Universidad Zaragoza
1542

Trabajo Fin de Grado

Tiempo de respuesta de sistema de almacenamiento e inversor bidireccional trifásico.

Autor/es

Miguel Torres Hornero

Director/es

Alfonso Blesa Gascón

Escuela Universitaria Politécnica de Teruel

2018

Tiempo de respuesta de sistema de almacenamiento e inversor bidireccional trifásico

Miguel Torres Hornero

Agradecimientos

Me gustaría agradecer a mi director de Trabajo Fin de Grado, el profesor Alfonso Blesa Gascón, la ayuda y consejo que me ha prestado a la hora de enfrentarme a los problemas derivados de la implementación de este trabajo.

Igualmente, querría mostrar mi agradecimiento a los miembros del Instituto de Investigación de Integración de Demanda Energética C4DSI (*Center for Demand Side Integration*) por darme la oportunidad y recursos necesarios para su elaboración. Especialmente a mis compañeros Sebastian Farrenkopf y Matthias Kühl por su dedicación y consejo. También a los profesores Hans Schäfers y Franz Schubert por la confianza que depositaron en mí.

Por último, agradecer su labor como coordinador de programas de movilidad al profesor Agustín Llorente por facilitar la realización de mi estancia Erasmus en Hamburgo y formar parte del centro de investigación.

Tiempo de respuesta de sistema de almacenamiento e inversor bidireccional trifásico

Miguel Torres Hornero

"Tiempo de respuesta de sistema de almacenamiento e inversor bidireccional trifásico"

RESUMEN

En el presente Trabajo Fin de grado, se propone y comprueba la implementación de una aplicación *software* para la comunicación y medida del tiempo de respuesta del sistema de almacenamiento energético compuesto por baterías VU-CAB52 e inversor bidireccional trifásico BAT50, mediante el uso de un osciloscopio controlado por dicha aplicación.

Así mismo, se analizan los requerimientos del posterior modelado de una máquina síncrona virtual (VISMA: *Virtual Synchronous Machine*) y, por último, se lleva a cabo el análisis estadístico de los datos obtenidos, para verificar que el sistema de almacenamiento es apto para el posterior modelado del VISMA.

Palabras clave: VISMA, *Smart Grid*, inversor trifásico, baterías, almacenamiento energético, tiempo de respuesta.

"Energy storage and three phase inverter time response"

ABSTRACT

This thesis describes and verifies the implementation of a software application for the communication and time response measurement of the energy storage system VU-CAB52 and three phase inverter BAT50, by means of the control of an oscilloscope.

Additionally, the requirements for the subsequent modelling of a Virtual Synchronous Machine (VISMA) are analysed. Finally, a statistic analysis of the obtained data is made to check whether the system is suitable for the VISMA modelling.

Keywords: VISMA, Smart Grid, three phase inverter, battery, energy storage, time response.

Tiempo de respuesta de sistema de almacenamiento e inversor bidireccional trifásico

Miguel Torres Hornero

ÍNDICE

Tiempo de respuesta de sistema de almacenamiento e inversor bidireccional trifásico

Miguel Torres Hornero

LISTA DE TABLAS

Tiempo de respuesta de sistema de almacenamiento e inversor bidireccional trifásico

Miguel Torres Hornero

LISTA DE FIGURAS

1 Introducción y objetivos del Trabajo Fin de Grado

Teniendo en cuenta la creciente implantación de los sistemas generadores de energía eléctrica basados en las fuentes de energías renovables, se hace necesario gestionar de forma eficiente el uso de los sistemas de almacenamiento.

La función principal de un sistema de almacenamiento, a partir del binomio batería - conversor, es la acumulación del excedente de producción energética con el fin de devolverlo a la red cuando sea necesario. Este excedente energético puede proceder por ejemplo de placas solares fotovoltaicas, y en este caso el pico de generación no suele coincidir con el pico de demanda. Actualmente existen diversas opciones comerciales para llevar a cabo este almacenamiento, que permiten tanto el almacenamiento de energía como la creación de una red alterna propia, aislada de la red eléctrica nacional para el autoabastecimiento.

El sistema de baterías y conversor utilizado para la realización del presente Trabajo Fin de Grado, se encuentra ubicado en el Instituto de Investigación de Integración de Demanda Energética C4DSI (*Center for Demand Side Integration*), situado en la ciudad de Hamburgo en Alemania y perteneciente a la Universidad de Ciencias Aplicadas de Hamburgo (HAW: *Hochschule für Angewandte Wissenschaften*). Una de las líneas de investigación de dicho centro se ocupa del estudio de las *Smart Grids*, o redes inteligentes de distribución de energía.

