltar el uso de un esquema de colores determinado para cada una de las magnitudes.

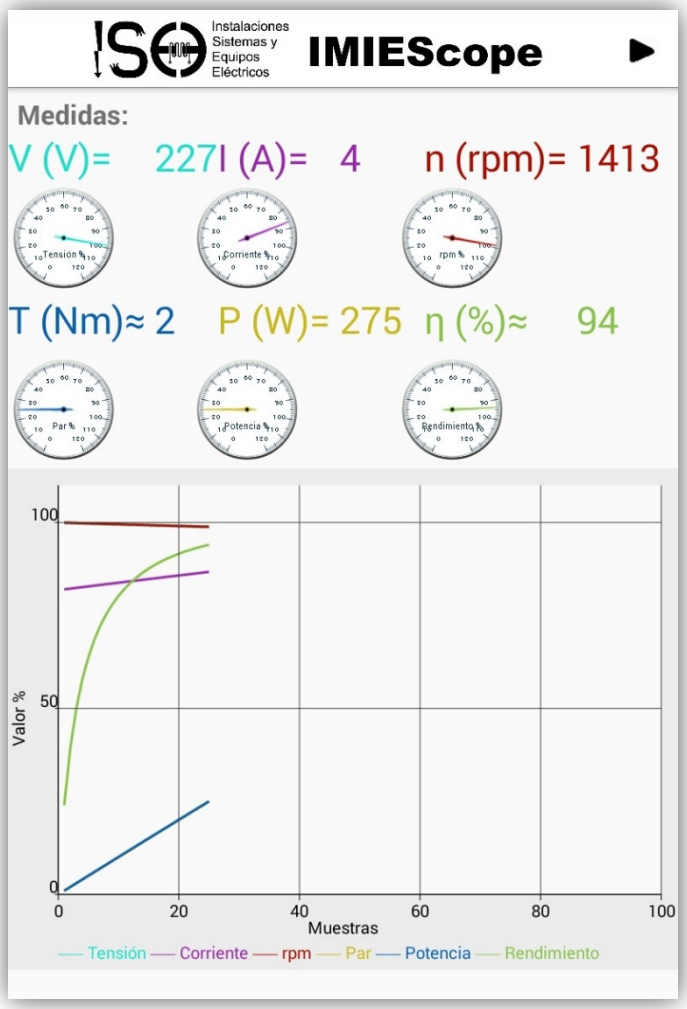


Fig. 53: Interfaz de la aplicación. Fuente: Alejandro RA.

Desde el punto de vista de la programación, es necesaria una estructura como se ve en la figura 44, con el árbol de componentes y la edición dividida del plano y vista del usuario. En el plano serán encargadas de todas las restricciones dimensionales. La vista de usuario permitirá ver a tiempo real cada uno de los cambios de la presente app, actuará de prototipo. Finalmente el árbol brinda la oportunidad de tener clara la conformación vertical y horizontal de los objetos.

Dicha disposición se consigue mediante un objeto nativo, de tipo *tableLayout*, que es una gradilla bidimensional para rellenar con objetos de tipo *TableRow*. Éstos son, a su vez, gradillas rectas sobre las que se dispondrán los objetos de tipo *gauge* y *TextView.* La vista del negativo posibilita ver los elementos *tableLayout* y *TableRow* con los recuadros en los que descansan los indicadores de aguja. En sucesivos subcapítulos serán tratadas las relaciones lógico-gráficas de cada elemento.

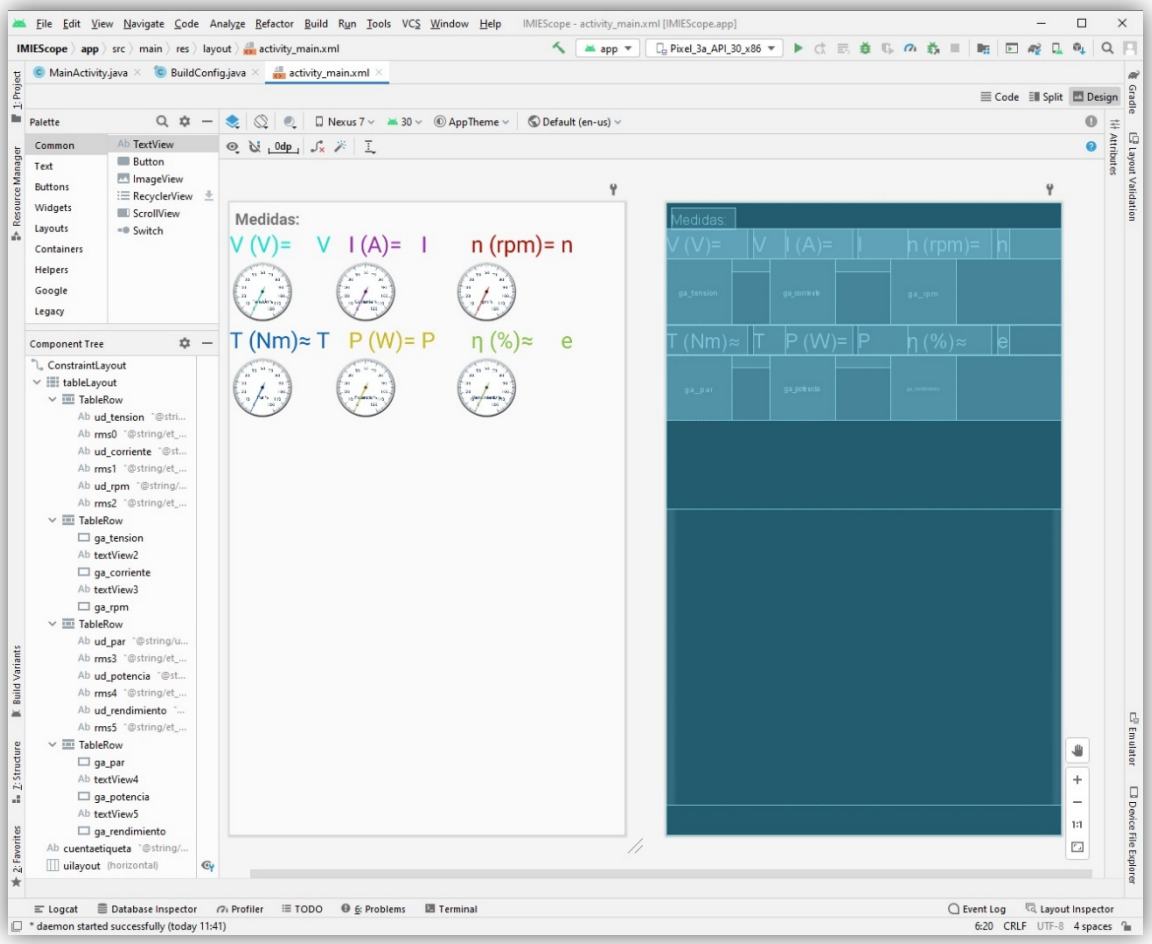


Fig. 54: Vista de diseño y cianotipo. Fuente: Alejandro RA.

Se observa una clara diferenciación de zonas, la de las medidas, indicado mediante un elemento de tipo *TextView*, un cuadro diseñado para mostrar datos por pantalla, llamado 'Medidas'. Seguido de los datos numéricos y los indicadores de aguja. Culmina en la parte inferior el gráfico de históricos, con el que se empieza la descripción en profundidad.

3.2.2.2.2.1.1. Gráfica de históricos

En primera instancia, se han de definir los valores nominales de la máquina asíncrona, presentes en la figura 55. Éstos están en unidades de ingeniería, esto es, voltios, amperios, revoluciones por minuto, Newton·metro y vatios.

```
51      private double Un = 230.0;
52      private double In = 4.4;
53      // private double I0 = 3.6;
54      private double n0 = 1500.0;
55      // private double Tn = 6.07;
56      private double Tn = 7.1;
57      private double Pn = 1260.0;
```

Fig. 55: Valores nominales. Fuente: Alejandro RA.

El conveniente apuntar que la potencia nominal de la máquina es 1’1kW, pero no tiene sentido representar el rendimiento en base a la potencia nominal sino a la consumida en régimen permanente, de ahí el valor de Pn.

Para localizar en la interfaz la gráfica, se necesita llamar a la función *findViewById*, ésta se encarga de buscar y enlazar con la variable que se asigna al objeto gráfico. En este caso, para al actual código, la gráfica se llamará *layout*, y es un objeto de tipo *LinearLayout*. A la variable se le aplicarán transformaciones y restricciones para su representación acorde a las necesidades del TFG. A pesar de la utilidad del gráfico de históricos para observar la evolución de las medidas, posee un inconveniente, y es que el *setup* es específico para la resolución y uso de la Tablet que se dispone. Así que la parte de diseño, restricción de márgenes y coloración de estos no se incluirán.

```
153         //Unión lógica-gráfica de la gráfica
154         LinearLayout layout = (LinearLayout) findViewById(R.id.uilayout);
155         mChartView = ChartFactory.getLineChartView(this, mDataset, mRenderer);
156         layout.addView(mChartView, new LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,
    LinearLayout.LayoutParams.WRAP_CONTENT));
```

Fig. 57: Unión lógica-gráfica de la gráfica de históricos. Fuente: Alejandro RA.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     xmlns:gauge="http://schemas.android.com/apk/res-auto"
8     tools:context="com.upv.isee.imiescope.MainActivity">
273     <LinearLayout
274         android:id="@+id/uilayout"
275         android:layout_width="0dp"
276         android:layout_height="450dp"
277         android:layout_alignParentStart="true"
278         android:layout_alignParentLeft="true"
279         android:orientation="horizontal"
280         android:visibility="visible"
281         app:layout_constraintBottom_toBottomOf="parent"
282         app:layout_constraintEnd_toEndOf="parent"
283         app:layout_constraintStart_toStartOf="parent"
284         app:layout_constraintTop_toBottomOf="@id/cuentaetiqueta"
285         app:layout_constraintVertical_bias="0.904"
286         tools:visibility="visible" />
287
288
289 </android.support.constraint.ConstraintLayout>
```

Fig. 56: Código .XML del gráfico de históricos. Fuente: Alejandro RA.

El elemento *LinearLayout* que será el lienzo sobre el que graficar, ha sido denominado en la parte de .XML como uilayout. Así es como aparecerá en el código escrito para referenciar la parte lógica y sobre el elemento que se aplican los atributos.

Todo el código .XML está escrito y englobado en las ocho primeras líneas de código que incluye por defecto Android Studio a la hora de programar, encargadas de mostrar un lienzo sobre el que diseñar. A excepción de la séptima, que se tratará con posterioridad. Como se ha comentado con anterioridad, Android Studio posibilita modificar los atributos sin necesidad de escribir código, con la parte de diseño, ver figura 54.

La interfaz completa de la aplicación es de tipo *ConstraintLayout*, se ha diseñado bajo este tipo de restricciones se pueden desarrollar vistas complejas sin necesidad estructuras anidadas en la sección global como un *tableLayout*, que serán empleadas en el futuro en una sección local. Para el posicionamiento de cada uno de los elementos globales que se incluyen en la interfaz, basta con diferenciar el elemento *ConstraintLayout*, que será definido en las líneas 5-6, para referir a sus bordes cada uno de los elementos globales.

```
47      private XYMultipleSeriesDataset mDataset;
48      private XYMultipleSeriesRenderer mRenderer;
49      private XYSeries ch0s, ch1s, ch2s, ch3s, ch4s, ch5s;
59      private XYSeriesRenderer ch0sr, ch1sr, ch2sr, ch3sr, ch4sr, ch5sr;
60      private GraphicalView mChartView;

119         mDataset = new XYMultipleSeriesDataset();
120         mRenderer = new XYMultipleSeriesRenderer();
121
122         //Creación de los canales XY para el dataset  y en string para el renderer
123         ch0s = new XYSeries("Tensión");
124         mDataset.addSeries(ch0s);
125         ch0sr = new XYSeriesRenderer();
126         mRenderer.addSeriesRenderer(ch0sr);
148         ch5s = new XYSeries("Rendimiento");
149         mDataset.addSeries(ch5s);
150         ch5sr = new XYSeriesRenderer();
151         mRenderer.addSeriesRenderer(ch5sr);
188         ch0sr.setColor(Color.rgb(26, 227, 208));
193         ch5sr.setColor(Color.rgb(139,195,74));
196         ch0sr.setLineWidth(3);
201         ch5sr.setLineWidth(3);
```

Fig. 58: Declaración y puesta a punto del gráfico de históricos. Fuente: Alejandro RA.

Como variables locales, se declaran las líneas de 47 a 49, 59 y 60. Éstas responden al *dataset*, que son propiamente los datos que serán representados en el gráfico. El *renderer* es una herramienta que controla el aspecto de dicho gráfico así como los fragmentos del *dataset* que se grafican. A renglón seguido, se enuncian las variables que serán nominadoras de hasta 100 datos por canal reservado. El canal 0 responde a la tensión y así se guarda para dar nombre en el *dataset*. Pero dichas variables se activarán para su caracterización mediante el *renderer,* momento en el que son declaradas como ch0sr. Las líneas 188 a 201 son la señalización propia de cada canal, en la que se define un color, que acompañará a la magnitud representante durante todo el proceso de visualización por pantalla. Subrayando la necesidad del uso de grosor tres como solución de compromiso entre la precisión y la distinción de series representadas.

Como se verá más adelante, existe un archivo .XML que contiene un código de colores, que se pueden referenciar como se hace en otros lenguajes de programación. En este caso, no es posible utilizarlo en el método *setColor* de los elementos *XYSeriesRenderer*, ya que como argumento recibe un valor entero, que es posible circunvalar mediante el método *Color.rgb*. Éste método devuelve un valor entero codificado en tipo *RedGreenBlue,* ampliamente usado.

Como consecuencia del uso de la librería de gráficos, apoyada en gran medida en una parte lógica, el propósito de la siguiente extracción de código es la preparación de dichos datos para su correcto funcionamiento. Los datos del gráfico se representarán el tanto por cien de su valor nominal, ya que es un valor mucho más representativo a la hora de identificar, por parte del usuario, las condiciones de funcionamiento de la máquina. Además presenta la viabilidad de uso de manera universal con el cambio de los valores definidos como nominales. Esto se consigue mediante el siguiente código;

```
261                         //Paso de digital a analógico en valor porcentual
262                         double ch0man = (ch0m*1.1/4096)*100;
 |
267                         double ch5man = (ch5m*1.1/4096)*100;
```

Fig. 59: Conversión DA porcentual. Fuente: Alejandro RA.

Simplemente se realiza una conversión digital-analógica inversa a como se vio en Arduino, figura 42.

En última instancia es menester asegurarse que el tamaño de la gráfica albergue siempre cien valores por canal y bloquee el tamaño de la gráfica. Independientemente del tamaño de *datarec.* En las siguientes líneas, se elimina el primer valor, una vez se hayan recibido cien líneas de datos, de las variables chXs, sea X un valor entre 0 y 5. Y se dispone para el índice *datarec*, que cabe recordar que es un simple contador como apunta la figura 51, en la variable chXs, su valor porcentual analógico.

```
284                         //A cada valor superior a 100, eliminamos el primero para mantener 100 datos
285                         if(datarec > 100){
286                             ch0s.remove(0);
 |
291                             ch5s.remove(0);
292                             mRenderer.setXAxisMax(datarec);
293                             mRenderer.setXAxisMin(datarec-100);
294                         }else{
295                             mRenderer.setXAxisMax(100);
296                             mRenderer.setXAxisMin(0);
297                         }
298
299                         //Añadimos a los valores de XY el número de dato y su valor analógico
300                         ch0s.add(datarec,ch0man);
 |
305                         ch5s.add(datarec,ch5man);
```

Fig. 60: Gráfica de 100 valores y adición de valor a series XY. Fuente: Alejandro RA.

3.2.2.2.2.1.2. Indicadores de aguja porcentuales

Para la implementación de los indicadores de aguja, se ha recurrido a la librería *nitri.gauge,* una solución sencilla y elegante de fácil modificación desde *activity_main.xml* como se observa a continuación.

Para incluir tanto los objetos *TextView* como los de tipo *gauge*, optando por un *tableLayout* que contiene a elementos *TableRow*. De manera análoga a los canales en la sección de tratado de datos, será mostrada la configuración de un elemento, el resto se producen con similaridad.

Como muestra la figura 61, se dispone de un elemento de tipo *TextView*, que indicará la magnitud a representar, en este caso tensión. A su derecha, el *TextView* que anuncia el valor de esa medida y bajo el indicador de aguja, *gauge.* Una personalización de éstos en .XML dará paso a su relación con el código Java.

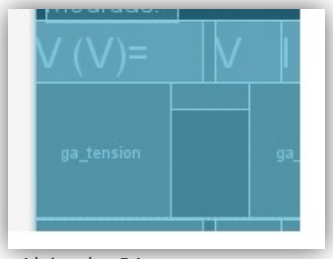


Fig. 61: Detalle TextView y gauge. Fuente: Alejandro RA.

Los indicadores van a ser un indicador porcentual del valor nominal de cada una de las medidas, se tomará como ejemplo la primera de todas las magnitudes, la tensión, para explicar la composición visual y la animación de la aguja con los datos reutilizados del apartado anterior. Al ser un objeto no incluido en la paleta de objetos de Android Studio, es de obligado cumplimiento programar cada indicador en líneas de XML, donde es posible modificar sus atributos. Para ejecutar exitosamente los indicadores, se recuerda incluir la línea 7 de la figura 51.

```
10      <TableLayout
11          android:id="@+id/tableLayout"
12          android:layout_width="match_parent"
13          android:layout_height="wrap_content"
14          android:layout_marginTop="40dp"
15          app:layout_constraintTop_toTopOf="parent">
16
17          <TableRow
18              android:layout_width="match_parent"
19              android:layout_height="match_parent">
 |
75          </TableRow>
76
77          <TableRow
78              android:layout_width="match_parent"
79              android:layout_height="match_parent">
80
81              <de.nitri.gauge.Gauge
82                  android:id="@+id/ga_tension"
83                  android:layout_width="100dp"
84
85                  android:layout_height="100dp"
86                  gauge:initialValue="0"
87                  gauge:lowerText="Tensión %"
88                  gauge:maxValue="120"
89                  gauge:minValue="0"
90                  gauge:needleColor="@color/color_tension"
91                  gauge:scaleColor="@color/color_escala"
92                  gauge:totalNicks="140"
93                  gauge:valuePerNick="1" />
 |
133          </TableRow>
```

Fig. 62: Código .XML del indicador de aguja porcentual, magnitud de ejemplo, tensión. Fuente: Alejandro RA.

Como se ha comentado con anterioridad, los indicadores, de ahora en adelante *gauge*, se inscriben en un *TableRow* y éstos, a su vez, en *tableLayout*, que se indica en las líneas 10-19 y 133. Cada *gauge* se puede personalizar, el texto que aparece en la sección inferior, el rango representable, el número de divisiones y los colores de la carcasa, escala, aguja y fondo. Todas las escalas poseen el mismo color, así como el fondo. Mientras, el texto y la aguja son función de la magnitud representada. Estas vicisitudes corresponden a las líneas 87-93.

Estos colores están codificados en *colors.xml*, éste permite, de manera dinámica, que cualquier cambio realizado en él se refleje a las variables internas que 'apunten' a ese color. No es realmente un apunte, ya que esta figura no existe en Java, pero el concepto es parecido, ya que para asignarlo se deben modificar los atributos, en este caso, del elemento *gauge*. Una vez 'apuntado' el atributo *needleColor*, por ejemplo, a cada cambio en *colors.xml*, se actualizará sin mediación del programador.

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3       <color name="colorPrimary">#000000</color>
4       <color name="colorPrimaryDark">#303F9F</color>
5       <color name="colorAccent">#FF4081</color>
6
7       <color name="color_tension">#1AE3D0</color>
12      <color name="color_rendimiento">#8BC34A</color>
13      <color name="color_escala">#123456</color>
14
15  </resources>
```

Fig. 63:C ódigo .XML del código de colores, magnitud de ejemplo, tensión y escala. Fuente: Alejandro RA.

Los colores se pueden configurar desde el recuadro situado a la derecha del número de línea, de este modo se abre una vista de arcoíris para escoger delicadamente el color deseado, o bien asignando el valor en formato HEX. Las líneas 3-5 se refieren a colores generales de la app, mientras que los utilizados de manera personalizada por las etiquetas, indicadores y demás están codificados en 7-13.

El diseño está hecho y localizado en la interfaz, ahora es necesario animar la aguja y para ello ha de enlazarse con la parte lógica. Para ello se ha dispuesto la siguiente codificación:

```
111         final Gauge gauge0, gauge1, gauge2, gauge3, gauge4, gauge5;
112         gauge0 = (Gauge) findViewById(R.id.ga_tension);
117         gauge5 = (Gauge) findViewById(R.id.ga_rendimiento);
269                          //Casting a float para la lectura del indicador de aguja y su movimiento al mismo
270                           float ch0manf = (float) ch0man;
275                           float ch5manf = (float) ch5man;
276
277                           gauge0.moveToValue(ch0manf);
282                           gauge5.moveToValue(ch5manf);
```

Fig. 64: Enlace lógico-gráfico de gauge y animación. Fuente: Alejandro RA.

Para declarar las variables *gauge*, se ha recurrido a la palabra clave *final*. Ésta impide una nueva referenciación de la variable que define, pero sí permite una modificación del estado interno de la misma. Cada variable de la parte lógica tiene asignado un *gauge* en la interfaz, por tanto, mediante el método *findViewById,* encuentra dicho indicador de tensión.

El método de animación de *gauge* está preparado para recibir como argumentos variables de tipo coma flotante. Esto lleva a que los valores calculados en analógico porcentual de la figura 59, requieran una transformación de tipo *casting*, para convertir en variables de tipo *float,* que no double, para lo que son destinadas las líneas 270-275. Finalmente la aguja se anima con el método *moveToValue*.

3.2.2.2.2.1.3. Impresión de las medidas en unidades de ingeniería

Este subcapítulo está dedicado a los valores que se representan por pantalla mediante elementos de tipo *TextView*, que contendrán las medidas de tensión, corriente, velocidad, par aproximado, potencia y rendimiento aproximado. Del mismo modo que se ha procedido con anterioridad, es menester crear estos elementos en el archivo .XML, para luego enlazarlos con la parte lógica y poder enviar los datos calculados en unidades de ingeniería.

```
21              <TextView
22                  android:id="@+id/ud_tension"
23                  android:layout_width="wrap_content"
24                  android:layout_height="wrap_content"
25                  android:text="@string/ud_tension"
26                  android:textColor="@color/color_tension"
27                  android:textSize="34sp" />
```

Fig. 65: ódigo .XML del cuadro de medidas para etiqueta de texto, magnitud de ejemplo, tensión. Fuente: Alejandro RA.

Éste es el último objeto incluido en el *TableRow* que serán ejemplificados. Comprobando los identificadores asignados, *id,* a los que se refiere en la sección lógica. El *TextView* definido en las líneas 21-27 corresponde con la etiqueta de la medida que se va a representar. Ésta etiqueta, al mostrar texto, conviene denotar la línea 25, que ha aparecido en otros elementos pero es ahora cuando cobra sentido el archivo *strings.xml*. Análogamente al archivo *colors.*xml, Éste aprueba modificar dinámicamente los atributos de objetos una vez establecida esa relación, de modo que una vez modificado el atributo *text*, a cada cambio en *strings.xml*, se actualizará automáticamente.

```
 1 <resources>
 2     <string name="app_name">IMIE Scope</string>
 3     <string name="et_inicial">Medidas:</string>
 4     <string name="et_tension">V</string>
 |
 9     <string name="et_rendimiento">e</string>
10
11     <string name="ud_tension">V (V)=</string>
 |
16     <string name="ud_rendimiento">  (%)≈</string>
 |
20 </resources>
```

Fig. 66: Código .XML de las cadenas de caracteres, magnitud de ejemplo, tensión y rendimiento. Fuente: Alejandro RA.

La segunda línea responde al nombre de la app, la segunda a la etiqueta de medidas, superior al *tableLayout*.

Ulteriormente, el *TextView* encargado de mostrar las medidas numéricas es generado. Éste sigue la misma estructura programática que el anterior.

```
29          <TextView
30              android:id="@+id/rms0"
31              android:layout_width="wrap_content"
32              android:layout_height="wrap_content"
33              android:layout_marginStart="8dp"
34              android:layout_marginLeft="8dp"
35              android:text="@string/et_tension"
36              android:textColor="@color/color_tension"
37              android:textSize="34sp" />
```

Fig. 67: Código .XML del TextView, magnidud de ejemplo, tensión. Fuente: Alejandro RA.

Destacando que su *id* es rms0 y su color es fácil comprobar que responde a la magnitud que representa. El *id*, será el encargado, una vez más de permitir el enlace lógico-gráfico. Procediendo a unir para luego enviar datos.

```
 82     private TextView rms0, rms1, rms2, rms3, rms4, rms5;

102         rms0 = (TextView) findViewById(R.id.rms0);
103         rms0.setText(String.valueOf("0"));

112         rms5 = (TextView)findViewById(R.id.rms5);
113         rms5.setText(String.valueOf("0"));
```

Fig. 68: Enlace lógico-gráfico de los TextView. Fuente: Alejandro RA.

Se ha usado de manera recurrente el método *findViewById* para reconocer en la interfaz el objeto y unirlos, por lo que se procede como hasta ahora, declarando las variables y luego buscándolas en el *layout.* El siguiente paso a seguir es insertarle un valor numérico al *TextView*, esto es posible realizarlo mediante el método *setText*. Pero los objetos que se están tratando esperan como argumento a mostrar una cadena de caracteres, ahí es donde entra en acción el método *String.valueOf*, para hacer algo parecido a un casting pero de número a cadena de caracteres. Le es asignado por defecto un valor cero, hasta que no se inicie la recepción de datos por parte de la app.

```
318                 rms0.setText(String.format("%.0f", ch0s.getY(datasto-1)*Un/100));
319                 rms1.setText(String.format("%.2f", ch1s.getY(datasto-1)*In/100.0));
320                 rms2.setText(String.format("%.0f", ch2s.getY(datasto-1)*nn/100.0));
321                 rms3.setText(String.format("%.2f", ch3s.getY(datasto-1)*Tn/100.0));
322                 rms4.setText(String.format("%.0f", ch4s.getY(datasto-1)*Pn/100.0));
323                 rms5.setText(String.format("%.2f", ch5s.getY(datasto-1)));
```

Fig. 69: Asignación de valores a las variables de representación de medidas. Fuente: Alejandro RA.