El término *Smart Grid* hace referencia a redes de distribución en las que el flujo de energía es bidireccional, es decir, los consumidores pueden ser también generadores y ceder energía a la red eléctrica. El nombre viene del alto grado de tecnología usado en la red para mantener una correcta gestión de la misma, ya que se usan gran cantidad de equipos de medida inteligentes que pueden transmitir en tiempo real el consumo. [1]

Esto implica que la *Smart Grid* puede responder a eventos tales como el fallo de un transformador; o incluso modificar la curva de demanda rebajando el pico de consumo máximo, haciendo que los consumidores vuelquen energía a la red ("*peak shaving*"), y "*valley filling*", es decir, que gasten más energía cuando más barata y disponible es.

Las *Smart Grid* comenzaron con la idea de mejorar la infraestructura de medida de la demanda, para conseguir una mejor eficiencia energética, así como la auto reparación de la red y protección contra sabotajes y desastres naturales, en esta línea podemos comentar la instalación de contadores inteligentes en España, contribuyendo al desarrollo del llamado *Smart Metering*.

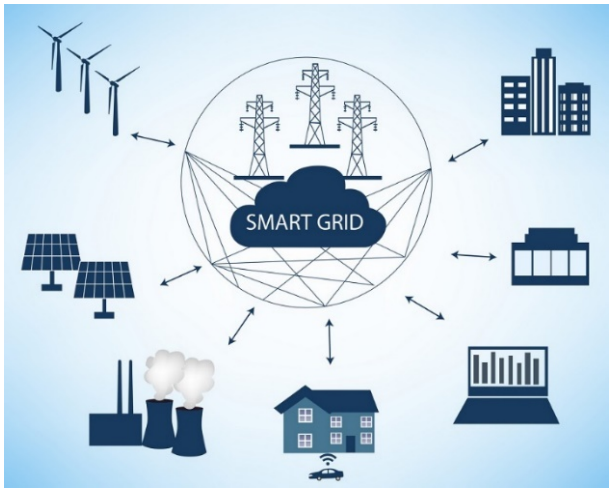

Figura 1: Ilustración de Smartgrid [2]

Otras ventajas de las *Smart Grids* son:

a) Reducción de la emisión de gases de efecto invernadero gracias a la mejor integración de fuentes de energía renovables y vehículos eléctricos.
b) Reducción del consumo de derivados de petróleo, ya que se reduce la necesidad de su uso para suplir los picos de consumo.
c) Menores pérdidas energéticas por transporte ya que la generación se lleva a cabo mucho más cerca de los centros de consumo.

Las redes tradicionales, intentan ajustar la producción a la demanda, pero esto aparte de ser costoso y poco práctico, puede llegar a ser insostenible en un futuro, ya que la demanda del consumidor no es fácilmente previsible, lo que provoca la necesidad de tener centrales de generación en modo *standby* para poder reaccionar a los cambios.

Se estima que el último 10 % de la capacidad de generación solo es requerido durante el 1% del tiempo, además el hecho de intentar hacer encajar la generación y el consumo puede dar lugar a variaciones locales de voltaje y cortes de suministro. [3]

Como contrapartida las *Smart Grid* proponen un sistema de respuesta a la demanda, es decir, intentan acoplar la demanda a las franjas de mayor generación disponible, por ejemplo, mediante tarifas de precio variables de modo que los consumidores voluntariamente cambien su consumo. También dentro de esta posibilidad está la tecnología *Vehicle to Grid* (V2G), que se basa en el uso masivo de los vehículos eléctricos como sistemas de almacenamiento para modificar el perfil de demanda energética. [3]

A medida que las reservas de combustibles fósiles se van acabando y se incrementa el precio de dichos combustibles, se espera una mayor utilización de las energías renovables en la producción energética. Las energías

renovables tienen una topología de generación distribuida (DG); sin embargo, la implementación de esta generación distribuida provoca grandes desafíos tales como la naturaleza fluctuante de la generación a través de energías renovables, lo que dificulta mucho su previsión. A pesar de ello, la futura *Smart Grid* tenderá a integrar una gran cantidad de esta generación distribuida.

Para solventar el problema de la generación fluctuante de estas fuentes de energía, se plantean medidas de mejora de previsión basada en algoritmos. También es muy importante la implantación de sistemas de almacenamiento, tales como sistemas de baterías e inversor trifásico, en los cuales puede modelarse sistemas de estabilización activa. [3]

El presente TFG se ocupa del análisis y verificación de un sistema compuesto por baterías e inversor trifásico, para comprobar si es posible su uso como sistema de estabilización de red activo.