Para una correcta asignación, es necesario emplear el método *String.format*, que devuelve una cadena de texto con formato indicado. En este caso, se le va a ofrecer una variable de tipo *float* (%f), pero algunas medidas tienen interés con dos decimales, éstas son la corriente, el par y el rendimiento. De este modo, se facilita la lectura por parte del usuario. Obteniendo el número de decimales deseado según se ponen entre el símbolo de porcentaje y la efe, un punto seguido de los decimales requeridos.

Concluye esta parte de la aplicación dotando los *TextView* de su valor. Es destacable que el valor que se desea imprimir está guardado en las variables de tipo *XYSeries*, en concreto en la componente de ordenadas. Para encontrar cada valor, cabe poner de manifiesto que sólo es necesario encontrar el índice donde está cada valor mediante *getY*. Recordando que *datasto* es simplemente un contador de datos guardados. Como se observa en la línea 262 de la figura 59, los valores analógicos están en formato porcentual para ser representados en la gráfica. Con una multiplicación de cada variable por su valor nominal y dividiendo entre cien, se encuentra el valor a representar.

3.2.2.2.3. Uso de la aplicación

La aplicación se ha desarrollado para ser lo más intuitiva posible y su puesta a punto sea mínima. Pulsando en el dispositivo Android la aplicación IMIEScope, se abrirá la *Splash Activity* que se muestra el logo del ISEE. Para facilitar el uso, se conecta de manera automática como se reservó en el apartado 3.2.2.2.1.3. Y el usuario simplemente ha de pulsar el *ImageButton* destinado a iniciar la lectura. Éste tiene la forma habitual de *play*, y dará comienzo a la *Main Activity*.

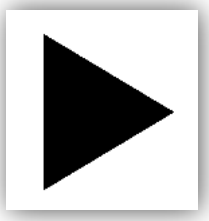


Fig. 70: Botón play. Fuente: Alejandro RA.

Dicha *Main Activity* graficará, como se ha comentado, las magnitudes medidas así como su representación en unidades de ingeniería y porcentualmente, quedando de la siguiente manera:

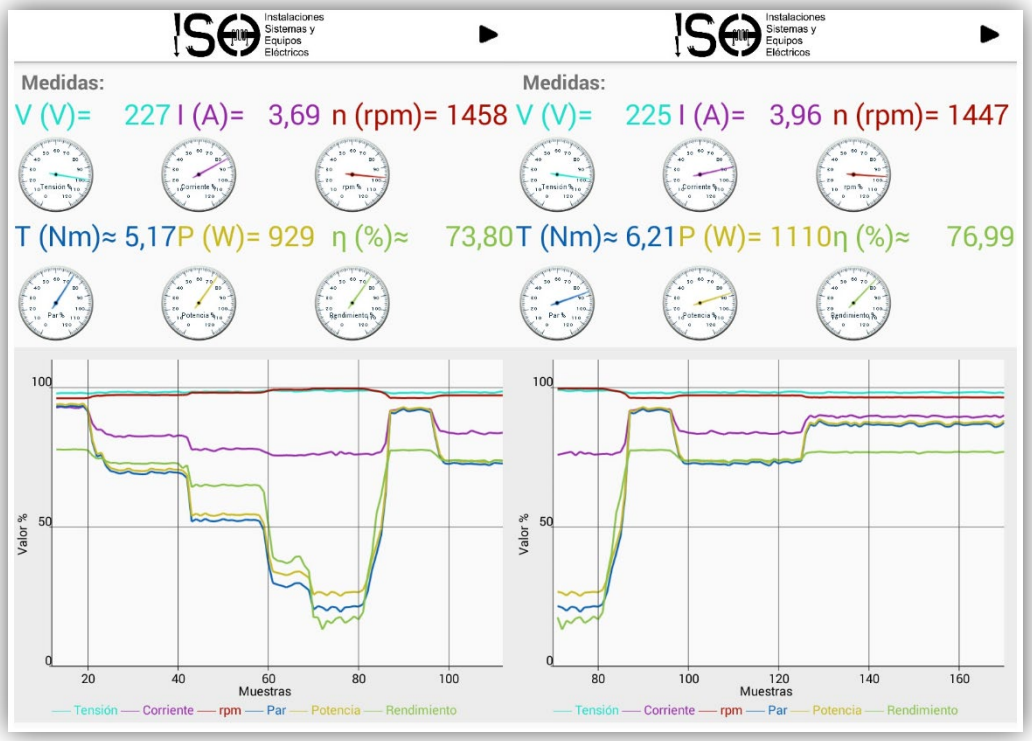


Fig. 71: Interfaz de la aplicación, muestra histórico. Fuente: Alejandro RA.

4. Descripción del banco de ensayos utilizado para probar el instrumento

Como se ha comentado previamente, el nuevo banco de trabajo constaría de la máquina de inducción, una máquina auxiliar, un autotransformador y el propio instrumento desarrollado en este trabajo. La idea sobre la que se asienta el proyecto es la de prescindir del analizador de redes, debido a sus limitaciones en cuanto a facilidad de uso, medidas mostradas y registro de valores en comparación con la del instrumento desarrollado y la conveniencia del mismo, así como el motivo económico. Los argumentos a favor del instrumento basado en ESP32 se expusieron en el apartado 3.1.2.

Sin embargo, para asegurar el correcto funcionamiento de la dupla instrumento-Tablet, ha sido necesaria la disposición del analizador de redes. En el anexo se adjuntan fotografías de las diferentes medidas tomadas por el analizador y las medidas tomadas por el ESP32.

Para realizar dichos ensayos, se utilizan siete puntos de carga diferentes, de carga completa, 1440 rpm a vacío, 1500 rpm. La experimentación de carga a vacío se espacia de 10 rpm en 10 rpm. En cada uno de los puntos se realiza una emisión de datos por parte del ESP32, es decir, cien muestras tomadas de cada medida y se anotan las proporcionadas por el analizador y el tacómetro digital. Después esas medidas se comparan con el catálogo para comprobar su coherencia.

Cabe destacar la necesidad de mantener en carga el conjunto máquina-máquina auxiliar hasta que los devanados alcancen la temperatura adecuada para que no haya mucha fluctuación en las magnitudes medidas debida al cambio de la resistencia de los devanados de la máquina.

Finalmente, el proceso de experimentación sigue unos pasos estrictos para garantizar la seguridad durante la conexión, toma de datos y desconexión así como la integridad de los materiales utilizados. El proceso es el siguiente:

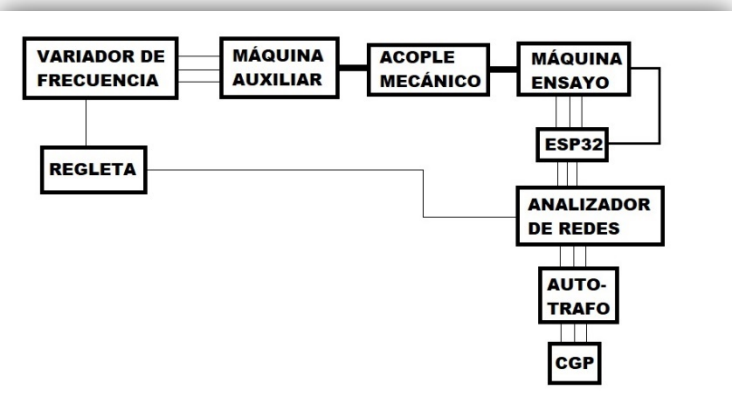


Fig. 72: Esquema de conexiones del banco de trabajo. Fuente: Alejandro RA.

1. Conexión del montaje: Primeramente se realizan las conexiones necesarias.
    A. Conexión monofásica del variador de frecuencia a la fuente alterna, pues la fuente de continua está conectada permanentemente. También está conectado el variador a la máquina auxiliar.
    B. Conexión trifásica de la máquina a ensayar con el ESP32.
    C. Conexión del sensor de velocidad con el ESP32.
    D. Conexión monofásica del analizador de redes.
    E. Conexión trifásica del ESP32 con el analizador de redes.
    F. Conexión trifásica del analizador de redes con el autotransformador.
    G. Conexión trifásica del autotransformador al CGP.
2. Comprobación del sentido de giro de la máquina auxiliar.
3. Comprobación del sentido de giro de la máquina a ensayar. Debe coincidir con el sentido de giro de la máquina auxiliar. En caso que no fuese el mismo, es necesario volver al punto D. para intercambiar dos fases entre sí. Para facilitar este paso, se utilizan los colores más parecidos entre el cable y el conector.
4. Encendido de la máquina auxiliar a través del variador de frecuencia a 50Hz.
5. Encendido de la máquina a ensayar a través del autotransformador, para que trabaje a tensión nominal, se gira el potenciómetro del autotransformador hasta un 55% aproximadamente.
6. Variación de la carga mediante el variador de frecuencia.

Para la desconexión, se ha de llevar el variador de frecuencia a 50Hz y deshacer los pasos a partir del 5.

No está de más recordar el riesgo eléctrico al que se está expuesto en experimentaciones con máquinas trifásicas y la necesidad de completar el protocolo al pie de la letra para minimizar estos riesgos. Por otra parte, se disponen elementos pasivos para evitar contactos directos, como puedan ser conectores de banana de seguridad. También activos, en el CGP.

4.1. Instrumento

Durante el desarrollo de la memoria se ha tratado extensamente la arquitectura del módulo encargado de la toma de medidas, el ESP32 y todos sus sensores asociados. Como se observa en la figura 72, el módulo posee ocho conexiones, seis de fases, una de tierra y otra del sensor de velocidad.

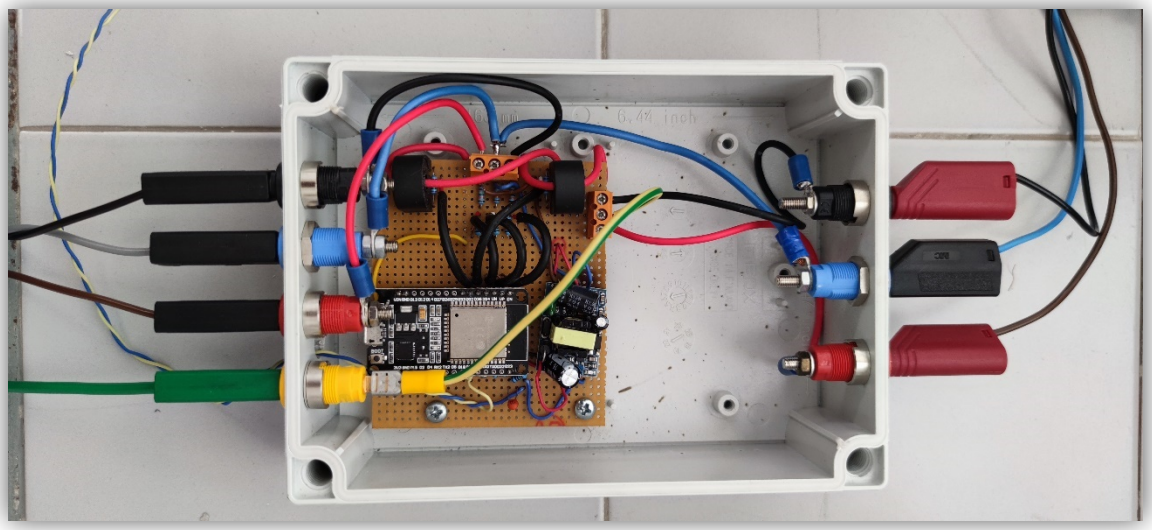


Fig. 73: Detalle del instrumento. Fuente: Alejandro RA.

Las conexiones de la izquierda son las de entrada, las de la derecha de salida y el cable trenzado el que recibe la información del *encoder*.

4.2. Regulador de frecuencia y máquina auxiliar

Gracias al regulador de frecuencia, se puede variar la carga, ya que éste alimenta la máquina auxiliar. Dicha máquina está acoplada mecánicamente a la máquina a ensayar. Motor y máquina auxiliar son completamente idénticos.

El regulador posee un potenciómetro que permite esa modulación de frecuencia.

Fig. 74: Variador de frecuencia. Fuente: Alejandro RA.

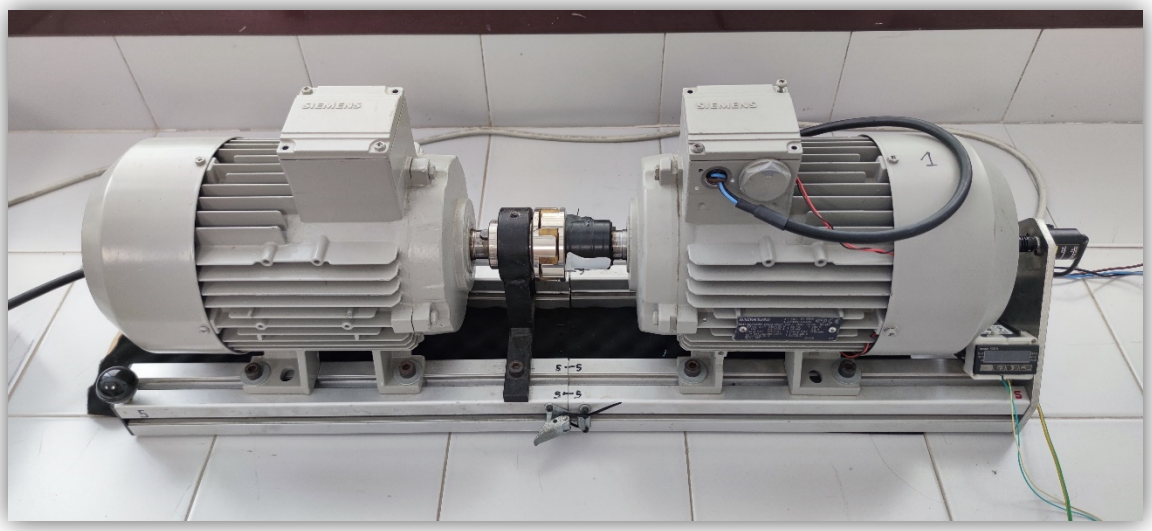


Fig. 75: Acople entre máquinas. Fuente: Alejandro RA.

4.3. Analizador

Un analizador de redes es un dispositivo que nos brinda medidas instantáneas de las fases a las que está conectado. Entre ellas corriente, tensión, potencias, factor de potencia… Es un dispositivo con un rango de uso amplio si dispone de transformadores de corriente para permitir una monitorización en redes de mayores magnitudes para las que estaría preparado de origen.

Fig. 76: Analizador de redes. Fuente: Alejandro RA.

En este caso se trata de un Circutor CVM 96 montado sobre una caja en la que se encuentran los transformadores de corriente adecuados para esas corrientes.

La entrada se conecta al autotransformador y la salida al instrumento desarrollado.

4.4. Máquina asíncrona

Tanto el motor como la máquina auxiliar son idénticas como se ha comentado, se trata de máquinas de jaula de ardilla de la marca Siemens, modelo 1LA7090-4AA10. Como características de tensión, nominal 230V, corriente de 4'5A en configuración estrella y 1'1kW de potencia. Para más información, la hoja de especificaciones está anexada.

Tras experimentación, se ha calculado el valor de par de pérdidas en el entrehierro y pérdidas mecánicas como un mismo par de pérdidas, que asciende a 1'07Nm. Es un valor de especial interés ya que no aparece en dicha hoja de especificaciones.

4.5. Autotransformador

Un autotransformador, a grandes rasgos, se podría definir como un transformador de una sola bobina, al que, mediante una derivación, se permite la obtención de un par de circuitos, primario y secundario, tal y como muestra la imagen.

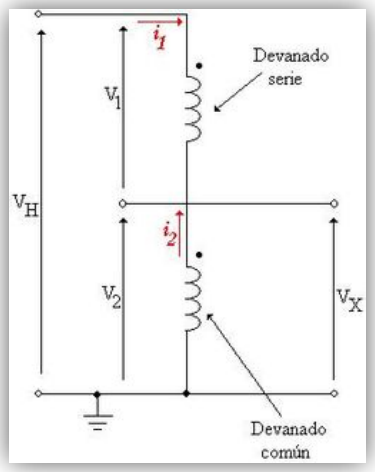


Fig. 78: Principio de funcionamiento del autotransformador. Fuente: Alejandro RA.

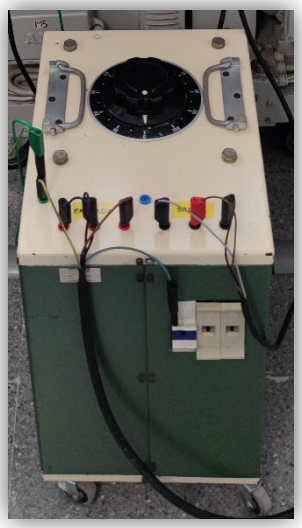


Fig. 77: Autotransformador. Fuente: Alejandro RA.

En el caso del utilizado para la experimentación, es un autotransformador trifásico de tipo 'Variac'. Así se conocen los autotransformadores que disponen de una escobilla para poder regular en tensión la salida o secundario.

5. Resultados

Durante el desarrollo de los experimentos, tal y como se puede ver en los anexos a la par que se ha explicado en el punto 3.1.6.3.4., el instrumento captura cien valores de cada una de las variables que se pretenden medir. Los datos recabados por el ESP32 durante los ensayos se han tratado mediante MS Excel.

Para una correcta comprensión del objeto del trabajo y de la capacidad de este tipo de duplas instrumento-Tablet, se dividirá el capítulo en dos partes. La primera a 1440 rpm, es decir, a valores nominales, y los resultados globales, es decir, las gráficas que acompañarían a la hoja de especificaciones del motor. Gracias a esta división, será posible discriminar la validez de las medidas que emite el instrumento y cómo afectan en diferentes puntos de funcionamiento. Por ello se recomienda revisar las hojas anexas, en concreto las 'Tablas y gráficas de ensayos, medidas.pdf'.

5.1. Resultados a 1440rpm

A 1440rpm se alcanzan los valores nominales de tensión y corriente, como se puede chequear de manera doble mediante el instrumento y el analizador de redes. Para ello se hace una captura entera de datos y se tratan con Excel. Para una mejor lectura, se graficarán los datos quedando de la siguiente manera:

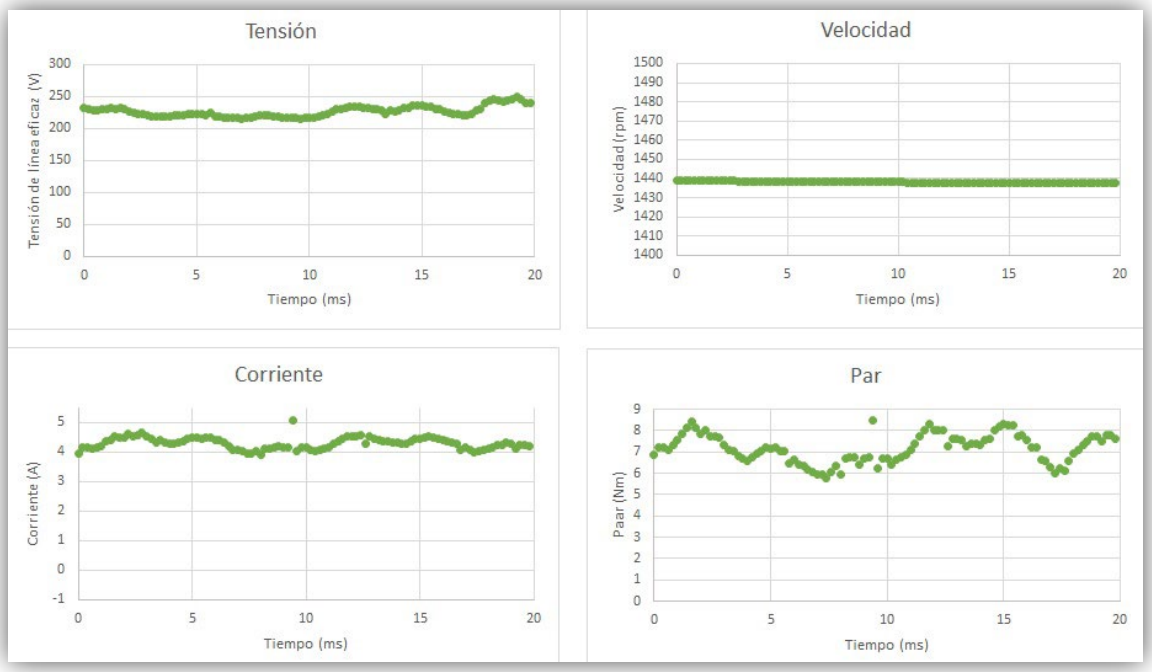


Fig. 79: Gráficas de V, I, n y T del instrumento a 1440rpm. Fuente: Alejandro RA.

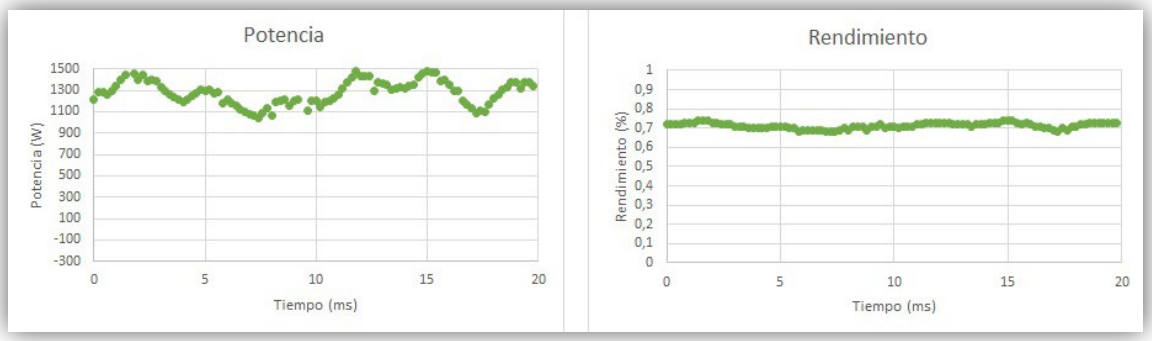


Fig. 80: Gráficas de P y η del instrumento a 1440rpm. Fuente: Alejandro RA.

Debido a los cien valores recabados por el instrumento, la variación de los valores muestreados es relativamente elevada y traduce la variabilidad real de la tensión (desequilibrio de la red) y de la corriente (causada principalmente por variaciones de la carga y del consumo de potencia reactiva por posibles excentricidades del rotor). Estas variaciones son completamente normales, aunque no se muestran al usuario del analizador de redes. En nuestro caso tampoco se mostrarán al enviar a la interfaz de usuario valores medios para un periodo.

Si a estos datos, se les calcula el promedio, obtenemos los siguientes valores:

Mediciones a 1440 rpm							
Promedios:	226,4351	4,3048	14382,09	7,1688	1285,8418	0,7137	1438,209
Tiempo (ms)	Tensión (V)	Corriente (A)	Velocidad*10 (rpm)	Par (Nm)	Potencia (W)	Rendimiento (%)	Velocidad (rpm)

Fig. 82: Promedio de mediciones del instrumento a 1440rpm. Fuente: Alejandro RA.

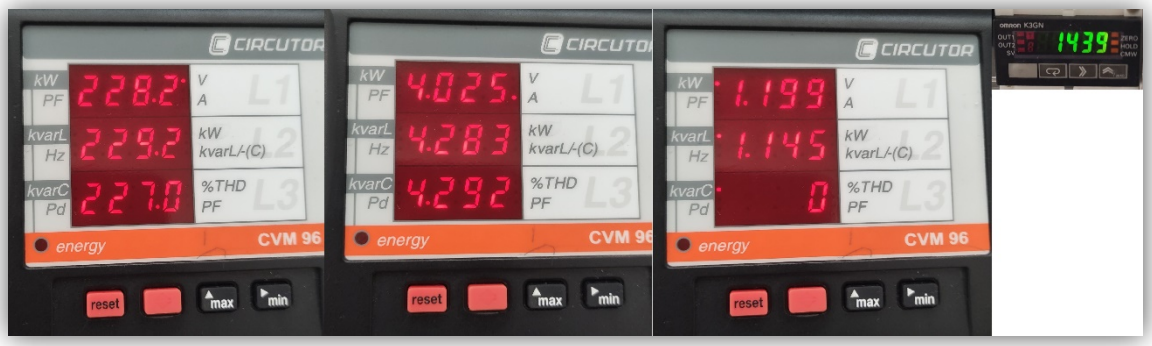


Fig. 81: Mediciones del analizador de redes a 1440rpm. Fuente: Alejandro RA.

A priori, parece haber una buena relación entre las medidas proporcionadas por el analizador de energía, el tacómetro digital y las indicadas por el instrumento desarrollado. Es cierto que existe una diferencia de cerca de 80W en las lecturas de potencia que en realidad representa sólo un 1,2% del fondo de escala del analizador de energía y, por tanto, también dentro de los márgenes esperados.

Es conveniente indicar que el instrumento desarrollado utiliza sensores de corriente y de tensión de precisión 0,5% que ya están cerca de la discrepancia mostrada y, además, que el analizador de energía está diseñado para medir energía en redes con una variabilidad de la potencia más baja de la que corresponde al banco de ensayos, lo que puede degradar la precisión real respecto de la especificada.

5.2. Resultados globales

Visto el primer valor en carga, completa, se repite la mecánica con los siguientes puntos de carga. Están espaciados por 10rpm hasta 1490, no llegamos a vacío porque es un punto que no nos interesa, ya que las máquinas de inducción como esta no trabajan en vacío.

Para visualizar estos puntos y cómo se ajustan a los valores dados en el catálogo por parte del fabricante, se han graficado también en MS Excel. Algunas de estas medidas serán contrastadas con el analizador de redes y otras simplemente con el catálogo. Siempre teniendo en cuenta que las medidas tomadas por el instrumento una vez calibrado, presenta una precisión muy superior al resto de elementos. Pues los ensayos por parte del fabricante se realizan con máquinas nuevas, en un ambiente ideal para arrojar los mejores números de rendimiento, etc. Y por parte del analizador en que toma muchas menos muestras que el instrumento.

Separaremos cada una de las gráficas generadas para comentar la importancia o no de la posible desviación de los valores.

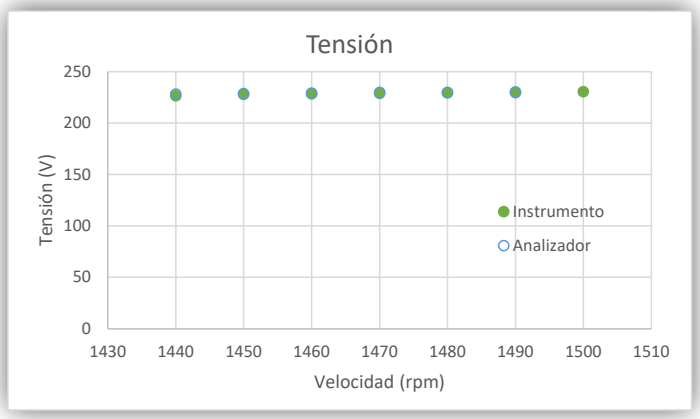


Fig. 83: Gráfica de tensión carga-vacío. Fuente: Alejandro RA.

Se mantiene durante todo el ensayo de carga a vacío tanto en el analizador de redes como en el instrumento. Es un resultado para nada sorprendente debido a la naturaleza del diseño del motor. Una sobretensión provocaría calentamientos para los que no está preparada la máquina y una deficiencia de tensión imposibilitaría el correcto establecimiento de los enlaces de flujo necesarios para funcionar.

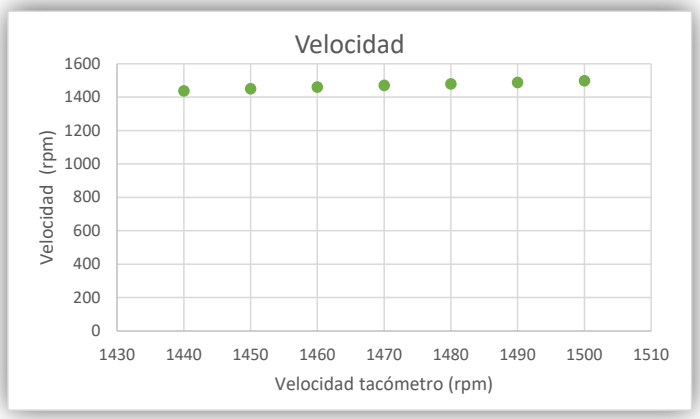


Fig. 84: Gráfica de la velocidad carga-vacío. Fuente: Alejandro RA.

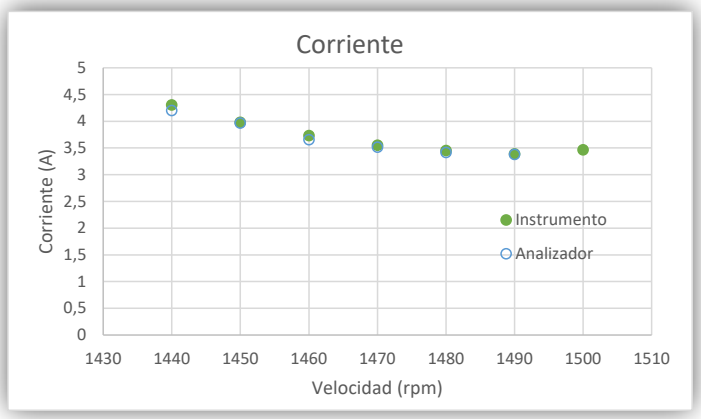


Fig. 86: Gráfica de corriente carga-vacío. Fuente: Alejandro RA.

Observamos una ligera discrepancia, de alrededor del 2'3% en su punto más disperso debido en parte a la propia precisión del analizador externo y del instrumento desarrollado y por otra, probablemente, a la oscilación de las medidas de corriente (relativamente elevada) que impide tomar una lectura estable de la pantalla del analizador y asegurar además que la lectura tomada coincide en el tiempo con las medidas tomadas por el instrumento desarrollado, al no contar ninguno de los dos aparatos de medida con mecanismos de sincronización de la medida (*trigger* externo).

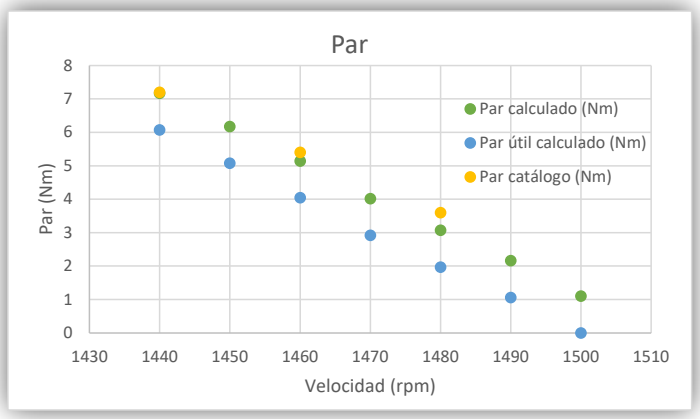


Fig. 85: Gráfica de par carga-vacío. Fuente: Alejandro RA.

En cuanto al par, vemos un ajuste muy bueno del calculado por el instrumento y el rendimiento de catálogo, sobre todo a valor de carga nominal, que están solapados prácticamente. Conforme se acercan a vacío, las medidas varían, pero no es importante, ya que como hemos comentado, las condiciones del fabricante siempre son óptimas. Se ha aprovechado para representar también el par útil, esto es restándole el par de pérdidas mecánicas y pérdidas en el entrehierro, como se ha comentado anteriormente.

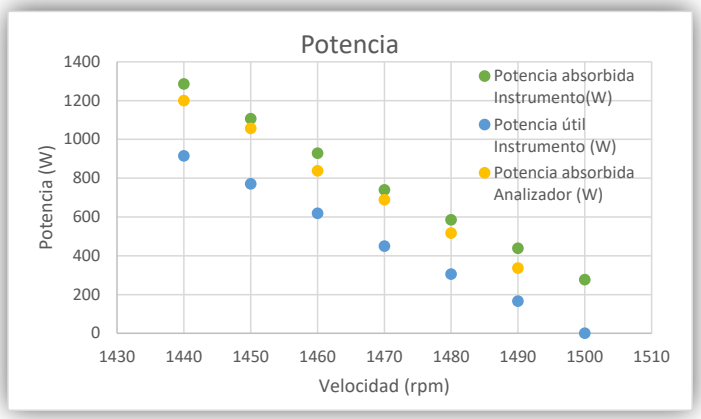


Fig. 88: Gráfica de potencia carga-vacío. Fuente: Alejandro RA.

La gráfica anterior muestra la misma variabilidad observada en el apartado anterior, unos 80W entre el analizador y el instrumento. Esta variabilidad, en realidad representa sólo un 1,2% del fondo de escala del analizador de energía y, por tanto, también dentro de los márgenes esperados. Si tenemos en cuenta la tolerancia de los sensores utilizados por el instrumento, está dentro de los valores esperados, como ya se ha comentado.

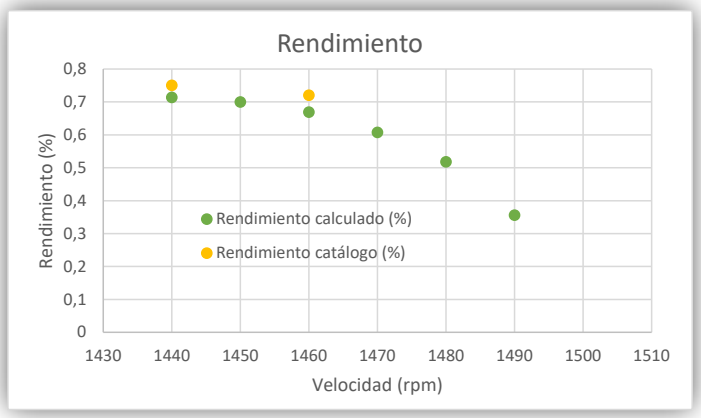


Fig. 87: Gráfica de rendimiento carga-vacío. Fuente: Alejandro RA:

Finalizando datos graficados, se muestra el rendimiento comparado con el rendimiento de catálogo que, como era de esperar, es de hasta un 5% de diferencia en plena carga.

Dichas diferencias son más sencillas de comprender con las tablas comparativas a continuación, éstas resumen todas las gráficas anteriores.

Velocidad tacómetro (rpm)	Velocidad calculada Instrumento (rpm)	Error velocidad (%)	Urms medida Instrumento (V)	Urms medida Analizador (V)	Error Urms (%)	Irms medida Instrumento (A)	Irms medida Analizador (A)	Error Irms (%)
1439	1438,209	0,055	226,435	228,133	0,750	4,305	4,200	2,434
1451	1450,531	0,032	227,896	228,533	0,280	3,981	3,967	0,351
1461	1460,536	0,032	228,420	229,100	0,298	3,732	3,655	2,055
1470	1471,198	0,081	228,866	229,667	0,350	3,552	3,518	0,967
1480	1480,309	0,021	229,294	229,800	0,221	3,453	3,415	1,102
1490	1488,915	0,073	229,489	230,133	0,281	3,389	3,378	0,300
1500	1498,434	0,104	230,533	-	-	3,466	-	-

Fig. 90: Tabla comparativa de velocidad, tensión y corriente carga-vacío. Fuente: Alejandro RA.

Par calculado Instrumento (Nm)	Par catálogo (Nm)	Error par (%)	Par útil calculado Instrumento (Nm)	Potencia absorbida caculada Instrumento (W)	Potencia absorbida Analizador (%)	Error potencia absorbida	Potencia útil calculada Instrumento (Nm)	Rendimiento calculado Instrumento (%)	Rendimiento catálogo (%)	Error rendimiento (%)
7,169	7,200	0,435	6,069	1285,842	1199,000	6,754	914,608	0,714	0,750	5,086
6,175	-	-	5,075	1106,444	1056,000	4,559	771,184	0,699	-	-
5,145	5,400	4,962	4,045	928,199	837,000	9,825	618,913	0,669	0,720	7,607
4,017	-	-	2,917	739,704	688,000	6,990	449,083	0,607	-	-
3,068	3,600	17,348	1,968	584,703	517,000	11,579	305,073	0,518	-	-
2,160	-	-	1,061	438,228	336,000	23,328	165,488	0,356	-	-
1,099	-	-	-	276,243	-	-	-	-0,202	-	-

Fig. 89: Tabla comparativa de par, potencia y rendimiento carga-vacío. Fuente: Alejandro RA.

6. Conclusiones

Como resumen del presente TFG, se ha realizado un análisis detallado del problema conceptual para establecer una línea de solución dando paso a la materialización de un diseño.

Para diseñar una solución óptima, se ha realizado un análisis de tecnologías competitivas para el problema en cuestión dando a luz a la dupla microcontrolador-Tablet. El primero se dispone en la máquina a analizar y se conecta inalámbricamente al dispositivo móvil. Dicha asociación permite la escisión del problema material en dos.

Para el módulo de medida, se ha diseñado una placa de circuitos electrónicos que permiten medir y acondicionar la medida para el correcto funcionamiento del microcontrolador. Éste se conecta con la Tablet para proporcionarle dichas medidas, las cuales ha procesado mediante un programa de cálculo. Para todo el proceso de medida, cálculo y reporte de medidas, se ha desarrollado un código en Arduino IDE.

Los datos reportados por el módulo de medida se transmiten mediante Bluetooth a la Tablet, que los recoge y les proporciona un segundo tratamiento para la representación por pantalla. Se presentan en unidades de ingeniería, seguidos de un indicador de aguja porcentual y una gráfica porcentual. Todo el proceso ha sido codificado en Android Studio IDE.

La verificación del correcto funcionamiento de los códigos ha permitido visualizar ligeras discrepancias, dentro de la tolerancia esperada y aceptable, de algunas de las medidas, siempre teniendo en cuenta que las ecuaciones presentadas en el apartado de cálculo describen muy acertadamente el comportamiento de la máquina, por lo que las magnitudes medidas por el módulo son ajustadas a la máquina en cuestión.

En resumidas cuentas, se ha desarrollado un instrumento que presenta ventajas de modularidad y universalidad por un precio muy reducido en comparación con las alternativas ofrecidas por el mercado.

7. Bibliografía

Android developer. (s.f.). Obtenido de https://developer.android.com/guide/topics/connectivity/Bluetooth?hl=es-419

Kolban. (12 de Mayo de 2018). *ESP 32 Forum*. Obtenido de https://ESP32.com/viewtopic.php?t=4978#p21478

Martínez Román, J. A. (2020). Ampliación de Máquinas eléctricas: Modelos dinámicos de máquinas de alterna. Valencia.

Mena Roa, M. (30 de Julio de 2020). *Statista*. Obtenido de https://es.statista.com/grafico/18920/cuota-de-mercado-mundial-de-smartphones-por-sistema-operativo/

Open handset alliance. (s.f.). Obtenido de https://www.openhandsetalliance.com/

Peñalba, I. (30 de Enero de 2020). *El Español*. Obtenido de https://www.elespanol.com/elandroidelibre/moviles-android/20200130/no-vas-creer-moviles-vendieron/463705303_0.html

Programar fácil. (s.f.). Obtenido de https://programarfacil.com/esp8266/ESP32/

Programar fácil. (s.f.). Obtenido de https://programarfacil.com/blog/arduino-blog/arduino-uno-r3/

Random nerd tutorials. (s.f.). Obtenido de https://randomnerdtutorials.com/ESP32-pinout-reference-gpios/

SWA. (s.f.). Obtenido de https://www.survivingwithandroid.com/android-chart-tutorial-achartengine/

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

DISEÑO DE UN INSTRUMENTO CON INTERFAZ MÓVIL PARA LA MEDICIÓN DE MAGNITUDES ELÉCTRICAS Y MECÁNICAS EN MÁQUINAS ASÍNCRONAS DE 1.1KW

PRESUPUESTO

TRABAJO DE FINAL DE GRADO

ALEJANDRO ROMAGUERA ASENSIO

SEPTIEMBRE 2021

En el presente documento se expone, mediante capítulos, el desglose de costos asociados al desarrollo del proyecto Diseño de un instrumento con interfaz móvil para la medición de magnitudes eléctricas y mecánicas en máquinas asíncronas de 1.1kW.

El presupuesto se compone de ocho capítulos y obedece a una separación de materiales, alquileres y mano de obra.

El primer cuadro corresponde al coste de materiales. Como se puede comprobar, el coste del microcontrolador es irrisorio en comparación con el coste de la caja y conexiones con elementos externos. Para un montaje en el laboratorio es imprescindible este tipo de montaje, pero en un entorno industrial, el coste del capítulo se reduciría drásticamente. Siendo esta cuestión, motivo de estudio de adaptabilidad a un módulo más compacto.

Capítulo 1: Materiales				
Nº	Concepto	Precio unitario	Cantidad	Precio
1.	ESP32 Devkit V1	9,500 Uds	1	9,500
2.	Resistencias 75	0,014 Uds	2	0,028
3.	Resistencia 1 5	0,013 Uds	3	0,039
4.	Resistencia 240	0,108 Uds	3	0,324
5.	Resistencia 150	0,028 Uds	2	0,056
6.	Condensador 10nF	0,559 Uds	1	0,559
7.	Fuente de alimentación 5V	3,270 Uds	1	3,270
8.	Cable de cobre 3x2,5mm^2	0,360 m	1	0,360
9.	Conector banana hembra 32A 1KV	3,700 Uds	7	25,900
10.	Talema AX0500	4,050 Uds	2	8,100
11.	Caja eléctrica FIBOX TEMPO TPC 241911T	20,600 Uds	1	20,600
12.	PLACA PERFORADA BAQUELITA CUADROS 100x80mm	2,630 Uds	1	2,630
13.	Tornillería	0,010 Uds	14	0,140
14.	Bloques terminales de montaje en PCB tricontacto	1,410 Uds	2	2,820
	Coste del capítulo (€)			**74,326**

Debido a la naturaleza del TFG, para el desarrollo del proyecto se disfruta de las ventajas que aporta la universidad en materia de software para el estudiantado. De ahí que el coste de dicho capítulo sea nulo. Sin embargo, cabe destacar que las dos herramientas esenciales de programación disponen de una licencia totalmente abierta y libre.

Capítulo 2: Software				
Nº	Concepto	Precio unitario	Cantidad	Precio
1.	Arduino IDE	0,000	1	0,000
2.	Android Studio IDE	0,000	1	0,000
3.	AutoCad 2021	Licencia universitaria	1	0,000
4.	Microsoft Office	Licencia universitaria	1	0,000
	Coste del capítulo (€)			**0,000**

Continuando, el capítulo actual responde al alquiler por meses de las herramientas de trabajo para el desarrollo de dicho proyecto. Corresponden al ordenador personal, con un coste reducido, que se podría haber anulado con uno de los ordenadores del servicio de préstamos de la universidad, pero no se ha contemplado debido a la inconveniencia de renovar el préstamo. Por otro lado, la Tablet, como se comenta a lo largo de la memoria, la facilita el departamento.

Con este capítulo se cubren los capítulos de materiales y alquileres.

Capítulo 3: Hardware					
Nº	Concepto	Precio unitario		Cantidad	Precio
1.	Ordenador personal	10,000	Mes	6,000	60,000
2.	Tablet	0,000	Mes	1,000	0,000
	Coste del capítulo (€)				**60,000**

En tablas posteriores, se tratará el coste de la mano de obra requerida. Para ello se ha tomado el sueldo base de un graduado en Ingeniería de Tecnologías Industriales en España de modo aproximado. Se ha estimado 37.000€ anuales.

$$\frac{37000€}{año}\frac{1año}{12\ meses}\frac{1mes}{20días\ hábiles}\frac{1día\ hábil}{8h}=\frac{19'29€}{h}$$

Por otra parte, se ha requerido ayuda aportada por el tutor del presente TFG, Doctor Ingeniero Industrial, del cual se ha estimado el coste de una hora de mano de obra a 90€. Su labor es la revisión sistemática de los productos derivados de la actividad del proyecto. Siendo un valor fundamental para su correcto desarrollo.

Cabe destacar que cada capítulo posee un concepto de costes directos complementarios. Responde a todos los costes de difícil cuantificación, que se añaden como un porcentaje, en este caso, a la mano de obra.

El siguiente capítulo describe el coste del estudio previo, en el que se inscriben tanto el análisis del montaje previo, como la evaluación de alternativas tecnológicamente viables. Además del diseño de la placa asociada al ESP 32, su desarrollo e implementación para llegar al resultado final. En dicho capítulo se incluye la programación del SoC y las pruebas de funcionamiento.

Capítulo 4: Diseño y desarrollo del instrumento					
Nº	Concepto	Precio unitario		Cantidad	Precio
1.	Graduado en Ingeniería de Tecnologías Industriales	19,290	Hora	75	1446,750
2.	Doctor Ingeniero Industrial	90,000	Hora	9	810,000
3.	Costes directos complementarios	5,000	%		112,838
	Coste del capítulo (€)				**2369,588**

Prosiguiendo, se muestra el coste del capítulo referente a la aplicación de Android, análogamente al capítulo anterior, consta de un estudio previo, diseño, implementación y pruebas. Como se comenta a lo largo de la memoria, el concepto de entorno de diseño integral que posee Android Studio permite que, pese a que las horas de trabajo sean ligeramente superiores, no requieran de tanta supervisión.

Capítulo 5: Diseño y desarrollo de la aplicación				
Nº Concepto	Precio unitario		Cantidad	Precio
1. Graduado en Ingeniería de Tecnologías Industriales	19,290	Hora	80	1543,200
2. Doctor Ingeniero Industrial	90,000	Hora	5	450,000
3. Costes directos complementarios	5,000	%		99,660
Coste del capítulo (€)				**2092,860**

Tras ser planteadas, desarrolladas y realizadas las partes por separado, conviene realizar un ensamblaje de las mismas. Para ello, se destina un capítulo de ensayos y toma de muestras, de uso y buen funcionamiento. Dicho capítulo se compone de los sucesivos ensayos en diferentes puntos de funcionamiento de la máquina. También la toma de muestras y tratado de datos para su posterior verificación y contraste con datos de fabricante.

Capítulo 6: Realización de ensayos				
Nº Concepto	Precio unitario		Cantidad	Precio
1. Graduado en Ingeniería de Tecnologías Industriales	19,290	Hora	35	675,150
2. Doctor Ingeniero Industrial	90,000	Hora	3	270,000
3. Costes directos complementarios	5,000	%		47,258
Coste del capítulo (€)				**992,408**

Una vez terminados los capítulos designados a las piezas fundamentales del proyecto, conviene realizar una redacción de la memoria del TFG, y la elaboración de la documentación técnica asociada a la misma. Es el capítulo de mayor coste debido a las horas dedicadas al planteamiento unívoco de los conceptos tratados a lo largo del mismo. Además de la cuidada revisión que merece la elaboración de dicha memoria.

Capítulo 7: Redacción de la documentación técnica				
Nº Concepto	Precio unitario		Cantidad	Precio
1. Graduado en Ingeniería de Tecnologías Industriales	19,290	Hora	100	1929,000
2. Doctor Ingeniero Industrial	90,000	Hora	12	1080,000
3. Costes directos complementarios	5,000	%		150,450
Coste del capítulo (€)				**3159,450**

Dando por terminado el detallado análisis y planteamiento del proyecto, el último paso es el montaje. Debido al trabajo ya realizado en capítulos anteriores, el montaje es rápido y no presenta complicación alguna.

Capítulo 8: Montaje del instrumento en el banco de trabajo				
Nº Concepto	Precio unitario		Cantidad	Precio
1. Graduado en Ingeniería de Tecnologías Industriales	19,290	Hora	10	192,900
2. Doctor Ingeniero Industrial	90,000	Hora	1	90,000
3. Costes directos complementarios	5,000	%		14,145
Coste del capítulo (€)				**297,045**

Por tanto, el coste total del presupuesto, atendiendo a los siguientes parámetros;

- IVA: Impuesto sobre el Valor Añadido, contabilizado en un 21% para este TFG.
- Beneficio industrial: porcentaje de beneficio reportado al proyectista. Asciende al 6%.
- Gastos generales: corresponde a todos los gastos asociados a la actividad productiva, computando un 12% del PEM.