El sistema de estabilización activo elegido es el VISMA (*Virtual Synchronous Machine*) o máquina síncrona virtual que será explicado en detalle en el capítulo 2.2.

1.1 Principios de funcionamiento del convertidor trifásico y baterías. Estudio de funcionamiento y requerimiento del grupo de baterías y convertidor

El funcionamiento básico del inversor trifásico se fundamenta en la conversión de voltaje continuo en corriente alterna. Para ello se basa en el uso de tiristores o transistores bipolares de puerta aislada (IGBT) para hacer fluir corriente desde el lado DC al AC en una secuencia determinada para emular la corriente alterna. Los inversores pueden ser monofásicos o trifásicos, los trifásicos proporcionan una salida de corriente alterna distribuida a tres fases desplazadas 120 grados eléctricos entre sí. El esquema de un inversor trifásico es el siguiente:

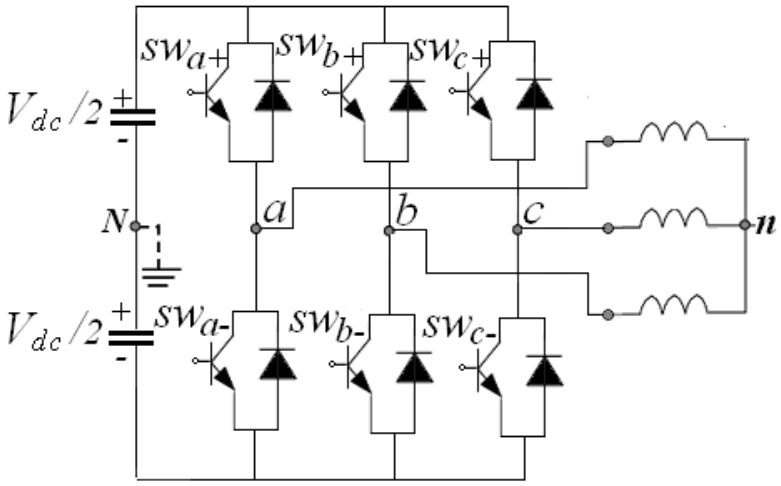

Figura 2: Esquema de inversor trifásico [4]

Como vemos en la imagen, se necesita una fuente de tensión continua que provee la energía necesaria, un circuito compuesto por componentes semiconductores y su correspondiente circuito de control, que comandará dichos componentes controlando la tensión de puerta para conseguir emular a la salida el comportamiento de un generador de corriente alterna.

Para conseguir esta conversión, existen diversos tipos de modulación:

1.1.1 Modulación SPWM

El proceso de conversión parte de una señal sinusoidal de muestra (señal moduladora) que es comparada con una señal portadora (normalmente triangular) de una frecuencia y amplitud mayores a la sinusoidal, mediante esta comparación se consigue una modulación de pulsos en anchura PWM, que van directamente a la puerta de los componentes semiconductores.

Como vemos en la figura 3, cuando el valor de la señal moduladora es mayor que el de la portadora, se activará el transistor correspondiente a un lado, y cuando sea menor, se activará el del otro lado. Este sistema de modulación se denomina SPWM o PWM sinusoidal. Dentro del sistema de modulación PWM sinusoidal, debemos distinguir entre inversores de un solo nivel de voltaje o multinivel (más de un nivel de voltaje de entrada), siendo los conversores multinivel los que mayor calidad de forma de onda de salida aportan [4]. En el caso de inversor de un solo nivel de voltaje a la entrada, el funcionamiento es el siguiente:

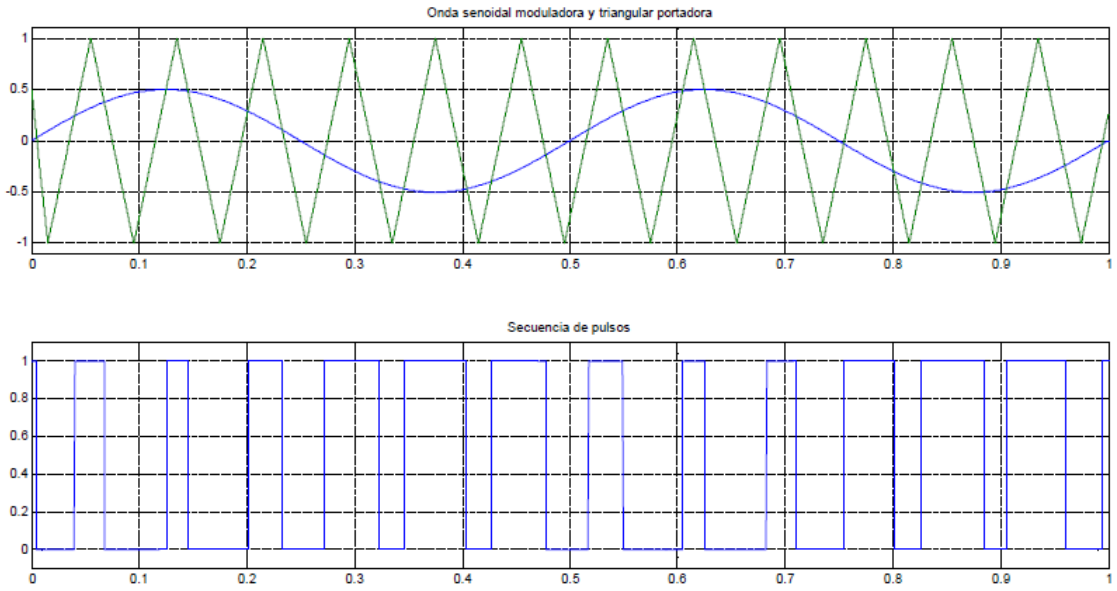

Figura 3: Señales de inversor [4]

Es importante la sincronización del sistema de control para que no se produzcan cortocircuitos entre los terminales positivo y negativo de la fuente de tensión continua.

En el caso de inversores multinivel (con varios niveles de voltaje de entrada), se necesita más de una portadora para realizar la comparación:

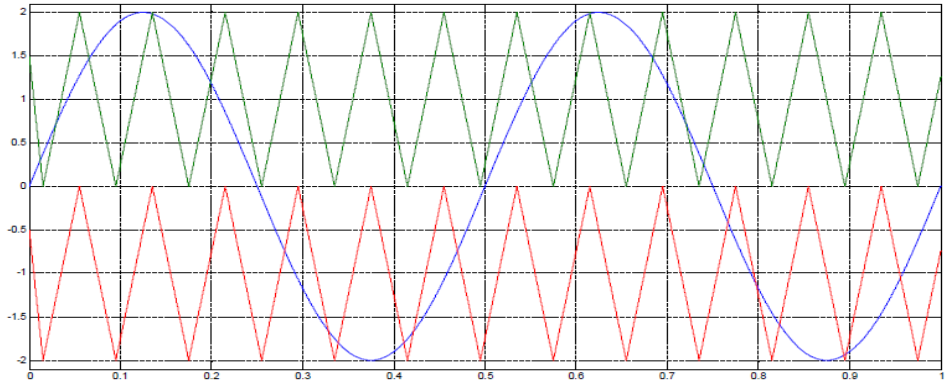

Figura 4: Señales inversor multinivel [5]

1.1.2 Modulación vectorial espacial (SVM)

Este tipo de modulación se basa en obtener la secuencia de disparo de los interruptores a partir de los voltajes del sistema trifásico mediante el uso de la transformada de *Park* (d-q), se pasa de los 3 vectores de voltaje trifásicos a su descomposición en vectores ortonormales, obteniendo un vector que gira en el plano a la misma frecuencia que la red. Esta técnica de modulación puede mejorar el contenido en armónicos de la señal de salida y minimizar las conmutaciones. [4]

El objetivo de la modulación vectorial espacial PWM, es conseguir, mediante la combinación óptima de vectores espaciales del inversor, la mejor aproximación al vector de referencia. Se intenta una representación muestreada del vector de referencia [5]. Es decir, la combinación de vectores del sistema trifásico, nos dará la secuencia de conmutación necesaria en los componentes semiconductores para conseguir de este modo una salida en corriente alterna.

Con cualquier tipo de modulación lo más importante para conseguir una adecuada calidad de onda a la salida es la frecuencia de conmutación, a mayor frecuencia, mayor calidad de onda a la salida.

Sin embargo, a la salida no dejamos de tener una señal cuadrada, que puede convertirse en sinusoidal mediante filtros de potencia. El proceso de filtrado de los armónicos más cercanos al fundamental requiere voluminosos condensadores y bobinas, que reducirán el rendimiento del sistema. Por esto, un objetivo a tener cuenta al diseñar un inversor es obtener señales de salida en las cuales los armónicos que aparecen sean de pequeño valor y estén lo más lejos posible del fundamental, esto se conseguirá aumentado la frecuencia de conmutación de los semiconductores y filtrando la señal obtenida. [6]

1.1.3 Funcionamiento de rectificador

El rectificador controlado permite obtener a partir de una tensión de red alterna, una onda de tensión continua cuyo valor medio puede ser controlado externamente. Para ello se usan tiristores o transistores controlados, cada tiristor puede ser disparado desde el momento en que su tensión de fase es la máxima. La tensión de salida contiene cierto rizado por lo que se hace necesario el uso de filtro a la salida: condensador.