RESUMEN PRESUPUESTO			
	CONCEPTO	COSTE	
Capítulo 1.	Materiales	74,33	€
Capítulo 2.	Software	0,00	€
Capítulo 3.	Hardware	60,00	€
Capítulo 4.	Diseño y desarrollo del instrumento	2369,59	€
Capítulo 5.	Diseño y desarrollo de la aplicación	2092,86	€
Capítulo 6.	Realización de ensayos	992,41	€
Capítulo 7.	Redacción de la documentación técnica	3159,45	€
Capítulo 8.	Montaje del instrumento en el banco de trabajo	297,05	€
	Presupuesto de ejecución material (PEM)	**9045,68**	**€**
	Gastos generales 12%	1085,48	€
	Beneficio Industrial 6%	542,74	€
	Presupuesto de ejeución por contrata (PEC)	**10673,90**	**€**
	IVA 21%	2241,52	€
	Presupuesto base de licitación (PBL)	**12915,42**	**€**

Finalmente, el presupuesto supone un importe de DOCE MIL NOVECIENTOS QUINCE EUROS CON CUARENTA Y DOS CÉNTIMOS.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

DISEÑO DE UN INSTRUMENTO CON INTERFAZ MÓVIL PARA LA MEDICIÓN DE MAGNITUDES ELÉCTRICAS Y MECÁNICAS EN MÁQUINAS ASÍNCRONAS DE 1.1KW

ANEXOS

TRABAJO DE FINAL DE GRADO

ALEJANDRO ROMAGUERA ASENSIO

SEPTIEMBRE 2021

Tablas y gráficos de resultados de ensayos de la máquina de inducción
Código App
Catálogo motor

Tablas y gráficos de resultados de ensayos de la máquina de inducción

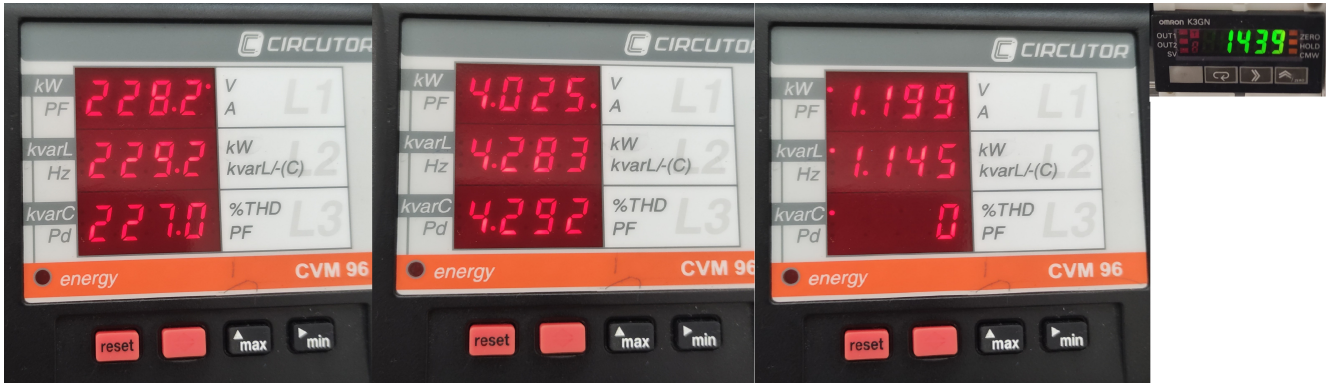
Mediciones a 1440 rpm

Promedios:	226,4351	4,3048	14382,09	7,1688	1285,8418	0,7137	1438,209
Tiempo (ms)	Tensión (V)	Corriente (A)	Velocidad*10 (rpm)	Par (Nm)	Potencia (W)	Rendimiento (%)	Velocidad (rpm)
0	233,02	3,96	14391	6,89	1216,53	0,72	1439,1
0,2	230,82	4,16	14391	7,2	1279,95	0,72	1439,1
0,4	229,21	4,17	14390	7,22	1283,24	0,72	1439
0,6	228,44	4,12	14391	7,13	1265,48	0,72	1439,1
0,8	229,32	4,16	14390	7,31	1296,77	0,73	1439
1	230,4	4,2	14390	7,59	1344,51	0,73	1439
1,2	231,81	4,36	14390	7,85	1396,11	0,73	1439
1,4	230,27	4,42	14389	8,15	1449,05	0,74	1438,9
1,6	231,41	4,55	14389	8,43	1502,31	0,74	1438,9
1,8	230,01	4,48	14389	8,17	1456,18	0,74	1438,9
2	226	4,5	14388	7,83	1404,3	0,73	1438,8
2,2	225,11	4,61	14388	8,04	1445,09	0,73	1438,8
2,4	223,17	4,55	14388	7,71	1389,53	0,72	1438,8
2,6	222,14	4,59	14388	7,73	1395,5	0,72	1438,8
2,8	220,34	4,65	14387	7,66	1389,53	0,72	1438,7
3	219,21	4,53	14387	7,32	1326,9	0,71	1438,7
3,2	218,31	4,47	14387	7,13	1291,24	0,71	1438,7
3,4	218,57	4,32	14387	7,03	1265,02	0,71	1438,7
3,6	218,44	4,43	14387	6,81	1237,86	0,7	1438,7
3,8	217,78	4,32	14387	6,71	1214,79	0,7	1438,7
4	220,13	4,29	14386	6,61	1197,35	0,7	1438,6
4,2	220,79	4,3	14386	6,73	1215,9	0,7	1438,6
4,4	221,28	4,35	14386	6,91	1248,01	0,7	1438,6
4,6	222,72	4,37	14386	7,02	1267,82	0,71	1438,6
4,8	223,13	4,47	14386	7,22	1306,72	0,71	1438,6
5	221,96	4,49	14386	7,14	1295,59	0,71	1438,6
5,2	222,35	4,49	14386	7,21	1306,37	0,71	1438,6
5,4	221,21	4,46	14386	7,05	1277,56	0,7	1438,6
5,6	224,05	4,48	14386	7,05	1280,22	0,7	1438,6
5,8	218,56	4,49	14386	6,44	1185,28	0,68	1438,6

6	218,47	4,43	14386	6,63	1210,59	0,69	1438,6
6,2	216,47	4,41	14386	6,41	1174,5	0,69	1438,6
6,4	216,06	4,35	14386	6,34	1158,7	0,69	1438,6
6,6	216,01	4,22	14386	6,18	1124,09	0,69	1438,6
6,8	216,14	4,1	14386	6,04	1094,02	0,69	1438,6
7	215,67	4,08	14386	5,92	1074,11	0,68	1438,6
7,2	216,54	4,02	14385	5,93	1069,68	0,68	1438,5
7,4	216,88	3,96	14385	5,79	1044,07	0,68	1438,5
7,6	217,73	3,97	14385	6,04	1083,6	0,69	1438,5
7,8	220,05	4,02	14385	6,33	1133,63	0,7	1438,5
8	220,51	3,93	14385	5,94	1065,9	0,69	1438,5
8,2	221	4,11	14385	6,68	1194,43	0,71	1438,5
8,4	217,86	4,12	14385	6,74	1204,39	0,71	1438,5
8,6	217,78	4,16	14384	6,77	1212,71	0,71	1438,4
8,8	217,04	4,21	14384	6,39	1156,76	0,69	1438,4
9	216,68	4,18	14384	6,72	1206,6	0,71	1438,4
9,2	217,03	4,17	14383	6,76	1211,25	0,71	1438,3
9,4	216,86	5,09	14383	8,5	1558,38	0,72	1438,3
9,6	214,22	4,02	14383	6,22	1116,46	0,7	1438,3
9,8	216,24	4,18	14382	6,71	1204,56	0,71	1438,2
10	216,61	4,15	14382	6,7	1200,16	0,71	1438,2
10,2	217,03	4,08	14381	6,42	1151,28	0,7	1438,1
10,4	218,04	4,05	14380	6,66	1187,31	0,71	1438
10,6	220,69	4,08	14380	6,75	1203,44	0,71	1438
10,8	222,59	4,11	14379	6,89	1227,6	0,71	1437,9
11	226,74	4,15	14378	7,09	1262,78	0,72	1437,8
11,2	230,1	4,27	14378	7,42	1321,32	0,72	1437,8
11,4	230,38	4,37	14377	7,71	1376,21	0,73	1437,7
11,6	231,72	4,46	14376	8	1427,47	0,73	1437,6
11,8	234,91	4,56	14376	8,3	1481,98	0,73	1437,6
12	234,23	4,53	14376	8	1433,04	0,73	1437,6
12,2	234,17	4,56	14376	8,01	1436,75	0,73	1437,6
12,4	232,76	4,59	14376	8,01	1439,33	0,73	1437,6

12,6	231,93	4,3	14376	7,27	1301,39	0,72	1437,6
12,8	230,19	4,52	14376	7,65	1378,14	0,72	1437,6
13	229,52	4,46	14376	7,62	1368,66	0,72	1437,6
13,2	227,66	4,42	14376	7,57	1357,69	0,72	1437,6
13,4	221,95	4,37	14376	7,25	1302,58	0,71	1437,6
13,6	228,11	4,37	14376	7,39	1324,39	0,72	1437,6
13,8	226,7	4,34	14376	7,41	1325,97	0,72	1437,6
14	229,11	4,31	14376	7,36	1316,34	0,72	1437,6
14,2	231,31	4,3	14376	7,54	1344,04	0,73	1437,6
14,4	232,94	4,28	14376	7,63	1355,42	0,73	1437,6
14,6	236,6	4,39	14376	8	1422,33	0,73	1437,6
14,8	236,7	4,46	14375	8,18	1456,59	0,74	1437,5
15	236,21	4,46	14376	8,33	1479,99	0,74	1437,6
15,2	234,38	4,5	14376	8,25	1470,2	0,74	1437,6
15,4	233,58	4,54	14377	8,26	1474,45	0,73	1437,7
15,6	230,98	4,51	14377	7,72	1386,89	0,72	1437,7
15,8	229,4	4,46	14377	7,8	1396,88	0,73	1437,7
16	226,69	4,42	14378	7,55	1354,13	0,72	1437,8
16,2	223,9	4,37	14378	7,23	1300,49	0,71	1437,8
16,4	222,14	4,34	14379	7,19	1291,99	0,71	1437,9
16,6	221,72	4,29	14378	6,66	1204,77	0,7	1437,8
16,8	219,86	4,09	14379	6,56	1174,05	0,7	1437,9
17	220,87	4,16	14378	6,28	1134,68	0,69	1437,8
17,2	222,4	4,08	14378	6,02	1088,41	0,68	1437,8
17,4	227,5	4	14379	6,22	1114,58	0,7	1437,9
17,6	230,21	4,03	14379	6,12	1101,3	0,69	1437,9
17,8	240,1	4,06	14379	6,57	1174,3	0,71	1437,9
18	243,63	4,13	14378	6,91	1231,1	0,71	1437,8
18,2	245,43	4,18	14378	7,09	1263,74	0,72	1437,8
18,4	243,62	4,23	14378	7,35	1308,22	0,72	1437,8
18,6	242,6	4,24	14379	7,5	1332,04	0,73	1437,9
18,8	243,13	4,31	14379	7,72	1373,05	0,73	1437,9
19	245,42	4,29	14379	7,72	1370,89	0,73	1437,9

19,2	248,67	4,13	14379	7,48	1321,23	0,73	1437,9
19,4	245,12	4,25	14379	7,79	1379,23	0,73	1437,9
19,6	239,21	4,24	14379	7,8	1379,23	0,73	1437,9
19,8	238,85	4,22	14379	7,6	1346,99	0,73	1437,9



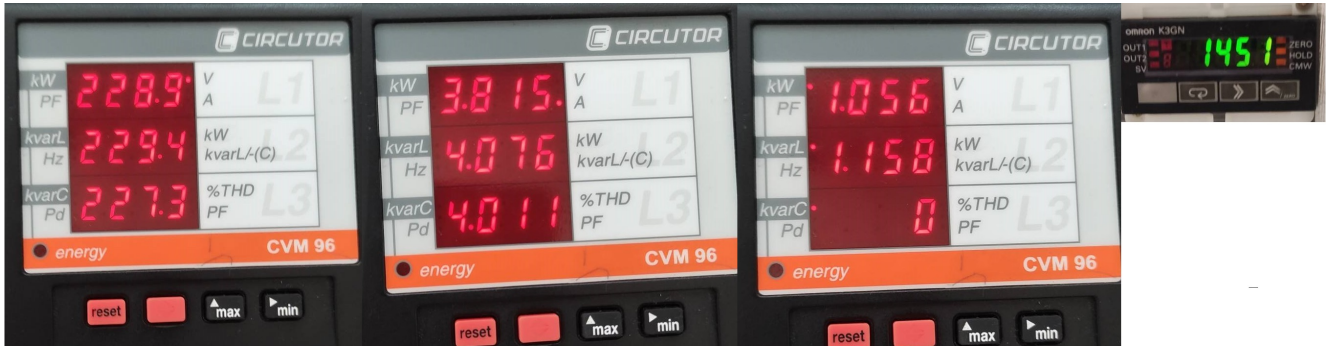
Mediciones a 1450 rpm

Promedios:	227,8958	3,9813	14505,31	6,1747	1106,4444	0,6993	
Tiempo (ms)	Tensión (V)	Corriente (A)	Velocidad*10 (rpm)	Par (Nm)	Potencia (W)	Rendimiento (%)	Velocidad (rpm)
0	224,12	3,74	14501	5,08	917,77	0,66	1450,1
0,2	227	3,68	14502	5,03	905,89	0,66	1450,2
0,4	232	3,75	14502	5,24	944,75	0,67	1450,2
0,6	237,73	3,73	14502	5,48	980,18	0,68	1450,2
0,8	243,1	3,73	14502	5,61	1000,99	0,69	1450,2
1	246,98	4,18	14501	7,25	1288,18	0,73	1450,1
1,2	247,28	3,9	14501	6,21	1106,35	0,71	1450,1
1,4	245,77	4,02	14501	6,67	1185,92	0,72	1450,1
1,6	245,27	3,96	14501	6,54	1161,23	0,72	1450,1
1,8	246,13	3,94	14501	6,59	1169,17	0,72	1450,1
2	247,94	3,94	14501	6,6	1169,45	0,72	1450,1
2,2	250,22	3,96	14501	6,78	1199,09	0,72	1450,1
2,4	249,63	3,93	14501	6,76	1195,31	0,72	1450,1
2,6	243,48	3,92	14501	6,63	1172,76	0,72	1450,1
2,8	242,59	3,89	14501	6,65	1174,47	0,72	1450,1
3	239,87	3,93	14501	6,72	1188,31	0,72	1450,1
3,2	236,47	3,81	14501	6,51	1147,67	0,72	1450,1
3,4	236,06	3,84	14500	6,59	1161,07	0,72	1450
3,6	236,2	3,84	14500	6,68	1175,99	0,72	1450
3,8	236,8	3,9	14500	6,92	1218,25	0,73	1450
4	237,87	3,97	14499	7,11	1252,89	0,73	1449,9
4,2	234,99	4,11	14499	7,4	1306,66	0,74	1449,9
4,4	235,79	4,26	14499	7,73	1370,12	0,74	1449,9
4,6	233,12	4,16	14499	7,46	1319,78	0,73	1449,9
4,8	231,24	4,21	14499	7,32	1302,12	0,73	1449,9
5	227,89	4,26	14499	7,13	1277,02	0,72	1449,9
5,2	226,21	4,24	14500	6,98	1251,64	0,72	1450
5,4	225,03	4,34	14500	7,05	1270,32	0,72	1450
5,6	220,78	4,22	14501	6,55	1181,56	0,7	1450,1
5,8	223,13	4,17	14501	6,51	1172,16	0,71	1450,1

6	220,15	4,21	14501	6,22	1129,59	0,69	1450,1
6,2	218,85	4,1	14503	6,02	1090,23	0,69	1450,3
6,4	218,91	4	14503	5,77	1043,6	0,68	1450,3
6,6	219,26	4,02	14504	5,73	1039,58	0,68	1450,4
6,8	220,99	3,92	14504	5,64	1017,96	0,68	1450,4
7	223,44	3,91	14505	5,75	1034,79	0,69	1450,5
7,2	226,38	3,92	14505	5,9	1059,79	0,69	1450,5
7,4	227,95	3,77	14505	5,72	1020,57	0,69	1450,5
7,6	228,5	4,01	14505	6,14	1101,79	0,7	1450,5
7,8	226,44	4,09	14506	6,24	1124,9	0,7	1450,6
8	226,31	4,12	14506	6,26	1129,9	0,7	1450,6
8,2	224,15	4,11	14506	6,2	1119,73	0,7	1450,6
8,4	222,45	4,13	14506	6,08	1102,05	0,69	1450,6
8,6	222,67	4,01	14506	5,93	1069,19	0,69	1450,6
8,8	221,76	4,2	14507	5,86	1071,68	0,68	1450,7
9	218,53	4,07	14507	5,57	1016,71	0,67	1450,7
9,2	220,81	4,01	14507	5,42	989,76	0,67	1450,7
9,4	218,15	3,97	14507	5,22	956,37	0,66	1450,7
9,6	218,14	3,88	14507	5,19	945,51	0,66	1450,7
9,8	216,63	3,78	14507	5,07	918,47	0,66	1450,7
10	218,85	3,82	14507	4,96	904,37	0,65	1450,7
10,2	218,91	3,74	14507	5,08	917,97	0,66	1450,7
10,4	219,63	3,68	14507	5,18	929,45	0,67	1450,7
10,6	221,23	3,66	14507	5,27	942,51	0,68	1450,7
10,8	222,35	3,65	14507	5,23	935,85	0,68	1450,7
11	220,82	3,74	14507	5,39	966,33	0,68	1450,7
11,2	220,54	3,83	14506	5,86	1046,96	0,7	1450,6
11,4	218,93	3,88	14507	5,9	1055,76	0,69	1450,7
11,6	217,97	3,92	14506	5,92	1062,18	0,69	1450,6
11,8	216,94	3,91	14507	5,86	1052,33	0,69	1450,7
12	217,6	3,91	14507	5,87	1053,8	0,69	1450,7
12,2	217,97	3,89	14507	5,77	1036,82	0,69	1450,7
12,4	217,52	3,83	14507	5,71	1022,19	0,69	1450,7

12,6	217,52	3,83	14507	5,67	1017,35	0,69	1450,7
12,8	217,08	3,81	14507	5,63	1009,5	0,69	1450,7
13	217,79	3,78	14507	5,65	1009,93	0,69	1450,7
13,2	218,49	3,77	14507	5,7	1016,78	0,69	1450,7
13,4	219,62	3,81	14508	5,86	1045,9	0,7	1450,8
13,6	221,96	3,79	14508	5,91	1052,8	0,7	1450,8
13,8	225,16	3,8	14507	6,02	1069,17	0,7	1450,7
14	227,97	3,89	14507	6,23	1109,08	0,71	1450,7
14,2	229,97	4,07	14507	6,73	1199,38	0,72	1450,7
14,4	231,71	4,17	14507	6,97	1244,97	0,72	1450,7
14,6	232,87	4,23	14507	7,06	1262,38	0,72	1450,7
14,8	231,92	4,23	14507	6,89	1235,73	0,72	1450,7
15	232,18	4,26	14507	6,92	1242,79	0,71	1450,7
15,2	231,68	4,32	14507	6,95	1252,41	0,71	1450,7
15,4	231,85	4,27	14507	6,74	1216,06	0,71	1450,7
15,6	229,91	4,28	14508	6,55	1185,5	0,7	1450,8
15,8	228,1	4,16	14508	6,28	1135,43	0,7	1450,8
16	226,98	4,11	14508	6,21	1120,98	0,7	1450,8
16,2	227,02	4,14	14508	6,19	1118,91	0,7	1450,8
16,4	224,8	4	14508	6	1080,93	0,69	1450,8
16,6	226,13	3,92	14509	5,92	1062,38	0,69	1450,9
16,8	226,79	3,95	14509	6	1076,14	0,7	1450,9
17	227,54	3,85	14509	6,12	1088,95	0,71	1450,9
17,2	229,83	3,93	14509	6,26	1116,57	0,71	1450,9
17,4	232,74	3,92	14509	6,64	1175,48	0,72	1450,9
17,6	232,53	4,05	14509	6,64	1183,72	0,71	1450,9
17,8	232,08	4,08	14509	6,75	1203,74	0,72	1450,9
18	230,96	4,14	14509	6,79	1213,96	0,72	1450,9
18,2	230,38	4,22	14510	6,81	1221,87	0,71	1451
18,4	231,97	4,16	14510	6,86	1227,02	0,72	1451
18,6	227,89	4,16	14510	6,63	1190,07	0,71	1451
18,8	226,05	4,14	14510	6,35	1144,14	0,7	1451
19	224,06	4,05	14511	6,09	1098,15	0,7	1451,1

19,2	222,86	4	14511	5,86	1058,53	0,69	1451,1
19,4	220,08	3,94	14512	5,57	1009,3	0,68	1451,2
19,6	220,87	3,87	14512	5,26	955,13	0,67	1451,2
19,8	221,82	3,81	14513	5,05	917,55	0,66	1451,3



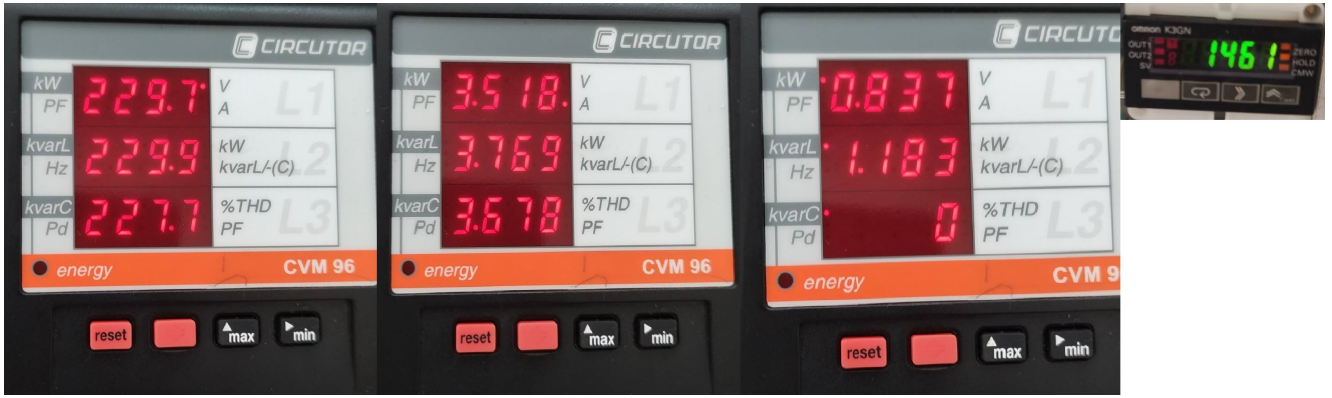
Mediciones a 1460 rpm

Promedios:	228,4195	3,7317	14605,36	5,1447	928,1991	0,6691	
Tiempo (ms)	Tensión (V)	Corriente (A)	Velocidad*10 (rpm)	Par (Nm)	Potencia (W)	Rendimiento (%)	Velocidad (rpm)
0	223	3,75	14603	4,9	891,04	0,66	1460,3
0,2	220,12	3,73	14603	4,59	841,22	0,64	1460,3
0,4	222,21	3,53	14603	4,37	794,36	0,64	1460,3
0,6	222,86	3,58	14604	4,11	755,98	0,62	1460,4
0,8	226	3,51	14604	3,97	729,88	0,61	1460,4
1	228,72	3,54	14604	4,27	778,27	0,63	1460,4
1,2	234,15	3,43	14604	3,91	715,92	0,61	1460,4
1,4	241,56	3,51	14604	4,48	809,04	0,64	1460,4
1,6	245,63	3,56	14604	4,76	857,28	0,66	1460,4
1,8	247,81	3,67	14604	4,78	866,96	0,66	1460,4
2	248,14	3,75	14604	5,09	920,89	0,67	1460,4
2,2	246,91	3,68	14604	5,34	955,08	0,68	1460,4
2,4	247,22	3,7	14604	5,43	970,58	0,69	1460,4
2,6	249,13	3,71	14604	5,52	986,01	0,69	1460,4
2,8	250,46	3,7	14604	5,59	995,53	0,69	1460,4
3	250,86	3,67	14605	5,61	998,13	0,7	1460,5
3,2	247,51	3,66	14605	5,87	1037,88	0,71	1460,5
3,4	244,97	3,62	14605	5,49	974,87	0,69	1460,5
3,6	242,11	3,6	14605	5,5	976,05	0,7	1460,5
3,8	240,66	3,58	14605	5,49	972,38	0,7	1460,5
4	238,49	3,57	14605	5,54	980,32	0,7	1460,5
4,2	236,96	3,56	14606	5,5	972,58	0,7	1460,6
4,4	237,07	3,6	14606	5,78	1019,36	0,71	1460,6
4,6	238,58	3,67	14606	5,95	1050,06	0,71	1460,6
4,8	238,38	3,68	14606	6,15	1081,77	0,72	1460,6
5	236,7	3,96	14606	6,65	1179,76	0,72	1460,6
5,2	235,73	3,98	14606	6,66	1183,01	0,72	1460,6
5,4	234,08	4,01	14606	6,61	1176,72	0,72	1460,6
5,6	229,01	3,99	14606	6,2	1111,31	0,71	1460,6
5,8	226,76	4	14606	5,98	1077,09	0,7	1460,6

6	226,19	4,05	14607	6,01	1084,73	0,7	1460,7
6,2	225,25	4,01	14607	5,8	1048,83	0,69	1460,7
6,4	221,91	3,99	14607	5,57	1012	0,68	1460,7
6,6	220,98	3,94	14608	5,31	967,43	0,67	1460,8
6,8	220,31	3,83	14608	5,1	927,38	0,67	1460,8
7	219,24	3,83	14608	4,89	893,94	0,65	1460,8
7,2	219,41	3,78	14608	4,73	865,42	0,65	1460,8
7,4	216,94	3,71	14608	4,49	823,46	0,64	1460,8
7,6	223,15	3,69	14609	4,6	839,11	0,64	1460,9
7,8	226,62	3,68	14609	4,67	849,92	0,65	1460,9
8	227,47	3,72	14609	4,88	885,9	0,66	1460,9
8,2	227,48	4,14	14609	4,21	808,89	0,59	1460,9
8,4	229,64	3,81	14609	5,12	929,7	0,67	1460,9
8,6	228,35	3,87	14609	5,21	947,05	0,67	1460,9
8,8	227,45	3,81	14609	5,29	955,69	0,68	1460,9
9	226,55	3,89	14609	5,16	940,96	0,67	1460,9
9,2	225,1	3,88	14609	4,96	908,48	0,66	1460,9
9,4	222,75	3,86	14610	5,33	965,06	0,68	1461
9,6	221,88	4,08	14610	4,7	881,4	0,63	1461
9,8	219,95	3,81	14610	4,54	837,43	0,63	1461
10	217,21	3,75	14610	4,31	798,01	0,62	1461
10,2	218,49	3,65	14610	4,21	775,74	0,62	1461
10,4	217,89	3,62	14610	3,81	711,21	0,59	1461
10,6	218,98	3,43	14610	4,12	748,4	0,62	1461
10,8	219,61	3,42	14610	4,13	748,89	0,63	1461
11	219,62	3,5	14609	4,01	735,08	0,61	1460,9
11,2	221,96	3,44	14609	4,33	781,86	0,64	1460,9
11,4	222,48	3,49	14608	4,37	791,04	0,64	1460,8
11,6	222,68	3,46	14608	4,68	838,09	0,66	1460,8
11,8	221,06	3,41	14608	4,82	857,75	0,67	1460,8
12	221,28	3,54	14607	4,82	864,65	0,66	1460,7
12,2	219,62	3,53	14607	4,52	817,24	0,65	1460,7
12,4	219,27	3,65	14607	4,84	874,22	0,66	1460,7

12,6	219,38	3,68	14607	5,02	905,47	0,67	1460,7
12,8	218,76	3,61	14606	4,65	843	0,65	1460,6
13	218,35	3,66	14606	4,86	879,21	0,66	1460,6
13,2	216,99	3,63	14606	4,73	855,94	0,65	1460,6
13,4	217,22	3,75	14606	5,08	919,23	0,67	1460,6
13,6	217,39	3,57	14606	4,72	851,6	0,66	1460,6
13,8	217,79	3,52	14606	4,64	836,01	0,65	1460,6
14	219,92	3,52	14605	4,91	877,79	0,67	1460,5
14,2	220,69	3,51	14605	4,79	859,03	0,66	1460,5
14,4	222,93	3,53	14605	4,96	887,13	0,67	1460,5
14,6	225,6	3,61	14604	5,16	921,94	0,68	1460,4
14,8	228,16	3,69	14604	5,49	980,11	0,69	1460,4
15	229,57	3,79	14604	5,76	1029,09	0,7	1460,4
15,2	232,29	3,91	14604	6,02	1077,03	0,7	1460,4
15,4	233,39	3,93	14603	5,91	1061,85	0,7	1460,3
15,6	236,5	3,91	14603	5,84	1048,83	0,7	1460,3
15,8	231,56	3,95	14603	5,78	1042,25	0,69	1460,3
16	231,18	3,99	14602	5,74	1038,21	0,69	1460,2
16,2	229,69	4	14602	5,63	1022,67	0,68	1460,2
16,4	228,88	3,97	14602	5,49	998,55	0,68	1460,2
16,6	229,12	3,9	14602	5,27	958,16	0,67	1460,2
16,8	226,33	3,84	14602	5,2	943,92	0,67	1460,2
17	225,5	3,78	14602	5,04	914,02	0,66	1460,2
17,2	225,01	3,75	14602	5,01	907,62	0,66	1460,2
17,4	225,1	3,69	14602	4,96	895,88	0,66	1460,2
17,6	230,83	3,62	14602	5,16	922,55	0,68	1460,2
17,8	227,5	3,7	14601	5,19	932,95	0,68	1460,1
18	229,36	3,71	14601	5,36	959,34	0,68	1460,1
18,2	231,91	3,78	14601	5,66	1012	0,69	1460,1
18,4	233,2	3,82	14601	5,77	1032,21	0,7	1460,1
18,6	228,93	3,87	14601	5,73	1028,51	0,69	1460,1
18,8	231,96	3,87	14601	5,88	1053,06	0,7	1460,1
19	229,95	3,85	14601	5,72	1025,55	0,69	1460,1

19,2	228,54	3,88	14601	5,74	1031,35	0,69	1460,1
19,4	226,35	3,88	14601	5,57	1003,98	0,69	1460,1
19,6	224,62	3,91	14600	5,33	967,99	0,67	1460
19,8	222,94	3,82	14600	5,07	921,59	0,66	1460



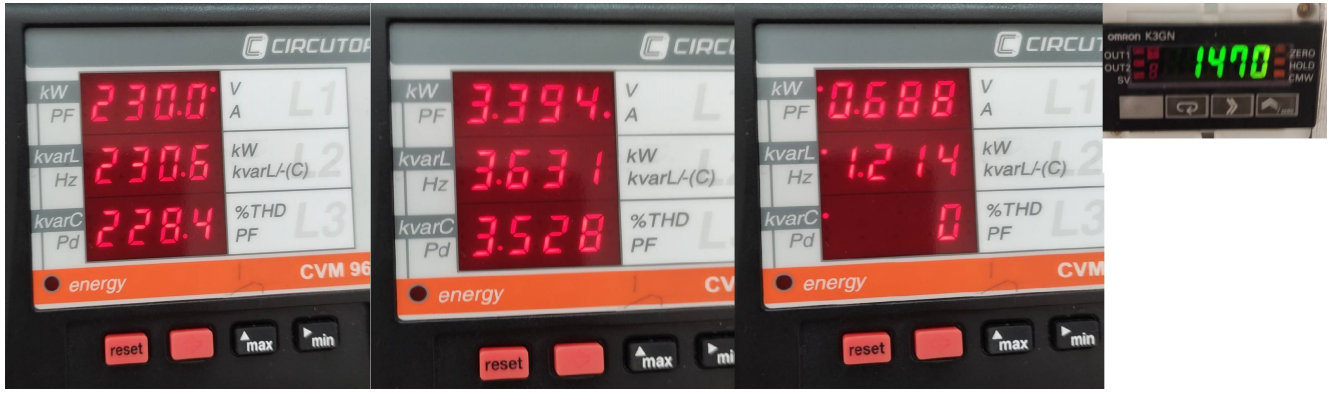
Mediciones a 1470 rpm

Promedios:	228,8655	3,552	14711,98	4,0167	739,7038	0,6069	
Tiempo (ms)	Tensión (V)	Corriente (A)	Velocidad*10 (rpm)	Par (Nm)	Potencia (W)	Rendimiento (%)	Velocidad (rpm)
0	226,88	3,81	14704	5,01	912,6	0,67	1470,4
0,2	224,68	3,91	14705	5,08	928,76	0,66	1470,5
0,4	222,66	3,78	14705	4,54	835,47	0,64	1470,5
0,6	221,51	3,74	14706	4,36	805,71	0,63	1470,6
0,8	220,42	3,7	14706	4,06	754,61	0,61	1470,6
1	217,82	3,65	14707	3,78	707,76	0,59	1470,7
1,2	221,15	3,77	14707	3,67	699,65	0,57	1470,7
1,4	223,11	3,5	14706	3,61	672,68	0,58	1470,6
1,6	225,21	3,5	14707	3,59	669,67	0,58	1470,7
1,8	229	3,53	14708	3,7	688,55	0,59	1470,8
2	231,15	3,48	14708	3,85	709,18	0,6	1470,8
2,2	230,38	3,72	14708	4,25	785,96	0,62	1470,8
2,4	230,6	3,82	14708	4,4	817,07	0,63	1470,8
2,6	230,4	3,66	14709	4,23	779,28	0,62	1470,9
2,8	228,96	3,68	14709	4,1	760,63	0,61	1470,9
3	227,89	3,7	14710	3,99	744,52	0,61	1471
3,2	225,13	4	14710	2,51	532,2	0,42	1471
3,4	224,41	3,68	14711	3,82	716,69	0,59	1471,1
3,6	222,3	3,64	14711	3,62	681,93	0,58	1471,1
3,8	220,5	3,67	14712	3,37	644,86	0,55	1471,2
4	219,93	3,57	14709	3,29	627,02	0,55	1470,9
4,2	219,37	3,5	14712	3,18	605,03	0,54	1471,2
4,4	219,89	3,42	14713	3,02	575,71	0,52	1471,3
4,6	219,86	3,36	14713	2,83	541,09	0,5	1471,3
4,8	221,74	3,34	14713	2,87	546,95	0,51	1471,3
5	222,17	3,31	14714	3,17	593,01	0,55	1471,4
5,2	223,5	3,35	14714	3,27	609,75	0,56	1471,4
5,4	224,2	3,31	14714	3,53	648,33	0,59	1471,4
5,6	223,61	3,36	14714	3,63	667,67	0,59	1471,4
5,8	220,84	3,34	14714	3,36	623,11	0,57	1471,4

6	220,1	3,43	14714	3,71	683,53	0,59	1471,4
6,2	218,44	3,43	14714	3,75	690,02	0,6	1471,4
6,4	218,6	3,45	14715	3,71	684,89	0,59	1471,5
6,6	218,96	3,41	14714	3,58	662,13	0,58	1471,4
6,8	218,53	3,43	14715	3,75	689,42	0,6	1471,5
7	218,57	3,4	14715	3,63	670,42	0,59	1471,5
7,2	217,58	3,31	14715	3,44	635,06	0,58	1471,5
7,4	217,62	3,36	14715	3,56	656,93	0,59	1471,5
7,6	218,12	3,32	14715	3,51	646,19	0,58	1471,5
7,8	218,73	3,3	14715	3,53	648,94	0,59	1471,5
8	217,61	3,29	14716	3,51	643,85	0,58	1471,6
8,2	221,57	3,29	14716	3,74	679,99	0,61	1471,6
8,4	223,92	3,34	14716	3,96	717,56	0,62	1471,6
8,6	227,28	3,41	14716	4,18	756,24	0,63	1471,6
8,8	228,02	3,54	14716	4,56	824,84	0,65	1471,6
9	230,1	3,65	14716	4,86	877,96	0,67	1471,6
9,2	231,41	3,69	14716	4,76	865,16	0,66	1471,6
9,4	232,5	3,74	14716	4,7	858,71	0,65	1471,6
9,6	230,76	3,76	14716	4,64	850,23	0,65	1471,6
9,8	232,76	3,84	14716	4,65	857,52	0,64	1471,6
10	233,22	3,79	14716	4,54	837,08	0,64	1471,6
10,2	227,74	3,77	14717	4,28	794,18	0,62	1471,7
10,4	228,91	3,74	14717	4,21	782,26	0,62	1471,7
10,6	226,92	3,71	14716	4,1	762,77	0,61	1471,6
10,8	225,9	3,6	14717	3,92	727,79	0,6	1471,7
11	225,07	3,5	14717	3,62	674,33	0,58	1471,7
11,2	222,75	3,68	14717	3,64	687,56	0,58	1471,7
11,4	225,7	3,57	14716	3,79	704,63	0,6	1471,6
11,6	227,04	3,48	14716	3,92	720,36	0,61	1471,6
11,8	228,96	3,45	14716	4,05	739,14	0,62	1471,6
12	230,59	3,42	14716	4,47	802,85	0,65	1471,6
12,2	239,66	3,64	14716	4,27	785,64	0,63	1471,6
12,4	229,7	3,65	14716	4,38	802,53	0,64	1471,6

12,6	229,7	3,64	14715	4,52	823,59	0,65	1471,5
12,8	226,93	3,67	14715	4,55	831,1	0,65	1471,5
13	230,58	3,66	14715	4,56	830,96	0,65	1471,5
13,2	226,59	3,7	14715	4,42	812,21	0,64	1471,5
13,4	227,1	3,71	14715	4,28	791,41	0,63	1471,5
13,6	224,14	3,65	14715	4,13	764,13	0,62	1471,5
13,8	223,3	3,62	14715	3,88	722,83	0,6	1471,5
14	221,96	3,59	14715	3,59	674,36	0,58	1471,5
14,2	220,89	3,53	14714	3,29	624,16	0,55	1471,4
14,4	222,23	3,51	14714	3,06	585,66	0,52	1471,4
14,6	223,56	3,37	14714	2,95	561,47	0,52	1471,4
14,8	226,31	3,39	14714	2,92	557,39	0,51	1471,4
15	231,7	3,31	14713	2,97	561,28	0,52	1471,3
15,2	236,96	3,36	14713	3	568,27	0,52	1471,3
15,4	242,74	3,41	14712	3,45	641,37	0,57	1471,2
15,6	244,85	3,43	14712	3,56	661,01	0,58	1471,2
15,8	247,03	3,48	14712	3,75	692,59	0,6	1471,2
16	249,31	3,43	14711	4,1	744,77	0,63	1471,1
16,2	249,08	3,53	14711	4,06	744,24	0,62	1471,1
16,4	249,47	3,58	14710	3,8	706,51	0,6	1471
16,6	250,33	3,51	14710	4,2	766,06	0,63	1471
16,8	250,6	3,5	14710	4,05	741,81	0,62	1471
17	248,87	3,47	14710	4,22	765,94	0,63	1471
17,2	246,23	3,45	14709	3,97	726,83	0,62	1470,9
17,4	244,87	3,68	14709	7,43	1283,25	0,76	1470,9
17,6	242,07	3,39	14708	4,25	765,65	0,64	1470,8
17,8	240,64	3,33	14708	4,08	736,49	0,63	1470,8
18	238,93	3,35	14708	4,02	727,17	0,62	1470,8
18,2	239,9	3,37	14707	4,5	804,39	0,66	1470,7
18,4	239,3	3,43	14707	4,67	834,03	0,66	1470,7
18,6	240,06	3,51	14706	5	891,15	0,68	1470,6
18,8	237,38	3,64	14706	5,37	958,14	0,69	1470,6
19	237,15	3,71	14705	5,55	990,05	0,7	1470,5

19,2	235,68	3,77	14705	5,53	991,33	0,69	1470,5
19,4	231,38	3,78	14705	5,27	949,72	0,68	1470,5
19,6	229,52	3,77	14704	5,16	932,18	0,68	1470,4
19,8	226,6	3,78	14704	4,95	900,72	0,66	1470,4



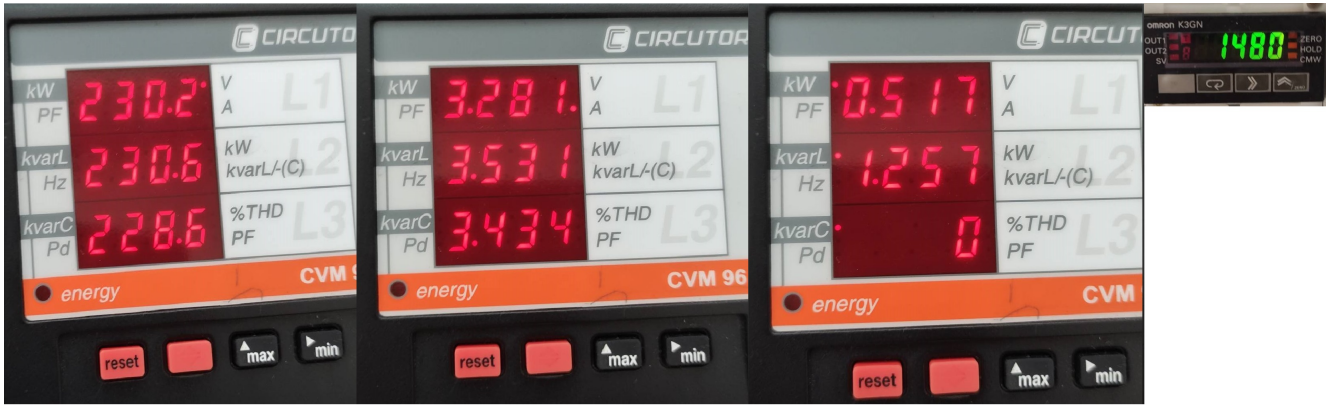
Mediciones a 1480 rpm

Promedios:	229,2935	3,4534	14803,09	3,0678	584,7026	0,5177	
Tiempo (ms)	Tensión (V)	Corriente (A)	Velocidad*10 (rpm)	Par (Nm)	Potencia (W)	Rendimiento (%)	Velocidad (rpm)
0	218,07	3,21	14811	2,45	473,12	0,45	1481,1
0,2	218,96	3,22	14812	2,53	487,02	0,47	1481,2
0,4	217,81	3,26	14812	2,58	496,99	0,47	1481,2
0,6	220,77	3,2	14811	2,73	516,98	0,5	1481,1
0,8	223,7	3,22	14811	2,84	535,67	0,51	1481,1
1	225,19	3,28	14811	3,03	569,07	0,54	1481,1
1,2	227,45	3,39	14810	3,39	630,61	0,57	1481
1,4	228,75	3,46	14810	3,72	687,83	0,6	1481
1,6	230,97	3,56	14810	3,92	724,98	0,61	1481
1,8	233,13	3,61	14809	3,83	713,91	0,6	1480,9
2	232,04	3,74	14809	3,91	734,11	0,6	1480,9
2,2	231,24	3,65	14809	3,6	680,03	0,58	1480,9
2,4	232,69	3,7	14808	3,66	692,95	0,58	1480,8
2,6	228,15	3,7	14808	3,58	680,29	0,57	1480,8
2,8	228,55	3,67	14808	3,29	632,15	0,54	1480,8
3	227,47	3,64	14808	3,16	610,57	0,53	1480,8
3,2	224,41	3,61	14807	3,05	591,24	0,52	1480,7
3,4	225,33	3,7	14807	3,3	636,75	0,54	1480,7
3,6	224,58	3,53	14807	2,93	566,91	0,51	1480,7
3,8	225,97	3,48	14807	2,87	555,68	0,5	1480,7
4	227,74	3,43	14806	2,9	556,47	0,51	1480,6
4,2	227,57	3,44	14806	3,04	579,43	0,53	1480,6
4,4	229,76	3,37	14806	3,11	585,69	0,54	1480,6
4,6	231,02	3,64	14805	3,68	692,92	0,59	1480,5
4,8	230,39	3,47	14805	3,57	665,08	0,58	1480,5
5	229,35	3,52	14805	3,79	701,2	0,6	1480,5
5,2	227,72	3,56	14805	3,68	687,29	0,59	1480,5
5,4	227,84	3,61	14805	3,65	686,09	0,58	1480,5
5,6	227,32	3,65	14804	3,51	665,4	0,57	1480,4
5,8	227,04	3,62	14804	3,5	663,1	0,57	1480,4

6	224,95	3,59	14804	3,26	623,34	0,55	1480,4
6,2	223,36	3,66	14804	3,15	610,53	0,53	1480,4
6,4	221,88	3,58	14804	2,84	556,39	0,49	1480,4
6,6	221,29	3,5	14804	2,54	503,48	0,45	1480,4
6,8	221,46	3,44	14803	2,27	458,71	0,41	1480,3
7	222,93	3,42	14803	2,04	421,44	0,36	1480,3
7,2	226,36	3,31	14803	1,9	391,84	0,33	1480,3
7,4	230,01	3,06	14803	2,18	423,19	0,41	1480,3
7,6	233,57	3,29	14802	2,01	408,81	0,36	1480,2
7,8	238,86	3,32	14802	2,13	428,59	0,38	1480,2
8	244,58	3,46	14802	2,12	436,6	0,38	1480,2
8,2	246,23	3,3	14802	2,09	422,71	0,38	1480,2
8,4	248,6	3,36	14801	2,87	548,11	0,51	1480,1
8,6	250,75	3,42	14801	2,96	565,25	0,52	1480,1
8,8	250,9	3,4	14801	3,12	588,69	0,54	1480,1
9	251,33	3,39	14801	3,04	575,76	0,53	1480,1
9,2	250,38	3,4	14801	3,05	577,87	0,53	1480,1
9,4	250,52	3,39	14801	3,23	605,43	0,55	1480,1
9,6	250,1	3,35	14801	3,36	624,87	0,57	1480,1
9,8	245,84	3,3	14801	3,17	591,89	0,55	1480,1
10	245,42	3,3	14800	3,35	619,77	0,57	1480
10,2	242,76	3,28	14801	2,71	517,62	0,49	1480,1
10,4	240,51	3,25	14801	3,24	600,41	0,56	1480,1
10,6	240,55	3,23	14800	3,29	607,01	0,57	1480
10,8	241,55	3,26	14801	3,66	665,67	0,6	1480,1
11	241,93	3,32	14801	3,92	710,86	0,62	1480,1
11,2	241,14	3,4	14800	4,21	760,86	0,64	1480
11,4	239,38	3,46	14800	4,42	797,77	0,65	1480
11,6	238,28	3,6	14800	4,77	860,06	0,67	1480
11,8	235,48	3,62	14800	4,48	816,96	0,65	1480
12	231,92	3,67	14800	4,33	795,92	0,64	1480
12,2	229,33	3,85	14800	4,31	804,51	0,62	1480
12,4	227,79	3,67	14799	4,09	758,53	0,62	1479,9

12,6	226,08	3,72	14799	3,94	738,59	0,6	1479,9
12,8	223,89	3,7	14799	3,73	703,87	0,59	1479,9
13	222,38	3,63	14798	3,4	647,17	0,56	1479,8
13,2	221,48	3,66	14798	3,4	649,96	0,56	1479,8
13,4	221,75	3,57	14798	2,99	579,69	0,52	1479,8
13,6	220,46	3,52	14798	2,72	534,32	0,48	1479,8
13,8	225,14	3,39	14798	2,47	486,01	0,45	1479,8
14	225,48	3,52	14798	2,48	496	0,44	1479,8
14,2	228,35	3,23	14798	2,44	472,82	0,45	1479,8
14,4	231,43	3,42	14799	2,76	533,32	0,49	1479,9
14,6	234,75	3,45	14799	2,98	570,8	0,52	1479,9
14,8	233,89	3,55	14800	2,9	564,49	0,5	1480
15	230,97	3,53	14799	3,25	616,9	0,55	1479,9
15,2	230,48	3,59	14799	3,22	616,96	0,54	1479,9
15,4	231,73	3,58	14799	3,31	630,4	0,55	1479,9
15,6	217,98	3,61	14800	3,12	601,72	0,53	1480
15,8	225,77	3,64	14800	2,9	570,07	0,5	1480
16	223,86	3,57	14800	3	579,89	0,52	1480
16,2	222,67	3,54	14801	2,63	520,38	0,46	1480,1
16,4	220,7	3,49	14800	2,4	480,89	0,43	1480
16,6	219,88	3,43	14801	2,27	458,64	0,41	1480,1
16,8	220,08	3,41	14801	2,08	426,38	0,37	1480,1
17	220,41	3,43	14802	2,12	434,25	0,38	1480,2
17,2	220,5	3,28	14802	2,09	420,45	0,38	1480,2
17,4	221,23	3,25	14802	2,02	408,31	0,36	1480,2
17,6	222,8	3,21	14803	2,46	474,61	0,45	1480,3
17,8	224,16	3,24	14803	2,09	418,11	0,38	1480,3
18	223,98	3,37	14803	2,6	505,96	0,47	1480,3
18,2	223,69	3,28	14803	2,76	526,24	0,5	1480,3
18,4	220,4	3,31	14803	2,84	540,86	0,51	1480,3
18,6	219,74	3,33	14803	2,89	549,33	0,51	1480,3
18,8	219,46	3,34	14803	2,95	559,35	0,52	1480,3
19	219,38	3,35	14803	3,18	595,47	0,55	1480,3

19,2	219,61	3,36	14804	2,91	554,4	0,52	1480,4
19,4	218,05	3,32	14803	2,92	553,3	0,52	1480,3
19,6	217,51	3,28	14803	2,85	540,85	0,51	1480,3
19,8	218,31	3,25	14804	2,8	530,52	0,51	1480,4



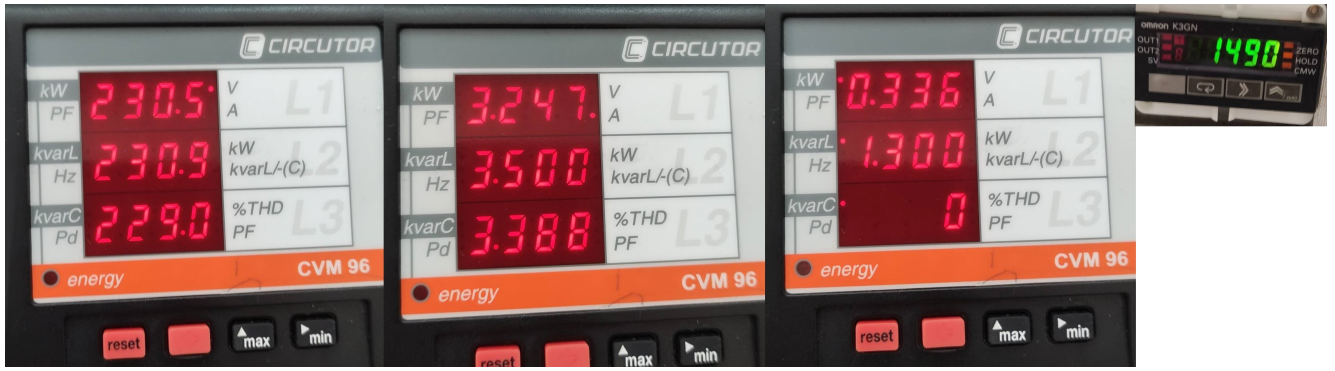
Mediciones a 1490 rpm

Promedios:	229,4889	3,3885	14889,15	2,16	438,2278	0,3555	
Tiempo (ms)	Tensión (V)	Corriente (A)	Velocidad*10 (rpm)	Par (Nm)	Potencia (W)	Rendimiento (%)	Velocidad (rpm)
0	226,16	3,53	14901	1,83	393,91	0,3	1490,1
0,2	224,59	3,57	14901	1,67	372,71	0,25	1490,1
0,4	223,29	3,42	14901	1,55	343,93	0,22	1490,1
0,6	224,7	3,51	14901	1,92	407,66	0,33	1490,1
0,8	226,66	3,32	14901	1,91	394,2	0,33	1490,1
1	227,86	3,4	14900	2,04	420,08	0,36	1490
1,2	224,39	3,43	14900	2,09	428,76	0,37	1490
1,4	229,58	3,46	14900	2,45	487,36	0,44	1490
1,6	228,37	3,44	14900	2,66	519,48	0,48	1490
1,8	227,45	3,5	14900	2,63	518,28	0,47	1490
2	225,27	3,52	14899	2,52	502,85	0,45	1489,9
2,2	226,68	3,54	14899	2,51	502,25	0,45	1489,9
2,4	226,9	3,5	14899	2,41	482,96	0,43	1489,9
2,6	224,79	3,58	14899	2,22	458,22	0,39	1489,9
2,8	224,19	3,57	14899	1,96	417,31	0,33	1489,9
3	222,24	3,54	14899	1,85	397,83	0,31	1489,9
3,2	222,51	3,61	14898	0,92	257,31	-0,09	1489,8
3,4	220,36	3,45	14898	1,33	311,62	0,13	1489,8
3,6	222,94	3,42	14898	1	258	-0,04	1489,8
3,8	225,04	3,36	14897	0,91	240,39	-0,1	1489,7
4	227,68	3,32	14897	0,88	233,14	-0,12	1489,7
4,2	230,51	3,28	14896	0,88	231,03	-0,13	1489,6
4,4	237,65	3,31	14896	1,16	276,65	0,05	1489,6
4,6	243,83	3,31	14895	1,51	330,73	0,21	1489,5
4,8	244,37	3,33	14894	1,51	332,23	0,21	1489,4
5	248,73	3,3	14894	1,29	295,76	0,11	1489,4
5,2	244,75	3,34	14893	1,5	332,19	0,2	1489,3
5,4	250,15	3,36	14892	1,77	374,8	0,29	1489,2
5,6	251,17	3,5	14891	2,69	527,37	0,48	1489,1
5,8	252,19	3,42	14891	2,15	438,96	0,39	1489,1