El valor de la tensión de salida puede ser controlado mediante el ángulo de disparo de cada tiristor.

En la siguiente imagen se muestra el esquema eléctrico de un rectificador trifásico controlado.

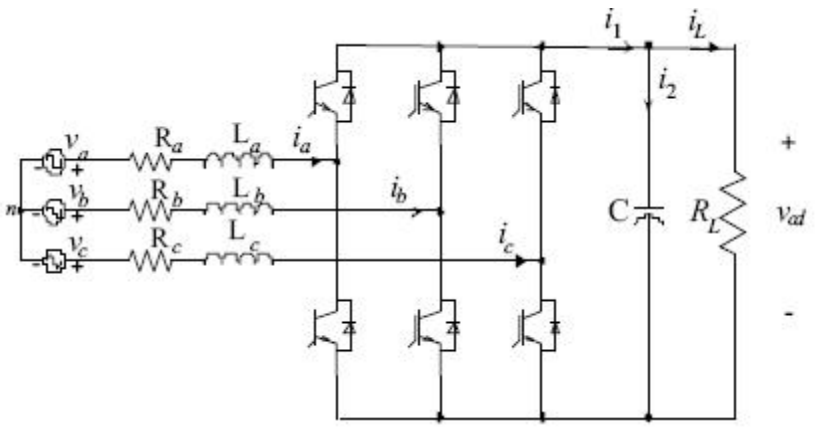

Figura 5: Esquema de rectificador trifásico[7]

1.1.4 Descripción de hardware utilizado

El sistema de baterías y conversor utilizados para la realización del presente trabajo es el siguiente:

Conversor: BAT-50 de la firma *WSTECH*. La característica principal de este conversor trifásico es que su potencia máxima es de 55 kW, y un voltaje DC en rango 530 - 890 V. También tiene la función opcional de red isla (aislada): generar su propia red trifásica. Dispone de comunicación vía Modbus TCP/IP con el conjunto de baterías. El conversor puede realizar la absorción y cesión tanto de potencia activa como reactiva.

Baterías: VU-CAB52: de la firma *Power Innovation*: compuesto por 5 celdas de ion de litio de 110 Vdc y 52 Ah, una batería de condensadores de 300 Wh y un conversor DC-DC (DCDC30k-400-600-k1), capaz de ofrecer un voltaje de salida de 300 a 750 V. El sistema de baterías contiene un modo de balance activo de celdas para garantizar un correcto funcionamiento en los ciclos de carga y descarga. Así mismo posee tarjeta de comunicación Modbus TCP/IP.

En las siguientes imágenes se puede observar a la izquierda el conversor BAT50 y a la derecha el sistema de baterías VU-CAB52:

Figura 6: Conversor BAT50

Figura 7: Baterías VU-CAB52

Para comandar el sistema compuesto por conversor y batería, se necesita una única conexión Modbus TCP/IP con la batería, ya que de esta parte otra conexión al conversor. De este modo solo es necesario transmitir los valores

deseados de potencia a la batería. El siguiente esquema muestra las conexiones Modbus entre los dos sistemas:

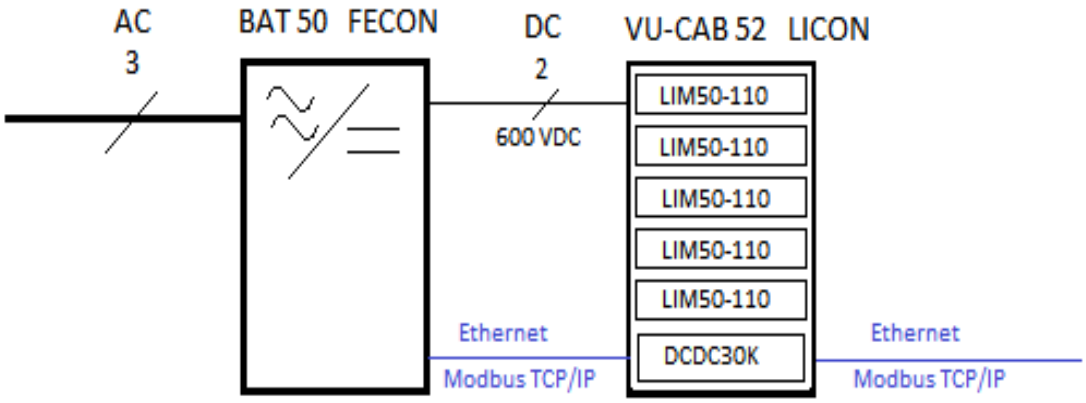

Figura 8: Conexiones entre batería y conversor

Importancia del balance de baterías:

La misión principal de este sistema de balance activo es garantizar una larga vida útil de las baterías, la carga uniforme de las diferentes celdas y hacer que toda la capacidad de la batería esté disponible. [7]

Además, asegura un correcto funcionamiento, impidiendo sobrecargas, sobretensiones, descargas demasiado profundas, etc. Asegurando que la batería trabaje siempre dentro de su zona de operación segura. [8]

Existen dos tipos de balance de baterías: balance activo y balance pasivo. En el caso de balance de baterías activo, se hace fluir la corriente de una celda a otra, o de la batería a una celda específica, mientras que, en el pasivo, se hace consumiendo la energía sobrante en resistencias. Se busca que todas las celdas de baterías tengan el mismo SOC (*State Of Charge*) durante la operación. [8]

1.1.5 Requerimientos de sistema de baterías e inversor

Inicialmente, se pidió a la empresa suministradora, un sistema de baterías e inversor con las siguientes características:

a) El sistema de almacenamiento debe estar conectado a la red trifásica mediante un conversor capaz de ceder y absorber tanto potencia activa como reactiva.
b) Las baterías deben estar compuestas por celdas de ion de litio, e incluir condensadores de alta capacidad.
c) La capacidad conjunta de las celdas de ion de litio debe ser de 30 kWh.
d) La capacidad de los condensadores será de 300 Wh.
e) Conversor de 50 kVA de potencia.
f) Sistema de baterías debe incluir un modo de *Battery-Management* (balance de baterías).
g) La comunicación con el sistema debe llevarse a cabo en un protocolo basado en Modbus, *ethernet*.
h) El tiempo de reacción del sistema debe ser inferior a 10 ms.

La necesidad de tales prestaciones viene del objetivo de modelar sistemas de estabilización de red en el conjunto batería-inversor y convertirlo de este modo en elemento activo dentro de las *Smart Grid*s.

Ya que el requerimiento del tiempo de respuesta de 10 milisegundos no ha sido verificado, y la empresa no ha podido asegurar su cumplimiento, el presente TFG brinda un análisis del tiempo de respuesta del sistema para comprobar si es válido para los objetivos deseados, y futuros modelados de sistema.

1.2 Introducción a Modbus TCP/IP

Modbus es un protocolo de comunicación creado en el año 1979, inicialmente concebido sobre bus serie. El protocolo se ha expandido para incluir aplicaciones como TCP/IP o UDP y hoy en día es un protocolo comúnmente utilizado para la automatización y comunicación industrial. Suele ser utilizado para la comunicación en red tipo SCADA, o control de PLCs.

Modbus fue diseñado como un protocolo de solicitud y respuesta; siguiendo una arquitectura de maestro y esclavo: el maestro realiza una petición y espera una respuesta del esclavo. Esto le da al maestro un control completo sobre el flujo de información y el comportamiento del esclavo.

1.2.1 Modelo de datos en Modbus

Originalmente solo soportaba dos tipos de datos: valor Booleano y un entero de 16 bits sin signo. Sin embargo, para cumplir con las necesidades modernas de tipos de datos, los valores de los datos Modbus se han divididos en cuatro rangos:

a) Bobinas (*Coils*): Es el tipo más sencillo, compuesto por un booleano de lectura y escritura por el maestro, esclavo puede realizar lectura o escritura. Bit simple.

b) Entradas discretas: Tipo booleano de solo lectura por el maestro, esclavo puede realizar lectura y escritura. Bit simple.

c) Registros de retención (*Holding register*): Tipo palabra sin signo, de lectura y escritura tanto por maestro como por esclavo.

d) Registros de entrada (*Input register*): Palabra sin signo de solo lectura por maestro y lectura y escritura por esclavo.

Modbus utiliza una representación *big-endian* para datos mayores de un byte, lo que significa que el byte más significativo es transmitido primero:

Por ejemplo, el valor hexadecimal 0x1234 es transmitido como 0x12 y 0x34. [9]

Al llevar Modbus sobre protocolo TCP/IP, se añade encabezado para permitir al receptor reconocer los límites del mensaje, y también hay un código de chequeo de error CRC-32 (*Cyclic Redundant Code*: verificación por

redundancia cíclica). Todas las solicitudes son enviadas vía TCP sobre el puerto registrado 502. [9]

1.2.2 Estructura de la trama Modbus

Los mensajes enviados en el protocolo Modbus, tienen la siguiente estructura de datos:

Posición del Byte	Significado
Byte 0	Identificador de transacción, byte alto
Byte 1	Identificador de transacción, byte bajo
Byte 2	Identificador de protocolo, byte alto (normalmente 0)
Byte 3	Identificador de protocolo, byte bajo (normalmente 0)
Byte 4	Campo de longitud, byte alto, = 0 ya que mensajes < 256
Byte 5	Campo de longitud, byte bajo (número de bytes que siguen)
Byte 6	Identificador de unidad (“dirección esclavo”)
Byte 7	Código de función de Modbus
Bytes 8 y siguientes	Datos transmitidos

Tabla 1: Estructura de trama Modbus

1.2.3 Códigos de función de Modbus

En el byte 7 tenemos el código de función de Modbus, que permite definir el tipo de mensaje: lectura o escritura de registro o bobina). En la siguiente tabla se describen los códigos de función Modbus que deben ir en el byte 7:

Código(decimal)	Código(hexadecimal)	Nombre	Función
2	0x02	*Read Discrete Inputs*	Leer un bit
3	0x03	*Read Holding Register*	Leer registro
4	0x04	*Read Input register*	Leer registro de entrada
5	0x05	*Write Single Coil*	Escribir bit
6	0x06	*Write Register*	Escribir registro
15	0x0F	*Write Multiple Coils*	Escribir varios bits
16	0x10	*Write Multiple Register*	Escribir varios registros

Tabla 2: Códigos de función Modbus

En esta aplicación, los códigos más usados son: 0x03 *Read holding register*, 0x04 *Read input register* y 0x06 *Write single/holding register*, ya que permiten el acceso a la información del estado del sistema y el control de todos los parámetros.

1.2.4 Ejemplo de mensaje en Modbus

En el siguiente ejemplo, se muestra un *frame* de Modbus para la consulta de la frecuencia de red que está midiendo el sistema:

frameFrecuencia = [0x00, 0x0a, 0x00, 0x00, 0x00, 0x06, 0x01, 0x04, 0x00, 0xa, 0x00, 0x01]

La siguiente tabla explica los bytes que componen el ejemplo anterior:

Posición	Valor hexadecimal	Valor decimal	Descripción
Bytes 0 y 1	0x00, 0x0A	10	Dirección del registro de frecuencia de red.
Bytes 2 y 3	0x00, 0x00	0	Identificador de protocolo.
Bytes 4 y 5	0x00, 0x06	6	Longitud en bytes de la trama que le sigue.
Byte 6	0x01	1	Identificador de unidad.
Byte 7	0x04	4	Código de función: Leer registro de entrada.
Bytes 8 y 9	0x00, 0x0A	10	Dirección del registro al que se quiere acceder: De nuevo registro 10.
Bytes 11 y 12	0x00, 0x01	1	Longitud de datos que se espera como respuesta a la petición.

Tabla 3: Ejemplo de mensaje Modbus

Para comprobar las tramas de mensajes enviadas y recibidas vía Modbus TCP/IP, y la detección de errores, se ha usado el software de análisis de redes *Wireshark*, desconectando el sistema batería-conversor de la red de comunicación existente y estableciendo una conexión directa con un ordenador; y una vez verificado su correcto funcionamiento, volviendo a poner el sistema en la red de comunicación.

2 Tiempo de reacción en el sistema de carga de baterías

2.1 Desarrollo teórico

Como se ha explicado anteriormente en la introducción, una de las características principales de las *Smart Grids* es su alta integración de energías renovables (distribuidas), esto puede llevar aparejados algunos problemas graves para la estabilidad y fiabilidad de la red eléctrica. La importancia de los sistemas activos de estabilización radica precisamente en este mayor porcentaje de generación distribuida respecto al total.

La forma tradicional de generación energética son las turbinas, estas pueden funcionar tanto con vapor procedente de una fuente de calor como con un flujo de agua procedente de una presa o acopladas al eje de un motor diésel. En las turbinas, se dispone de una gran masa rotórica que gira de forma constante a 50 Hz (en Europa), lo que define la frecuencia de la red eléctrica a la que está conectada. En la red eléctrica existen dos parámetros fundamentales a controlar: por un lado, el voltaje (que depende del consumo de potencia reactiva y tiene una variación de carácter local) y por otro la frecuencia (que depende del consumo de potencia activa y posee un valor igual en toda la red). [10]

Cuando se produce una sobrecarga de la red (aumenta rápidamente el consumo de energía), se genera una bajada de voltaje y frecuencia. Por una parte, la caída de voltaje puede compensarse de forma local, pero la caída de frecuencia repercute directamente sobre los generadores: al tener la mencionada masa rotórica girando a 50 Hz y demandarse un incremento de energía, se produce una disminución de la velocidad de giro del rotor, con lo que se cede energía a la red, es decir, se transforma parte de la energía almacenada en la inercia rotacional en energía eléctrica. Esto da lugar a que la bajada de frecuencia se suavice y se pueda aumentar la generación sin poner en peligro la estabilidad de la red; aumentando el margen temporal de reacción. Sin embargo, se debe limitar este caso ya que induce grandes tensiones mecánicas en los generadores.