6	250,98	3,36	14891	2,09	424,78	0,37	1489,1
6,2	249,21	3,31	14890	2,05	416,58	0,37	1489
6,4	246,46	3,28	14890	2	406,19	0,36	1489
6,6	247,63	3,23	14889	1,95	395,72	0,35	1488,9
6,8	243,75	3,2	14889	2,02	405,1	0,37	1488,9
7	243,03	3,2	14889	2,13	423,02	0,39	1488,9
7,2	241,63	3,16	14889	2,32	450,76	0,43	1488,9
7,4	241,73	3,19	14889	2,54	486,46	0,47	1488,9
7,6	242,79	3,21	14889	2,77	522,92	0,51	1488,9
7,8	243,73	3,3	14889	3,21	598,14	0,56	1488,9
8	241,13	3,38	14888	3,51	649,13	0,59	1488,8
8,2	240,3	3,46	14888	3,72	688,26	0,6	1488,8
8,4	238,74	3,47	14888	3,58	666,86	0,59	1488,8
8,6	233,16	3,54	14888	3,49	655,5	0,58	1488,8
8,8	230,11	3,58	14888	3,33	632,61	0,56	1488,8
9	229,12	3,61	14887	3,14	605,16	0,53	1488,7
9,2	225,35	3,67	14887	3,07	599,03	0,52	1488,7
9,4	225,73	3,58	14887	2,79	548,87	0,49	1488,7
9,6	223,69	3,52	14887	2,52	502,99	0,45	1488,7
9,8	223,16	3,52	14887	2,43	488,84	0,44	1488,7
10	221,78	3,5	14888	2,16	444,8	0,38	1488,8
10,2	221,91	3,43	14888	1,99	413,07	0,35	1488,8
10,4	222,87	3,42	14887	1,76	377,84	0,29	1488,7
10,6	222,45	3,34	14887	1,79	377,82	0,3	1488,7
10,8	229,08	3,35	14887	1,79	378,4	0,3	1488,7
11	230,99	3,35	14887	1,9	394,58	0,33	1488,7
11,2	236,28	3,41	14886	2,05	422,45	0,36	1488,6
11,4	235,37	3,3	14886	2,06	416,76	0,37	1488,6
11,6	231,02	3,41	14886	2,39	475,88	0,43	1488,6
11,8	233,21	3,44	14886	2,32	466,54	0,42	1488,6
12	231,21	3,49	14885	2,39	480,22	0,43	1488,5
12,2	230,53	3,56	14885	2,33	474,24	0,41	1488,5
12,4	228,57	3,73	14885	2,26	474,97	0,39	1488,5

12,6	224,32	3,55	14885	2,06	431,19	0,36	1488,5
12,8	224,07	3,51	14885	1,86	398,61	0,31	1488,5
13	221,24	3,48	14885	1,72	374,54	0,27	1488,5
13,2	222,54	3,45	14885	1,43	327,37	0,17	1488,5
13,4	220,99	3,36	14884	1,48	330,22	0,2	1488,4
13,6	221,16	3,3	14884	1,45	321,22	0,19	1488,4
13,8	221,06	3,27	14884	1,26	289,85	0,1	1488,4
14	220,96	3,24	14883	1,33	299,15	0,14	1488,3
14,2	221,74	3,22	14883	1,59	338,01	0,24	1488,3
14,4	224,91	3,2	14882	1,65	346,9	0,26	1488,2
14,6	224,48	3,24	14882	1,96	398,85	0,35	1488,2
14,8	222,33	3,28	14882	1,94	397,56	0,34	1488,2
15	221,21	3,26	14882	2,1	421,35	0,38	1488,2
15,2	219,47	3,28	14881	2,14	427,69	0,39	1488,1
15,4	220,27	3,31	14882	2,13	428,92	0,39	1488,2
15,6	219,62	3,28	14882	2,2	438,09	0,4	1488,2
15,8	219,5	3,25	14881	2,09	419,43	0,38	1488,1
16	216,39	3,26	14881	1,99	404,52	0,36	1488,1
16,2	218,62	3,2	14882	2,03	406,8	0,37	1488,2
16,4	218,74	3,18	14882	1,97	396,03	0,35	1488,2
16,6	219,7	3,1	14882	1,79	364,23	0,31	1488,2
16,8	218,33	3,42	14882	2,62	512,13	0,47	1488,2
17	219,2	3,11	14883	1,9	381,08	0,34	1488,3
17,2	220,13	3,1	14883	2,02	400,29	0,37	1488,3
17,4	222	3,1	14883	2,11	414,65	0,39	1488,3
17,6	224,83	3,13	14883	2,31	447,02	0,43	1488,3
17,8	227,74	3,21	14883	2,64	503,3	0,49	1488,3
18	227,88	3,36	14884	3,16	593,29	0,55	1488,4
18,2	233,52	3,41	14884	3,21	603,83	0,55	1488,4
18,4	231,12	3,28	14885	2,88	544,82	0,52	1488,5
18,6	231,23	3,58	14884	3,07	592,53	0,53	1488,4
18,8	231,31	3,5	14885	2,99	575,09	0,52	1488,5
19	231	3,6	14886	2,95	575,46	0,51	1488,6

19,2	230,83	3,72	14886	3,21	623,25	0,54	1488,6
19,4	229,62	3,53	14887	2,57	509,93	0,46	1488,7
19,6	227,6	3,46	14888	2,32	467,18	0,42	1488,8
19,8	227,53	3,48	14889	2,3	465,94	0,41	1488,9



Mediciones a 1500 rpm

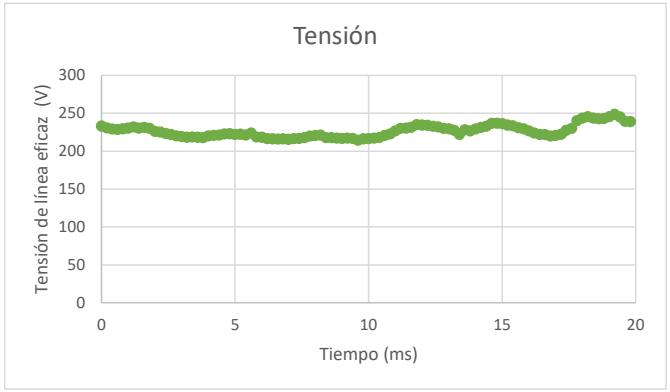
Promedios:	230,5332	3,4655	14984,34	1,0994	276,243	-0,2022	
Tiempo (ms)	Tensión (V)	Corriente (A)	Velocidad*10 (rpm)	Par (Nm)	Potencia (W)	Rendimiento (%)	Velocidad (rpm)
0	230,43	3,59	14997	1,52	349,91	0,2	1499,7
0,2	232,03	3,68	14997	1,62	371,45	0,23	1499,7
0,4	230,02	3,59	14997	1,03	272,16	-0,02	1499,7
0,6	229,96	3,72	14998	1,53	359,42	0,2	1499,8
0,8	228,46	3,69	14998	1,22	308,82	0,08	1499,8
1	228,33	3,67	14998	0,98	270,63	-0,05	1499,8
1,2	225,86	3,64	14998	0,95	263,47	-0,07	1499,8
1,4	225,58	3,48	14999	0,89	243,68	-0,12	1499,9
1,6	225,2	3,58	14998	0,75	228,74	-0,22	1499,8
1,8	225,47	3,53	14998	0,79	231,93	-0,19	1499,8
2	226,16	3,63	14998	1,21	302,91	0,07	1499,8
2,2	227,59	4,25	14998	1,83	442,39	0,27	1499,8
2,4	228,56	3,48	14998	1,16	286,47	0,05	1499,8
2,6	229,73	3,53	14998	1,41	328,95	0,16	1499,8
2,8	228,9	3,52	14998	1,5	341,55	0,2	1499,8
3	227,47	3,28	14998	2,19	435,68	0,4	1499,8
3,2	226,48	3,56	14998	1,54	351,21	0,21	1499,8
3,4	229,87	3,64	14997	1,65	373,85	0,25	1499,7
3,6	226,26	3,68	14997	1,54	357,57	0,21	1499,7
3,8	226,17	3,52	14997	1,36	320,14	0,14	1499,7
4	221,08	3,62	14997	1,31	318,55	0,12	1499,7
4,2	222,44	3,64	14997	0,86	248,51	-0,13	1499,7
4,4	221,96	3,57	14996	0,82	237,82	-0,17	1499,6
4,6	221,87	3,75	14996	-0,45	50,27	-4,74	1499,6
4,8	222,59	3,56	14996	-0,09	94,54	-1,92	1499,6
5	223,36	3,38	14995	0,04	103,86	-1,56	1499,5
5,2	227,33	3,44	14995	-0,26	61,2	-3,41	1499,5
5,4	232,85	3,39	14994	-0,07	88,49	-2,01	1499,4
5,6	236,21	3,37	14994	0,04	103,27	-1,57	1499,4
5,8	242,72	3,42	14993	0,17	126,96	-1,11	1499,3

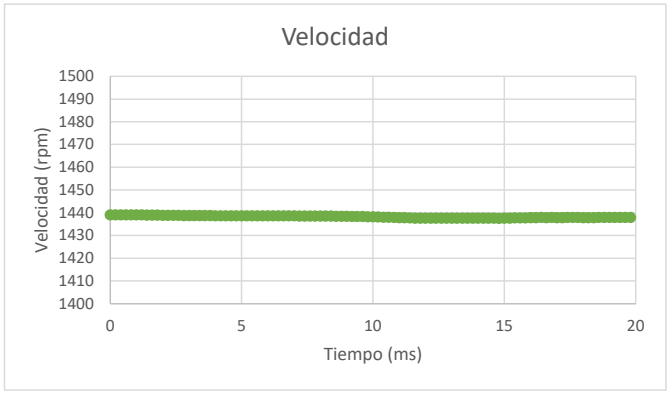
6	243,95	3,44	14992	0,34	155,18	-0,74	1499,2
6,2	244,81	3,44	14992	0,48	177,37	-0,52	1499,2
6,4	250,92	3,29	14991	1,03	255,61	-0,02	1499,1
6,6	250,66	3,45	14990	0,71	214,41	-0,26	1499
6,8	250,43	3,46	14990	0,74	218,57	-0,24	1499
7	250,78	3,48	14989	0,87	240,92	-0,13	1498,9
7,2	250,02	3,45	14988	0,97	254,52	-0,06	1498,8
7,4	249,25	3,29	14987	0,48	168,99	-0,54	1498,7
7,6	248,61	3,47	14986	0,7	213,27	-0,27	1498,6
7,8	247,4	3,37	14985	1,02	258,32	-0,03	1498,5
8	245,35	3,36	14985	1,13	275,04	0,04	1498,5
8,2	243,31	3,32	14983	1,09	266,58	0,02	1498,3
8,4	243,15	3,26	14983	1,18	276,88	0,06	1498,3
8,6	243,8	3,27	14981	1,49	325,48	0,2	1498,1
8,8	244,83	3,3	14981	1,68	358,08	0,27	1498,1
9	242,08	3,33	14980	1,86	388,2	0,32	1498
9,2	244,41	3,43	14979	2,53	498,01	0,46	1497,9
9,4	242,57	3,5	14978	2,98	573,87	0,52	1497,8
9,6	239,03	3,54	14977	2,81	549,35	0,5	1497,7
9,8	235,05	3,65	14977	2,7	538,28	0,47	1497,7
10	232,65	3,64	14976	2,52	509,91	0,45	1497,6
10,2	230,33	3,71	14976	2,43	499,45	0,43	1497,6
10,4	227,9	3,72	14975	2,2	464,25	0,38	1497,5
10,6	228,56	3,71	14975	2,09	446,65	0,36	1497,5
10,8	224,98	3,48	14975	0,91	246,75	-0,1	1497,5
11	225,3	3,68	14975	1,67	377,8	0,25	1497,5
11,2	222,61	3,67	14975	1,29	318,55	0,11	1497,5
11,4	223,31	3,65	14975	1,04	278,17	-0,01	1497,5
11,6	223,73	3,54	14974	0,83	237,28	-0,16	1497,4
11,8	225,95	3,52	14974	0,61	201,97	-0,36	1497,4
12	230,42	3,46	14974	0,67	207,98	-0,3	1497,4
12,2	233,66	3,49	14974	0,69	213,38	-0,28	1497,4
12,4	238,05	3,42	14975	0,81	227,7	-0,18	1497,5

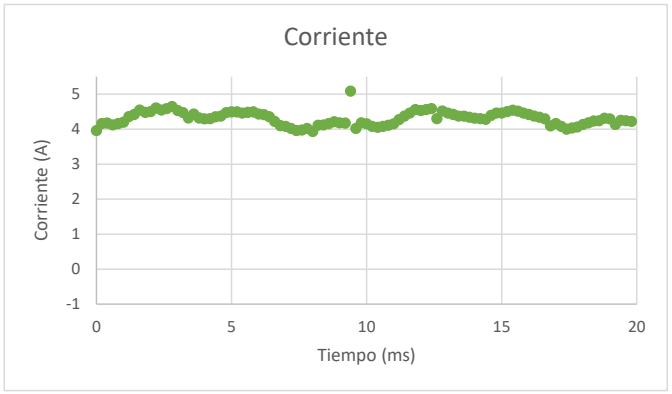
12,6	239,54	3,61	14975	1,18	298,2	0,06	1497,5
12,8	236,9	3,53	14975	1,13	284,5	0,03	1497,5
13	235,95	3,5	14976	1,21	295,48	0,08	1497,6
13,2	235,17	3,51	14976	1,21	296,15	0,08	1497,6
13,4	233,85	3,55	14976	1,27	307,36	0,1	1497,6
13,6	231,21	2,93	14977	1,15	254,29	0,05	1497,7
13,8	227,21	3,6	14977	0,97	263,36	-0,06	1497,7
14	224,24	3,58	14978	0,71	221,73	-0,25	1497,8
14,2	223,92	3,49	14978	0,67	209,39	-0,3	1497,8
14,4	222,09	3,57	14979	0,48	185,41	-0,49	1497,9
14,6	221,88	3,51	14979	0,33	157,79	-0,73	1497,9
14,8	221,34	3,46	14979	0,25	141,23	-0,91	1497,9
15	221,51	3,4	14979	0,17	125,49	-1,13	1497,9
15,2	221,32	3,37	14979	0,15	120,38	-1,2	1497,9
15,4	222,7	3,32	14979	0,23	131,34	-1	1497,9
15,6	224,59	3,35	14979	0,31	145,96	-0,81	1497,9
15,8	225,96	3,33	14979	0,59	188,1	-0,4	1497,9
16	235,57	3,33	14978	1,05	260,83	-0,01	1497,8
16,2	223,56	3,32	14978	0,93	241,86	-0,09	1497,8
16,4	224,82	3,34	14977	0,93	241,54	-0,09	1497,7
16,6	221,28	3,3	14977	1,22	284,84	0,08	1497,7
16,8	219,86	3,36	14977	0,95	246,93	-0,07	1497,7
17	220,36	3,36	14977	0,97	248,8	-0,06	1497,7
17,2	218,85	3,34	14977	0,93	242,4	-0,09	1497,7
17,4	219,12	3,18	14976	1,52	326,41	0,22	1497,6
17,6	219,19	3,28	14976	0,69	201,66	-0,29	1497,6
17,8	222,41	3,23	14976	0,82	218,71	-0,18	1497,6
18	219,38	3,22	14977	0,74	205,04	-0,25	1497,7
18,2	221,62	3,15	14977	0,81	213,39	-0,19	1497,7
18,4	220,37	3,22	14977	0,91	232,2	-0,11	1497,7
18,6	221,79	3,16	14977	0,94	232,82	-0,09	1497,7
18,8	224,1	3,18	14977	1,21	277,47	0,08	1497,7
19	226,37	3,2	14977	1,44	313,32	0,18	1497,7

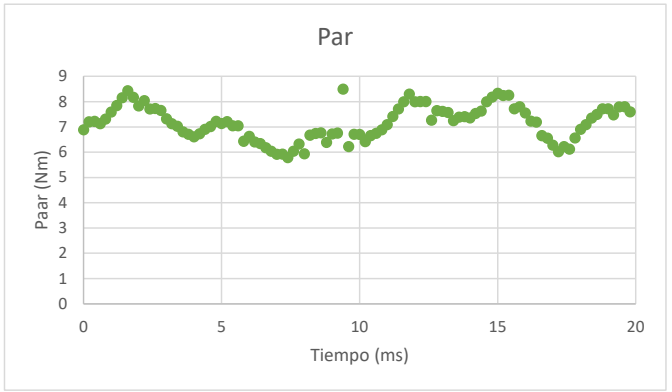
19,2	227,66	3,29	14977	1,83	381,28	0,32	1497,7
19,4	228,2	3,36	14977	2,05	419,45	0,37	1497,7
19,6	230,53	3,43	14977	2,15	438,72	0,39	1497,7
19,8	230,1	3,5	14977	2,25	459,3	0,4	1497,7

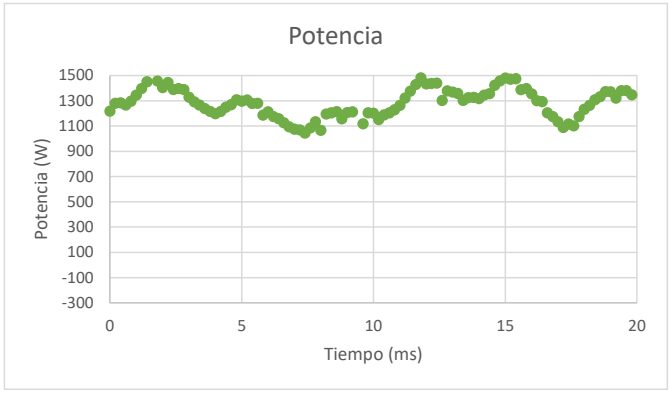
Gráficas de mediciones a 1440 rpm

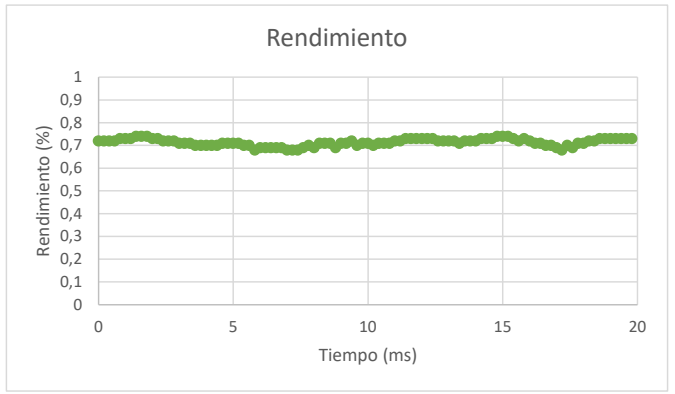




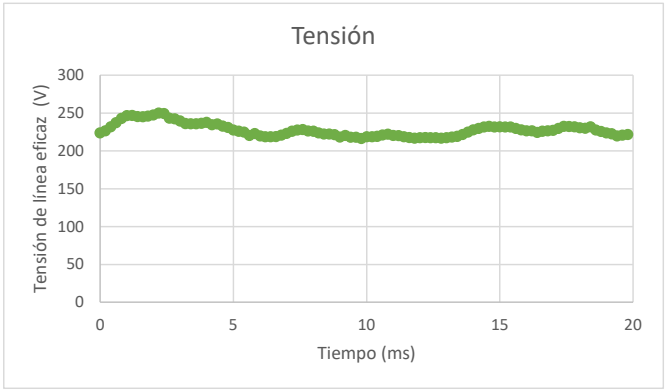


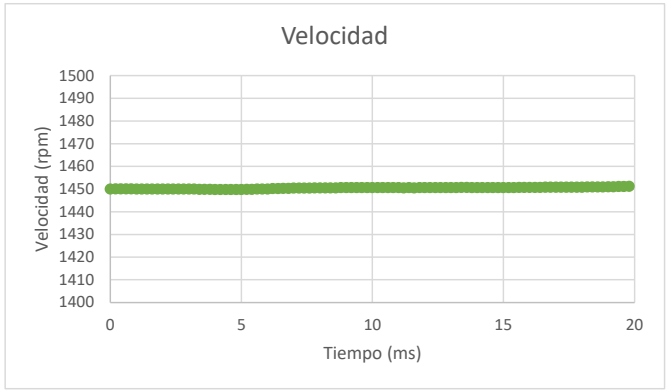


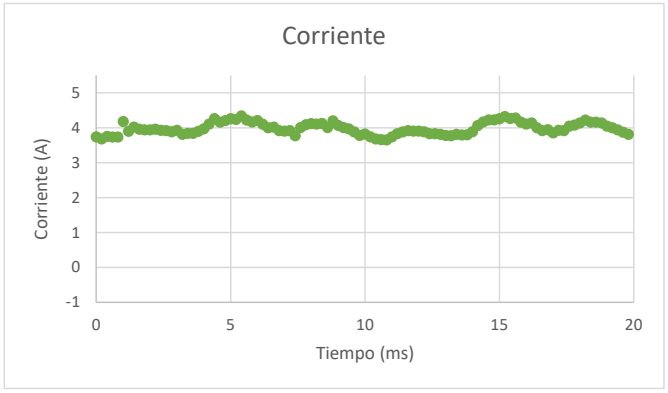


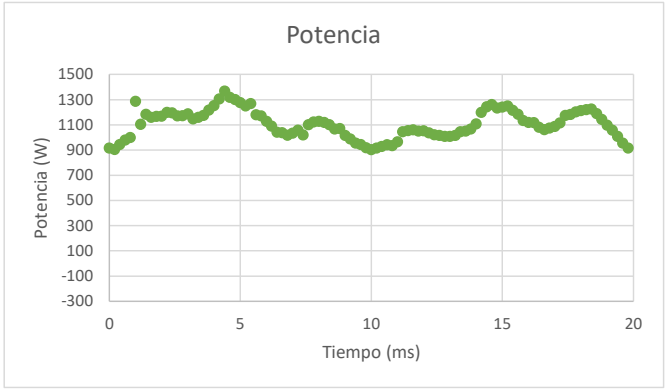


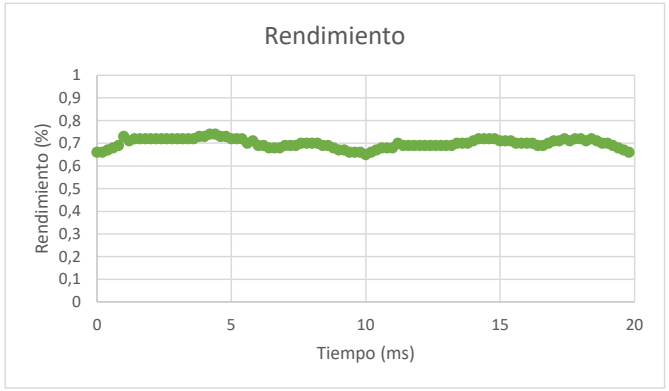
Gráficas de mediciones a 1450 rpm



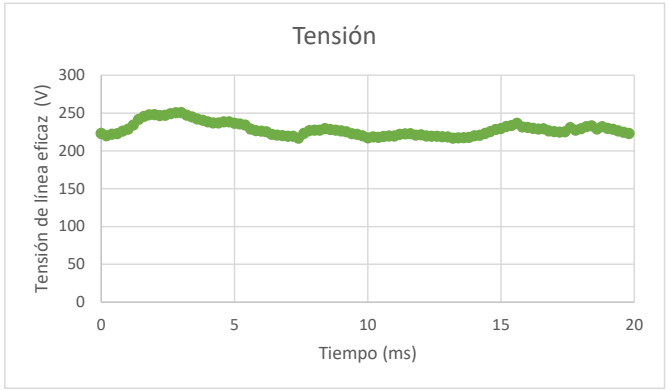


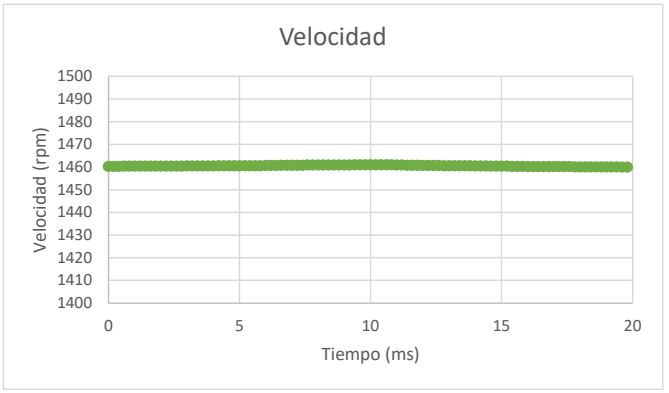


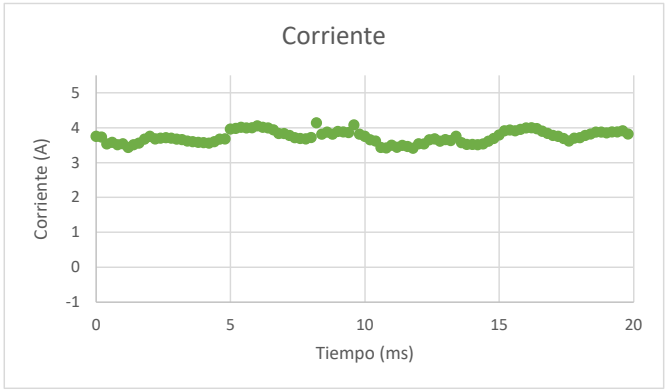


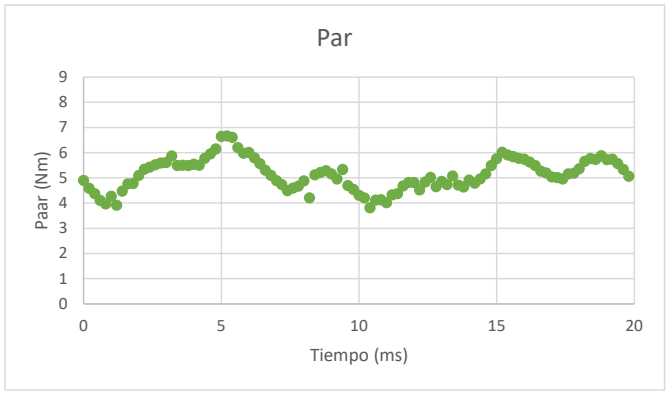


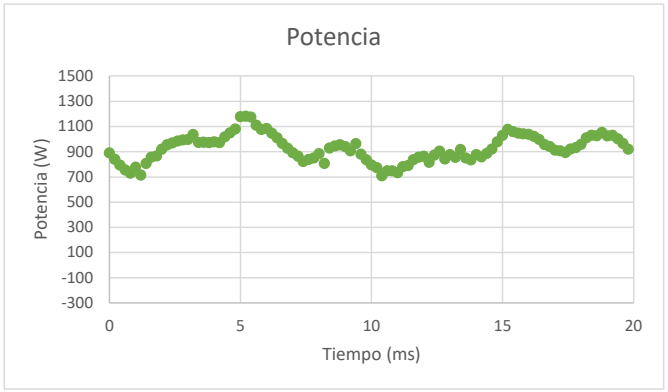
Gráficas de mediciones a 1460 rpm

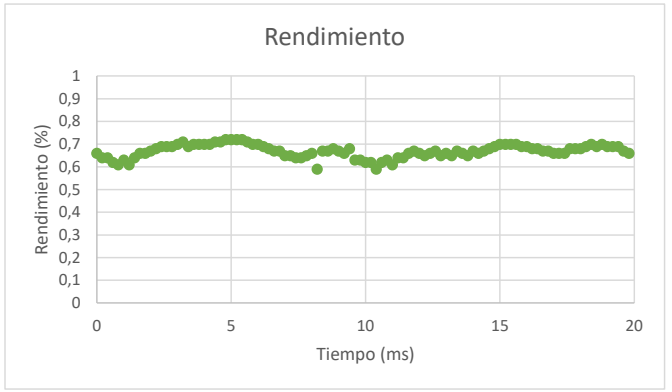




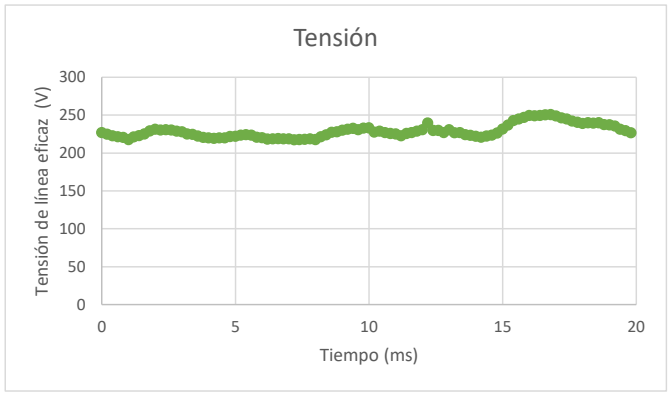


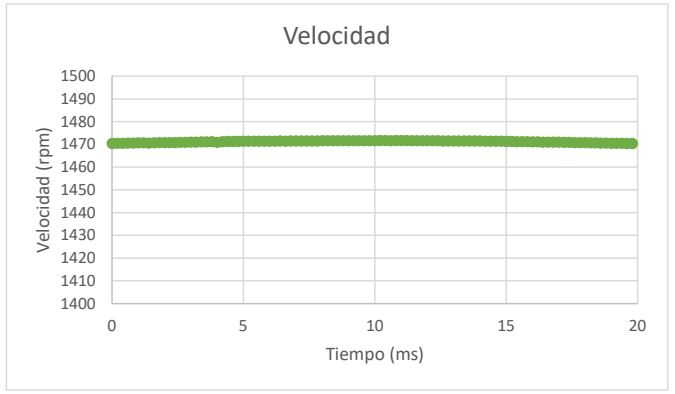


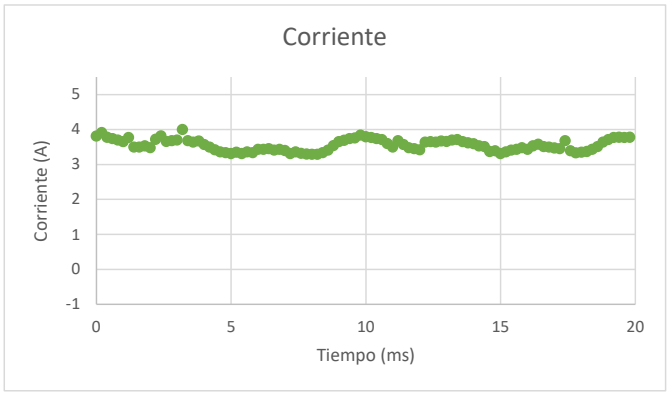


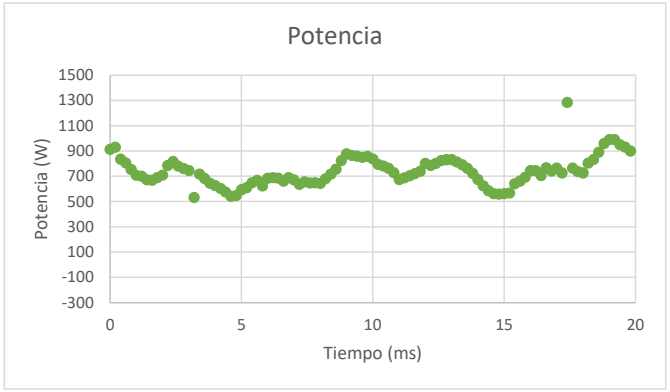


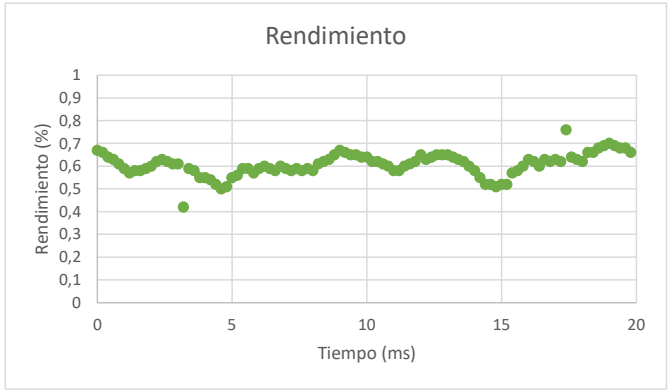
Gráficas de mediciones a 1470 rpm



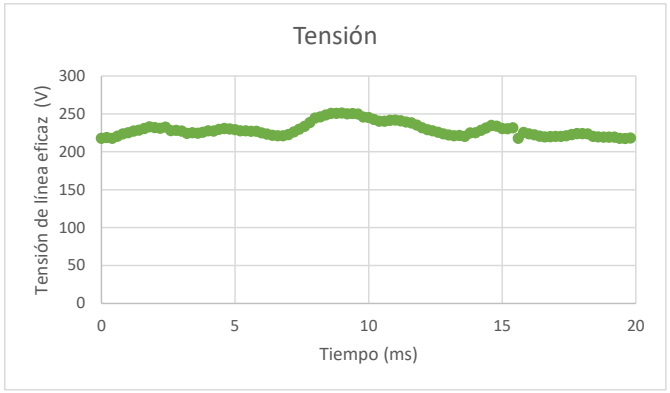


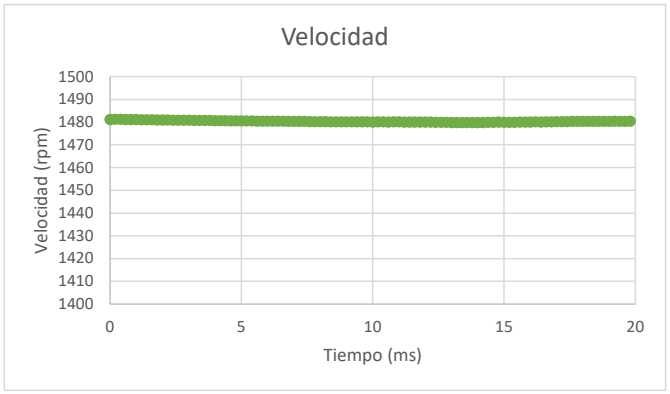


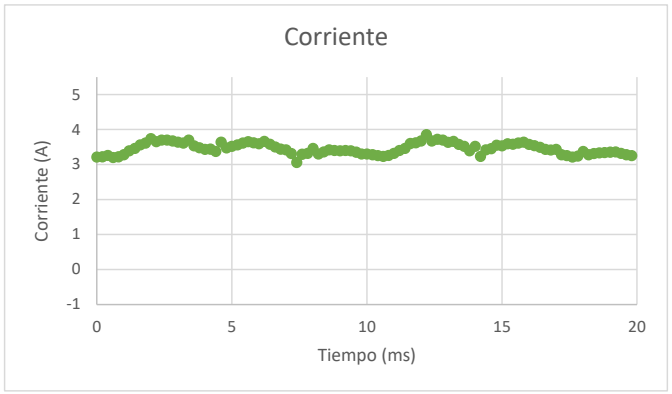


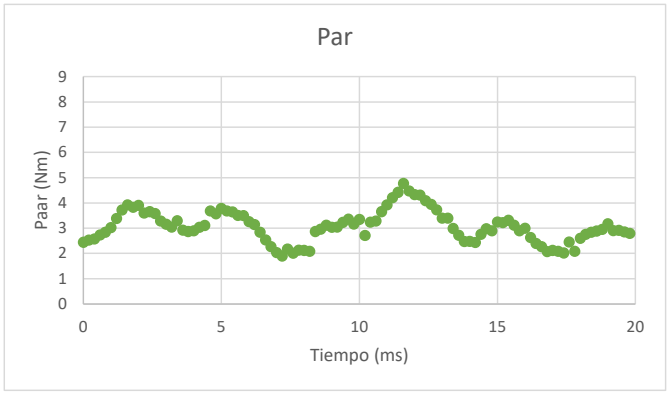


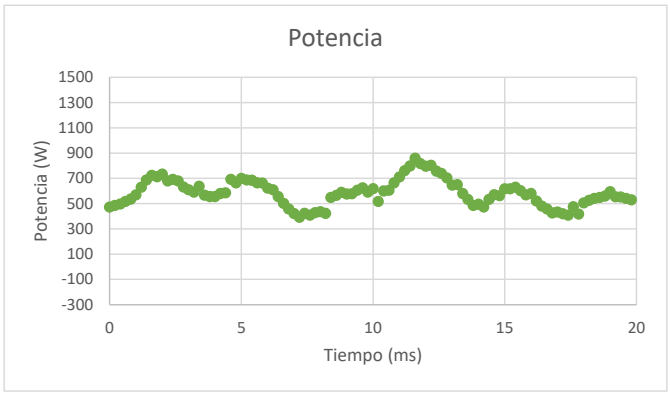
Gráficas de mediciones a 1480 rpm

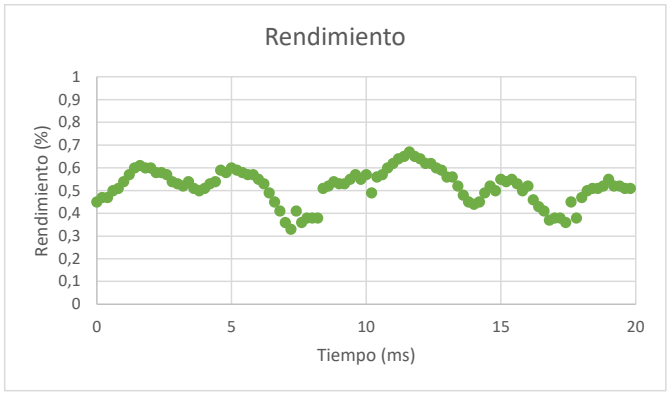




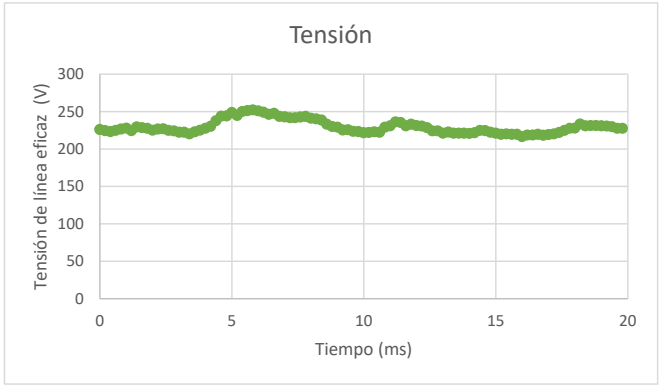


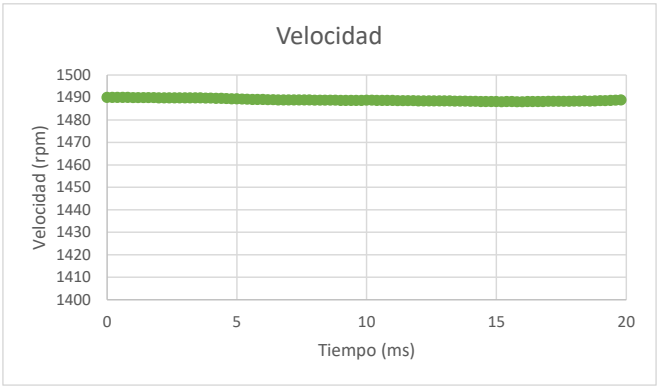


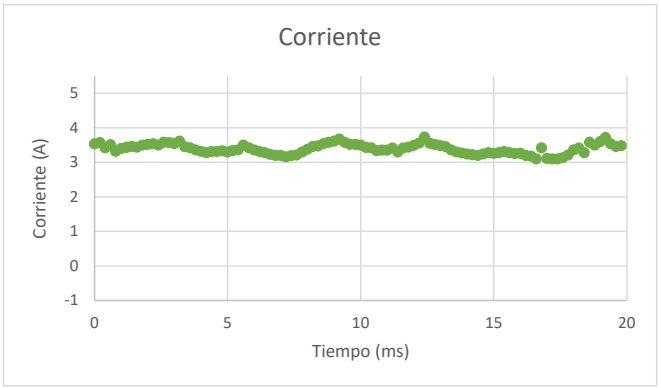


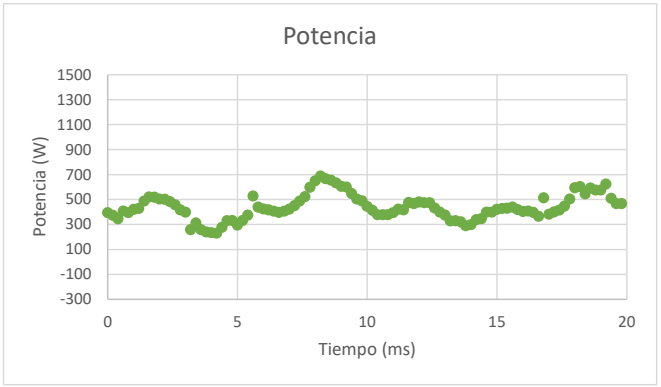


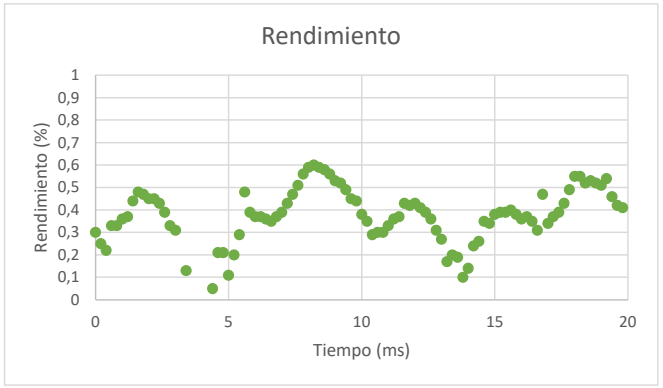
Gráficas de mediciones a 1490 rpm



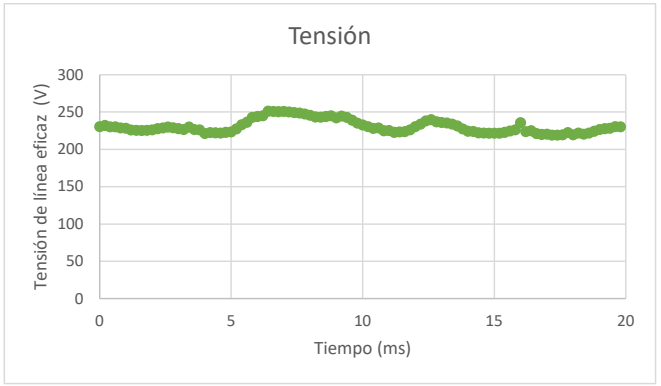


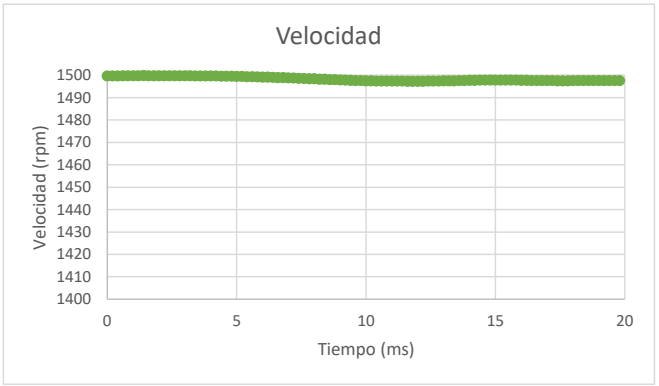


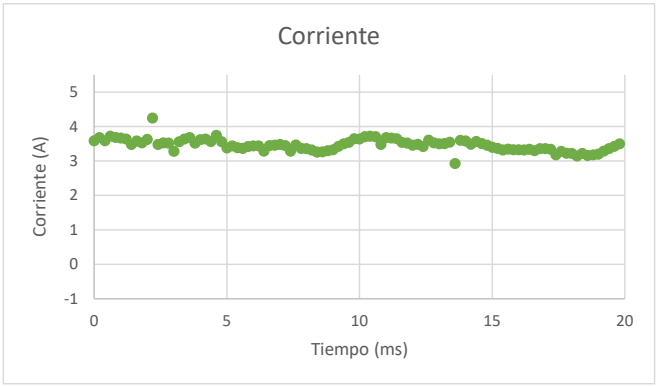


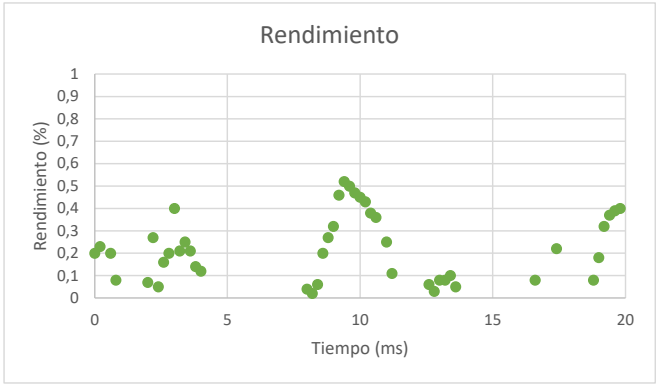


Gráficas de mediciones a 1500 rpm









CÓDIGO APP

File - C:\Users\aleja\AndroidStudioProjects\IMIEScope\app\src\main\java\com\upv\isee\imiescope\MainActivity.java

```
 1 package com.upv.isee.imiescope;
 2
 3 import android.bluetooth.BluetoothAdapter;
 4 import android.bluetooth.BluetoothDevice;
 5 import android.bluetooth.BluetoothSocket;
 6
 7 import android.graphics.Color;
 8 import android.graphics.Paint;
 9 import android.os.Handler;
10 import android.os.Message;
11 import android.os.SystemClock;
12 import android.support.v7.app.AppCompatActivity;
13 import android.os.Bundle;
14 import android.view.Menu;
15 import android.view.MenuInflater;
16 import android.view.MenuItem;
17 import android.widget.ArrayAdapter;
18 import android.widget.LinearLayout;
19 import android.widget.TextView;
20 import android.widget.Toast;
21
22 import org.achartengine.ChartFactory;
23 import org.achartengine.GraphicalView;
24 import org.achartengine.model.XYMultipleSeriesDataset;
25 import org.achartengine.model.XYSeries;
26 import org.achartengine.renderer.XYMultipleSeriesRenderer;
27 import org.achartengine.renderer.XYSeriesRenderer;
28
29 import java.io.IOException;
30 import java.io.InputStream;
31 import java.io.OutputStream;
32 import java.io.UnsupportedEncodingException;
33 import java.util.Set;
34 import java.util.UUID;
35
36 import de.nitri.gauge.Gauge;
37
```

```java
@SuppressWarnings("ALL")
public class MainActivity extends AppCompatActivity {


    private int cuenta;

    private double ch0[];
    private double us[];

    private XYMultipleSeriesDataset mDataset;
    private XYMultipleSeriesRenderer mRenderer;
    private XYSeries ch0s, ch1s, ch2s, ch3s, ch4s, ch5s;

    private double Un = 230.0;
    private double In = 4.4;
    private double I0 = 3.6;
    private double n0 = 1500.0;
  //  private double Tn = 6.07;
    private double Tn = 7.1;
    private double Pn = 1260.0;

    private XYSeriesRenderer ch0sr, ch1sr, ch2sr, ch3sr, ch4sr, ch5sr;
    private GraphicalView mChartView;

    private BluetoothAdapter mBTAdapter;
    private Set<BluetoothDevice> mPairedDevices;
    private ArrayAdapter<String> mBTArrayAdapter;

    private Handler mHandler; // Handler para tratar datos recibidos
    private ConnectedThread mConnectedThread; // Hilo de comunicación
    private BluetoothSocket mBTSocket = null; // Socket

    // Identificadores
    private final static int REQUEST_ENABLE_BT = 1; // Añadir dir
    private final static int MESSAGE_READ = 2; // Actualización mensaje
    private final static int CONNECTING_STATUS = 3; // Cómo esta el mensaje
    private int modereq = 4;
```

```
75      private int mode = 4;
76      private int datarec = 0;
77      private int datasto = 0;
78
79      private static final String TAG = "debugging bluetooth";
80      private static final UUID BTMODULEUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB"); // UUID
81
82      private TextView rms0, rms1, rms2, rms3, rms4, rms5;
83      @Override
84      protected void onCreate(Bundle savedInstanceState) {
85
86          ch0 = new double[100];
87          us = new double[100];
88          for(int i = 0; i< 100; i++){
89              us[i] = (short)(200*i)/1000.0;
90              ch0[i] = (short)(0x2000 + (0x1900*Math.sin((double)us[i]*100*Math.PI/1e6)));
91          }
92
93          super.onCreate(savedInstanceState);
94          setContentView(R.layout.activity_main);
95          getSupportActionBar().setDisplayShowHomeEnabled(true);
96          getSupportActionBar().setIcon(R.mipmap.iseeimiescopelogo);
97
98          //Unión lógica-gráfica de las lecturas enviadas por cada canal
99          rms0 = (TextView) findViewById(R.id.rms0);
100         rms0.setText(String.valueOf("0"));
101         rms1 = (TextView) findViewById(R.id.rms1);
102         rms1.setText(String.valueOf("0"));
103         rms2 = (TextView) findViewById(R.id.rms2);
104         rms2.setText(String.valueOf("0"));
105         rms3 = (TextView) findViewById(R.id.rms3);
106         rms3.setText(String.valueOf("0"));
107         rms4 = (TextView)findViewById(R.id.rms4);
108         rms4.setText(String.valueOf("0"));
109         rms5 = (TextView)findViewById(R.id.rms5);
110         rms5.setText(String.valueOf("0"));
111
```

```
112         //Unión lógica-gráfica de los indicadores de aguja
113         final Gauge gauge0, gauge1, gauge2, gauge3, gauge4, gauge5;
114         gauge0 = (Gauge) findViewById(R.id.ga_tension);
115         gauge1 = (Gauge) findViewById(R.id.ga_corriente);
116         gauge2 = (Gauge) findViewById(R.id.ga_rpm);
117         gauge3 = (Gauge) findViewById(R.id.ga_par);
118         gauge4 = (Gauge) findViewById(R.id.ga_potencia);
119         gauge5 = (Gauge) findViewById(R.id.ga_rendimiento);
120
121         mDataset = new XYMultipleSeriesDataset();
122         mRenderer = new XYMultipleSeriesRenderer();
123
124         //Creación de los canales XY para el dataset  y en string para el renderer
125         ch0s = new XYSeries("Tensión");
126         mDataset.addSeries(ch0s);
127         ch0sr = new XYSeriesRenderer();
128         mRenderer.addSeriesRenderer(ch0sr);
129
130         ch1s = new XYSeries("Corriente");
131         mDataset.addSeries(ch1s);
132         ch1sr = new XYSeriesRenderer();
133         mRenderer.addSeriesRenderer(ch1sr);
134
135         ch2s = new XYSeries("rpm");
136         mDataset.addSeries(ch2s);
137         ch2sr = new XYSeriesRenderer();
138         mRenderer.addSeriesRenderer(ch2sr);
139
140         ch3s = new XYSeries("Par");
141         mDataset.addSeries(ch3s);
142         ch3sr = new XYSeriesRenderer();
143         mRenderer.addSeriesRenderer(ch3sr);
144
145         ch4s = new XYSeries("Potencia");
146         mDataset.addSeries(ch4s);
147         ch4sr = new XYSeriesRenderer();
148         mRenderer.addSeriesRenderer(ch4sr);
```

```
149
150         ch5s = new XYSeries("Rendimiento");
151         mDataset.addSeries(ch5s);
152         ch5sr = new XYSeriesRenderer();
153         mRenderer.addSeriesRenderer(ch5sr);
154
155         //Unión lógica-gráfica de la gráfica
156         LinearLayout layout = (LinearLayout) findViewById(R.id.uilayout);
157         mChartView = ChartFactory.getLineChartView(this, mDataset, mRenderer);
158         layout.addView(mChartView, new LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,
    LinearLayout.LayoutParams.WRAP_CONTENT));
159
160
161         for(int i = 0; i< 100; i++){
162             ch0s.add(us[i],ch0[i]);
163             ch1s.add(us[i],ch0[i]);
164             ch2s.add(us[i],ch0[i]);
165             ch3s.add(us[i],ch0[i]);
166             ch4s.add(us[i],ch0[i]);
167             ch5s.add(us[i],ch0[i]);
168         }
169
170         //Primero es margen sup, segundo izq, tercero inf, cuarto der. Recomendado 20, 60, 40, 20
171         mRenderer.setMargins(new int[]{20, 60, 40, 20});
172         mRenderer.setMarginsColor(Color.rgb(236,236,236));
173         mRenderer.setAxesColor(Color.BLACK);
174         mRenderer.setXLabelsColor(Color.BLACK);
175         mRenderer.setYLabelsColor(0, Color.BLACK);
176         mRenderer.setYLabelsAlign(Paint.Align.RIGHT, 0);
177         mRenderer.setLabelsTextSize(20);
178         mRenderer.setXTitle("Muestras");
179         mRenderer.setYTitle("Valor %",0);
180         mRenderer.setAxisTitleTextSize(20);
181         mRenderer.setLabelsColor(Color.BLACK);
182         mRenderer.setYAxisMax(110.0);
183         mRenderer.setYAxisMin(0.0);
184         mRenderer.setGridColor(Color.DKGRAY);
```

```
185         mRenderer.setShowGrid(true);
186         mRenderer.setLegendTextSize(20);
187         //Altura que permite la correcta lectura de la leyenda y el título del eje es 50
188         mRenderer.setLegendHeight(50);
189
190         //Colores de cada canal, pasados a rgb, están escritos en colors.xml
191         ch0sr.setColor(Color.rgb(26, 227, 208));
192         ch1sr.setColor(Color.rgb(156, 39, 176));
193         ch2sr.setColor(Color.rgb(177, 12, 0));
194         ch4sr.setColor(Color.rgb(204, 186, 30));
195         ch3sr.setColor(Color.rgb(11,98,184));
196         ch5sr.setColor(Color.rgb(139,195,74));
197
198         //Grosor de línea
199         ch0sr.setLineWidth(3);
200         ch1sr.setLineWidth(3);
201         ch2sr.setLineWidth(3);
202         ch3sr.setLineWidth(3);
203         ch4sr.setLineWidth(3);
204         ch5sr.setLineWidth(3);
205         mChartView.repaint();
206
207         mHandler = new Handler(){
208             public void handleMessage(Message msg){
209                 if(msg.what == MESSAGE_READ){
210                     String readMessage = null;
211                     try {
212                         //Lectura del mensaje del esp32, codificación UTF-8
213                         readMessage = new String((byte[]) msg.obj,0,msg.arg1,"UTF-8");
214                     } catch (UnsupportedEncodingException e) {
215                         e.printStackTrace();
216                     }
217                     int last = readMessage.indexOf("\r\n");
218                     String header = readMessage.substring(0,last);
219                     String remains = readMessage.substring(last+2);
220
221                     if(header.equals("sampling finished")){
```

```
222
223                             boolean scontinue = false;
224                             int samples = 1;
225
226                             do{
227                                 int nextlf = remains.indexOf("\r\n");
228                                 String datoslinea = remains.substring(0,nextlf);
229                                 remains = remains.substring(nextlf+2);
230                                 scontinue = (remains.indexOf("\r\n")>0);
231                                 if(datoslinea.contains("sampling")){
232                                     nextlf = remains.indexOf("\r\n");
233                                     datoslinea = remains.substring(1,nextlf);
234                                     remains = remains.substring(nextlf+2);
235                                     scontinue = (remains.indexOf("\r\n")>0);
236                                 }
237
238                                 if(datoslinea.contains("Relay"))
239                                     break;
240
241                                 //datoslinea = datoslinea.substring(1);
242                                 int nextsc = datoslinea.indexOf(";");
243
244                                 //Recibimos y almacenamos hasta 100 valores
245                                 datarec = datarec +1;
246                                 if(datarec >100)
247                                     datasto = 100;
248                                 else
249                                     datasto = datarec;
250
251                                 //Leemos valor a valor lo enviado por el esp32
252                                 double timems =0.0;
253
254                                 String inicio = datoslinea.substring(0, nextsc);
255                                 double ch0m = Double.valueOf(inicio);
256                                 datoslinea = datoslinea.substring(nextsc+1);
257                                 nextsc = datoslinea.indexOf(";");
258                                 double ch1m = Double.valueOf(datoslinea.substring(0,nextsc));
```

```
259                         datoslinea = datoslinea.substring(nextsc+1);
260                         nextsc = datoslinea.indexOf(";");
261                         double ch2m = Double.valueOf(datoslinea.substring(0,nextsc));
262                         datoslinea = datoslinea.substring(nextsc+1);
263                         nextsc = datoslinea.indexOf(";");
264                         double ch3m = Double.valueOf(datoslinea.substring(0,nextsc));
265                         datoslinea = datoslinea.substring(nextsc+1);
266                         nextsc = datoslinea.indexOf(";");
267                         double ch4m = Double.valueOf(datoslinea.substring(0,nextsc));
268                         datoslinea = datoslinea.substring(nextsc+1);
269                         nextsc = datoslinea.indexOf(";");
270                         double ch5m = Double.valueOf(datoslinea);
271
272                         //Paso de digital a analógico en valor porcentual
273                         double ch0man = (ch0m*1.1/4096)*100;
274                         double ch1man = (ch1m*1.1/4096)*100;
275                         double ch2man = (ch2m*1.1/4096)*100;
276                         double ch3man = (ch3m*1.1/4096)*100;
277                         double ch4man = (ch4m*1.1/4096)*100;
278                         double ch5man = (ch5m*1.1/4096)*100;
279
280                         //Añadimos a los valores de XY el número de dato y su valor analógico
281                         ch0s.add(datarec,ch0man);
282                         ch1s.add(datarec,ch1man);
283                         ch2s.add(datarec,ch2man);
284                         ch3s.add(datarec,ch3man);
285                         ch4s.add(datarec,ch4man);
286                         ch5s.add(datarec,ch5man);
287
288                         //Casting a float para la lectura del indicador de aguja y su movimiento al mismo
289                         float ch0manf = (float) ch0man;
290                         float ch1manf = (float) ch1man;
291                         float ch2manf = (float) ch2man;
292                         float ch3manf = (float) ch3man;
293                         float ch4manf = (float) ch4man;
294                         float ch5manf = (float) ch5man;
295
```

```
296                             gauge0.moveToValue(ch0manf);
297                             gauge1.moveToValue(ch1manf);
298                             gauge2.moveToValue(ch2manf);
299                             gauge3.moveToValue(ch3manf);
300                             gauge4.moveToValue(ch4manf);
301                             gauge5.moveToValue(ch5manf);
302
303                             //A cada valor superior a 100, eliminamos el primero para mantener 100 datos
304                             if(datarec > 100){
305                                 ch0s.remove(0);
306                                 ch1s.remove(0);
307                                 ch2s.remove(0);
308                                 ch3s.remove(0);
309                                 ch4s.remove(0);
310                                 ch5s.remove(0);
311                                 mRenderer.setXAxisMax(datarec);
312                                 mRenderer.setXAxisMin(datarec-100);
313                             }else{
314                                 mRenderer.setXAxisMax(100);
315                                 mRenderer.setXAxisMin(0);
316                             }
317
318                         }while(scontinue);
319
320                     }
321                     mChartView.repaint();
322
323                     //Añadimos a cada etiqueta su valor en SI
324                     rms0.setText(String.format("%.0f", ch0s.getY(datasto-1)*Un/100));
325                     rms1.setText(String.format("%.2f", ch1s.getY(datasto-1)*In/100.0));
326                     rms2.setText(String.format("%.0f", ch2s.getY(datasto-1)*n0/100.0));
327                     rms3.setText(String.format("%.2f", ch3s.getY(datasto-1)*Tn/100.0));
328                     rms4.setText(String.format("%.0f", ch4s.getY(datasto-1)*Pn/100.0));
329                     rms5.setText(String.format("%.2f", ch5s.getY(datasto-1)));
330
331
332
```

```
333                     if((modereq!=3)&&(mode!=modereq)){
334                         mode = modereq;
335                         switch(mode) {
336                             case 0:
337                                 mConnectedThread.write("0");
338                                 break;
339                             case 1:
340                                 mConnectedThread.write("1");
341                                 break;
342                             case 2:
343                                 mConnectedThread.write("2");
344                                 break;
345                             default:
346                                 // code block
347                         }
348                     }
349
350                     if(mode == 3) mConnectedThread.write("3");
351
352
353                 }
354
355                 if(msg.what == CONNECTING_STATUS){
356                     if(msg.arg1 == 1);
357
358                     else ;
359
360                 }
361
362             }
363         };
364
365
366         mBTAdapter = BluetoothAdapter.getDefaultAdapter(); // Controla la radio BT
367
368         mPairedDevices = mBTAdapter.getBondedDevices();
369         int imiedevices = 0;
```

```
370         BluetoothDevice imiedevice =null;
371         for (BluetoothDevice device : mPairedDevices) {
372
373             if(device.getName().contains("MAQASINC")){
374                 Toast.makeText(getApplicationContext(), device.getAddress(), Toast.LENGTH_SHORT).show();
375                 imiedevices++;
376                 imiedevice = device;
377             }
378
379             SystemClock.sleep(500);
380         }
381
382
383
384         if(imiedevices ==1){
385             Toast.makeText(getApplicationContext(), "Connecting to " + imiedevice.getName(), Toast.
    LENGTH_SHORT).show();
386             final String address = imiedevice.getAddress();
387             final String name = imiedevice.getName();
388             new Thread()
389             {
390                 public void run() {
391                     boolean fail = false;
392
393                     BluetoothDevice device = mBTAdapter.getRemoteDevice(address);
394
395                     try {
396                         mBTSocket = createBluetoothSocket(device);
397                     } catch (IOException e) {
398                         fail = true;
399                         Toast.makeText(getBaseContext(), "Socket creation failed", Toast.LENGTH_SHORT).show();
400                     }
401                     // Apertura socket de conexión
402                     try {
403                         mBTSocket.connect();
404                     } catch (IOException e) {
405                         try {
```

```
406                             fail = true;
407                             mBTSocket.close();
408                             mHandler.obtainMessage(CONNECTING_STATUS, -1, -1)
409                                     .sendToTarget();
410                         } catch (IOException e2) {
411                             Toast.makeText(getBaseContext(), "Socket creation failed", Toast.LENGTH_SHORT).
    show();
412                         }
413                     }
414                     if(!fail) {
415                         mConnectedThread = new ConnectedThread(mBTSocket);
416                         mConnectedThread.start();
417
418                         mHandler.obtainMessage(CONNECTING_STATUS, 1, -1, name)
419                                 .sendToTarget();
420                     }
421                 }
422             }.start();
423
424         }
425         else if(imiedevices ==0){
426             Toast.makeText(getApplicationContext(), "No instrument paired", Toast.LENGTH_SHORT).show();
427
428         }
429         else{
430             Toast.makeText(getApplicationContext(), "More than one instrument paired", Toast.LENGTH_SHORT).
    show();
431         }
432
433
434     }
435
436
437
438     private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws IOException {
439         return  device.createRfcommSocketToServiceRecord(BTMODULEUUID);
440         //Conexión de salida segura con el UUID
```

```
441     }
442
443
444
445     @Override
446     public boolean onCreateOptionsMenu(Menu menu) {
447         MenuInflater inflater = getMenuInflater();
448         inflater.inflate(R.menu.my_options_menu, menu);
449         return super.onCreateOptionsMenu(menu);
450     }
451         //Método que selecciona el menú de spinner, finalmente sólo un método
452     public boolean onOptionsItemSelected(MenuItem item) {
453
454         switch (item.getItemId()) {
455             case R.id.Ext:
456                 ch0s.clear();
457                 ch1s.clear();
458                 ch2s.clear();
459                 ch3s.clear();
460                 ch4s.clear();
461                 ch5s.clear();
462                 datarec = 0;
463                 datasto = 0;
464                 if(mode !=3){
465                     if(mConnectedThread != null)
466                         mConnectedThread.write("3");
467                     modereq = mode = 3;}
468
469                 break;
470
471         }
472         return true;
473     }
474
475     private class ConnectedThread extends Thread {
476         private final BluetoothSocket mmSocket;
477         private final InputStream mmInStream;
```

```
478         private final OutputStream mmOutStream;
479
480         public ConnectedThread(BluetoothSocket socket) {
481             mmSocket = socket;
482             InputStream tmpIn = null;
483             OutputStream tmpOut = null;
484
485             // Flujos de input y output con variables intermedias ya que los flujos son *final*
486             try {
487                 tmpIn = socket.getInputStream();
488                 tmpOut = socket.getOutputStream();
489             } catch (IOException e) { }
490
491             mmInStream = tmpIn;
492             mmOutStream = tmpOut;
493         }
494
495         public void run() {
496             byte[] buffer = new byte[4000];  // Buffer para el flujo
497             int bytes;
498             // A la espera hasta que
499             while (true) {
500                 try {
501                     // Lectura de flujo input
502                     bytes = mmInStream.available();
503                     if(bytes != 0) {
504                         SystemClock.sleep(500); //Colas
505                         bytes = mmInStream.available();
506                         bytes = (bytes <4001)? bytes : 4000;
507                         bytes = mmInStream.read(buffer, 0, bytes); // Cuánto se ha leído?
508                         mHandler.obtainMessage(MESSAGE_READ, bytes, -1, buffer)
509                                 .sendToTarget();
510                     }
511                 } catch (IOException e) {
512                     e.printStackTrace();
513
514                     break;
```

```
515                     }
516                 }
517             }
518
519             public void write(String input) {
520                 byte[] bytes = input.getBytes();
521                 try {
522                     mmOutStream.write(bytes);
523                 } catch (IOException e) { }
524             }
525
526             public void cancel() {
527                 try {
528                     mmSocket.close();
529                 } catch (IOException e) { }
530             }
531         }
532
533 }
534
```

File - C:\Users\aleja\AndroidStudioProjects\IMIEScope\app\src\main\res\layout\activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:gauge="http://schemas.android.com/apk/res-auto"
    tools:context="com.upv.isee.imiescope.MainActivity">

    <TableLayout
        android:id="@+id/tableLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        app:layout_constraintTop_toTopOf="parent">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/ud_tension"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/ud_tension"
                android:textColor="@color/color_tension"
                android:textSize="34sp" />

            <TextView
                android:id="@+id/rms0"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="8dp"
                android:layout_marginLeft="8dp"
                android:text="@string/et_tension"
                android:textColor="@color/color_tension"
                android:textSize="34sp" />
```

```
38
39              <TextView
40                  android:id="@+id/ud_corriente"
41                  android:layout_width="wrap_content"
42                  android:layout_height="wrap_content"
43                  android:text="@string/ud_corriente"
44                  android:textColor="@color/color_corriente"
45                  android:textSize="34sp" />
46
47              <TextView
48                  android:id="@+id/rms1"
49                  android:layout_width="wrap_content"
50                  android:layout_height="wrap_content"
51                  android:layout_marginStart="8dp"
52                  android:layout_marginLeft="8dp"
53                  android:text="@string/et_corriente"
54                  android:textColor="@color/color_corriente"
55                  android:textSize="34sp" />
56
57              <TextView
58                  android:id="@+id/ud_rpm"
59                  android:layout_width="wrap_content"
60                  android:layout_height="wrap_content"
61                  android:text="@string/ud_rpm"
62                  android:textColor="@color/color_rpm"
63                  android:textSize="34sp" />
64
65              <TextView
66                  android:id="@+id/rms2"
67                  android:layout_width="wrap_content"
68                  android:layout_height="wrap_content"
69                  android:layout_marginStart="8dp"
70                  android:layout_marginLeft="8dp"
71                  android:text="@string/et_rpm"
72                  android:textColor="@color/color_rpm"
73                  android:textSize="34sp" />
74
```

```
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <de.nitri.gauge.Gauge
                android:id="@+id/ga_tension"
                android:layout_width="100dp"

                android:layout_height="100dp"
                gauge:initialValue="0"
                gauge:lowerText="Tensión %"
                gauge:maxValue="120"
                gauge:minValue="0"
                gauge:needleColor="@color/color_tension"
                gauge:scaleColor="@color/color_escala"
                gauge:totalNicks="140"
                gauge:valuePerNick="1" />

            <TextView
                android:id="@+id/textView2"
                android:layout_width="57dp"
                android:layout_height="wrap_content" />

            <de.nitri.gauge.Gauge
                android:id="@+id/ga_corriente"
                android:layout_width="100dp"

                android:layout_height="100dp"
                gauge:initialValue="0"
                gauge:lowerText="Corriente %"
                gauge:maxValue="120"
                gauge:minValue="0"
                gauge:needleColor="@color/color_corriente"
                gauge:scaleColor="@color/color_escala"
                gauge:totalNicks="140"
```

```
112                 gauge:valuePerNick="1" />
113
114             <TextView
115                 android:id="@+id/textView3"
116                 android:layout_width="84dp"
117                 android:layout_height="wrap_content" />
118
119             <de.nitri.gauge.Gauge
120                 android:id="@+id/ga_rpm"
121
122                 android:layout_width="100dp"
123                 android:layout_height="100dp"
124                 gauge:initialValue="0"
125                 gauge:lowerText="rpm %"
126                 gauge:maxValue="120"
127                 gauge:minValue="0"
128                 gauge:needleColor="@color/color_rpm"
129                 gauge:scaleColor="@color/color_escala"
130                 gauge:totalNicks="140"
131                 gauge:valuePerNick="1" />
132
133         </TableRow>
134
135         <TableRow
136             android:layout_width="match_parent"
137             android:layout_height="match_parent" >
138
139             <TextView
140                 android:id="@+id/ud_par"
141                 android:layout_width="wrap_content"
142                 android:layout_height="wrap_content"
143                 android:lineSpacingExtra="8sp"
144                 android:text="@string/ud_par"
145                 android:textColor="@color/color_par"
146                 android:textSize="34sp" />
147
148             <TextView
```

```
149                 android:id="@+id/rms3"
150                 android:layout_width="wrap_content"
151                 android:layout_height="wrap_content"
152                 android:layout_marginStart="8dp"
153                 android:layout_marginLeft="8dp"
154                 android:lineSpacingExtra="8sp"
155                 android:text="@string/et_par"
156                 android:textColor="@color/color_par"
157                 android:textSize="34sp" />
158
159             <TextView
160                 android:id="@+id/ud_potencia"
161                 android:layout_width="wrap_content"
162                 android:layout_height="wrap_content"
163                 android:lineSpacingExtra="8sp"
164                 android:text="@string/ud_potencia"
165                 android:textColor="@color/color_potencia"
166                 android:textSize="34sp" />
167
168             <TextView
169                 android:id="@+id/rms4"
170                 android:layout_width="wrap_content"
171                 android:layout_height="wrap_content"
172                 android:layout_marginStart="8dp"
173                 android:layout_marginLeft="8dp"
174                 android:lineSpacingExtra="8sp"
175                 android:text="@string/et_potencia"
176                 android:textColor="@color/color_potencia"
177                 android:textSize="34sp" />
178
179             <TextView
180                 android:id="@+id/ud_rendimiento"
181                 android:layout_width="wrap_content"
182                 android:layout_height="wrap_content"
183                 android:lineSpacingExtra="8sp"
184                 android:text="@string/ud_rendimiento"
185                 android:textColor="@color/color_rendimiento"
```

```
186                 android:textSize="34sp" />
187
188             <TextView
189                 android:id="@+id/rms5"
190                 android:layout_width="wrap_content"
191                 android:layout_height="wrap_content"
192                 android:layout_marginStart="8dp"
193                 android:layout_marginLeft="8dp"
194                 android:lineSpacingExtra="8sp"
195                 android:text="@string/et_rendimiento"
196                 android:textColor="@color/color_rendimiento"
197                 android:textSize="34sp" />
198
199         </TableRow>
200
201         <TableRow
202             android:layout_width="match_parent"
203             android:layout_height="match_parent" >
204
205             <de.nitri.gauge.Gauge
206                 android:id="@+id/ga_par"
207
208                 android:layout_width="100dp"
209                 android:layout_height="100dp"
210                 gauge:initialValue="0"
211                 gauge:lowerText="Par %"
212                 gauge:maxValue="120"
213                 gauge:minValue="0"
214                 gauge:needleColor="@color/color_par"
215                 gauge:scaleColor="@color/color_escala"
216                 gauge:totalNicks="140"
217                 gauge:valuePerNick="1" />
218
219             <TextView
220                 android:id="@+id/textView4"
221                 android:layout_width="wrap_content"
222                 android:layout_height="wrap_content" />
```

```
223
224             <de.nitri.gauge.Gauge
225                 android:id="@+id/ga_potencia"
226
227                 android:layout_width="100dp"
228                 android:layout_height="100dp"
229                 gauge:initialValue="0"
230                 gauge:lowerText="Potencia %"
231                 gauge:maxValue="120"
232                 gauge:minValue="0"
233                 gauge:needleColor="@color/color_potencia"
234                 gauge:scaleColor="@color/color_escala"
235                 gauge:totalNicks="140"
236                 gauge:valuePerNick="1" />
237
238             <TextView
239                 android:id="@+id/textView5"
240                 android:layout_width="wrap_content"
241                 android:layout_height="wrap_content" />
242
243             <de.nitri.gauge.Gauge
244                 android:id="@+id/ga_rendimiento"
245
246                 android:layout_width="100dp"
247                 android:layout_height="match_parent"
248                 gauge:initialValue="0"
249                 gauge:lowerText="Rendimiento %"
250                 gauge:maxValue="120"
251                 gauge:minValue="0"
252                 gauge:needleColor="@color/color_rendimiento"
253                 gauge:scaleColor="@color/color_escala"
254                 gauge:totalNicks="140"
255                 gauge:valuePerNick="1" />
256         </TableRow>
257
258     </TableLayout>
259
```

```
260     <TextView
261         android:id="@+id/cuentaetiqueta"
262         android:layout_width="wrap_content"
263         android:layout_height="wrap_content"
264         android:layout_marginStart="8dp"
265         android:layout_marginLeft="8dp"
266         android:layout_marginTop="8dp"
267         android:text="@string/et_inicial"
268         android:textSize="24sp"
269         android:textStyle="bold"
270         app:layout_constraintStart_toStartOf="parent"
271         app:layout_constraintTop_toTopOf="parent" />
272
273     <LinearLayout
274         android:id="@+id/uilayout"
275         android:layout_width="0dp"
276         android:layout_height="450dp"
277         android:layout_alignParentStart="true"
278         android:layout_alignParentLeft="true"
279         android:orientation="horizontal"
280         android:visibility="visible"
281         app:layout_constraintBottom_toBottomOf="parent"
282         app:layout_constraintEnd_toEndOf="parent"
283         app:layout_constraintStart_toStartOf="parent"
284         app:layout_constraintTop_toBottomOf="@id/cuentaetiqueta"
285         app:layout_constraintVertical_bias="0.904"
286         tools:visibility="visible" />
287
288
289 </android.support.constraint.ConstraintLayout>
290
```

SIEMENS

Data sheet for three-phase Squirrel-Cage-Motors

MLFB-Ordering data : **1LA7090-4AA10**

Client order no. :
Order no. :
Offer no. :
Remarks :

Item no. :
Consignment no. :
Project :

Electrical data:

Rated voltage : (1) 230 VD/400 VY, 50 Hz, 460 VY, 60 Hz

	50 Hz	60 Hz
Frequency :		
Rated power :	1.10 kW	1.30 kW
Rated speed :	1415 1/ min	1715 1/ min
Rated torque :	7.4 Nm	7.2 Nm
Rated current (IE) :	VD 4.52 A / VY 2.60 A	VY 2.45 A
Starting / rated current :	4.6	5.0
Breakdown / rated torque :	2.4	2.4
Starting / rated torque :	2.3	2.3

	4/4	3/4	2/4	4/4	3/4	2/4
Efficiency %	75.0%	75.0%	72.0%	79.0%	79.5%	77.0%
Power factor :	0.81	0.76	0.66	0.82	0.77	0.67
Efficiency class :	IE1			IE1		

Mechanical data:

Sound pressure level 50Hz/60Hz (load) :	48 dB(A) / 52 dB(A)
Moment of inertia :	0.0024 kg*m²
Bearing DE :	6205 2ZC3
Bearing NDE :	6004 2ZC3
Type of bearing :	Floating bearings pre-loaded DE (standard)
Condensate drainage holes :	No
Regreasing device :	No
Lubricants :	Esso Unirex N3
Grease lifetime/Relubrication interval :	40000 h
Quantity of grease for relubrication :	null g
External earthing terminal :	No
Coating :	Special paint finish RAL 7030 stone gray

Environmental conditions:

Ambient temperature :	-20 °C - +40 °C
Altitude above sea level :	1000 m
Standards and specifications :	IEC, DIN, ISO, VDE, EN

General data:

Frame size	090 S
Design of electrical machine :	(0) IM B3 / B6 / B7 / B8 / V5 without canopy
Weight in kg, without optional accessories :	13.00 kg
Frame material :	Aluminum
Degree of protection :	IP 55
Method of cooling, TEFC :	IC 411
Vibration class :	A (Standard)
Insulation :	155(F) to 130(B)
Duty type :	S1 - continuous duty
Direction of rotation :	Bi-directional

Terminal box:

Material of terminal box :	
Type of terminal box :	gk 030
Contact screw thread :	
Max. cross-sectional area :	
Cable diameter from ... to ... :	
Cable entry :	
Cable gland :	

Special design:

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DISEÑO DE UN INSTRUMENTO CON INTERFAZ MÓVIL PARA LA MEDICIÓN DE MAGNITUDES ELÉCTRICAS Y MECÁNICAS EN MÁQUINAS ASÍNCRONAS DE 1.1KW

AUTOR: **ALEJANDRO ROMAGUERA ASENSIO**

TUTOR: **JAVIER ANDRÉS MARTÍNEZ ROMÁN**

Curso Académico: **2020-21**