El problema con el que se encuentran las redes que están implementando mayor producción mediante fuentes de energía renovables es que dichas fuentes no suelen disponer de una masa rotórica, en la mayoría de los casos la corriente alterna proviene de la inversión de corriente continua, por lo que, al producirse una sobrecarga en la red, no se dispone del tiempo de reacción anterior para incrementar la producción y la caída frecuencial resulta mucho

más acusada (en este caso no se dispone de tiempo suficiente para poner en marcha los generadores auxiliares a base de gas o diésel mayormente). Además, este problema se acentúa dada la característica de generación fluctuante y la dificultad de previsión de la generación renovable. La posible consecuencia de la inestabilidad frecuencial es un incremento de los cortes eléctricos ya que los sistemas de protección locales producirán un deslastrado de cargas cuando los parámetros de la red se vean alterados por encima del límite permitido y la consiguiente dificultad para reestablecer el régimen normal de funcionamiento, además del riesgo de fallo completo en la red ya que al deslastrar es difícil calcular la magnitud de la carga desconectada. Por otro lado, proporcionar una calidad de onda suficiente es algo primordial, ya que son numerosos los equipos sensibles a su variación y que usan su frecuencia como medida temporal.

Este problema de inestabilidad frecuencial es más acusado en redes pequeñas, como las insulares, ya que el interconexionado de redes las dota de mayor estabilidad. Además, los países con mayor implantación de energías renovables pueden aprovecharse de la inercia de sus países vecinos con mayor número de turbinas. A pesar de todo, el futuro cambio a energías sostenibles dará lugar a la disminución del número de turbinas y de ahí surge la necesidad de sistemas de estabilización. Además, debido a la creciente descentralización de la producción eléctrica se hace más difícil el control de dicha generación, fomentando el uso de sistemas automáticos de estabilización. Por otra parte la generación distribuida tiene ventajas como la menor pérdida de potencia durante el transporte. [11]

2.2 Algoritmos: Implementaciones.

El sistema de estabilización propuesto en el presente Trabajo Fin de Grado es una máquina síncrona virtual o VISMA (*Virtual Synchronous Machine*) también llamado VSM, que consiste en modelar el comportamiento de un generador síncrono en un inversor trifásico, de modo que pueda replicarse la estabilidad que aportan a la red esta clase de generadores. La máquina síncrona virtual (VISMA), se basa en el uso del interfaz de la electrónica de forma que se emule el funcionamiento de un generador tradicional, con la ventaja de que los parámetros como la inercia o el voltaje de la excitación son fácilmente variables. [12]

Por un lado, el VISMA puede actuar como estabilizador frecuencial, ya que, cuando la frecuencia de la red cae, el VISMA comienza a introducir potencia activa de forma inmediata para compensar dicho cambio [13]. Se controla ajustando el parámetro J (Inercia Virtual). [14]

Por otro lado, también puede compensar potencia reactiva, ya que puede monitorizar el voltaje de red y reaccionar ante él. En caso de una caída de voltaje, el VISMA comenzará a dar potencia capacitiva hasta que la caída de voltaje cese [15].

Otra ventaja de este sistema es que la potencia que se disiparía por rozamiento en el caso de un generador síncrono tradicional; en un VISMA puede ser reutilizada, ya que no hay partes móviles.

A parte de elemento estabilizador de red, el VISMA también puede controlar la carga y descarga del sistema de almacenaje, mediante el parámetro M (torque virtual) el cual es un parámetro de referencia del controlador. [14]

Por ello, un sistema de almacenamiento puede, si es correctamente programado, comportarse como un sistema de estabilización activo de red, y una vez comprobados sus efectos puede llegar a implementarse a gran escala. Aunque para ello debe cumplir ciertas características de funcionamiento: En especial, un tiempo de respuesta lo bastante bajo para poder asegurar el correcto funcionamiento del VISMA. Por ello, el presente Trabajo Fin de Grado propone y verifica un algoritmo dedicado a la medida de dicho tiempo de respuesta.

Funcionamiento del VISMA:

El VISMA se basa en una medición continua del voltaje de la red trifásica, con esta medida se calcula el valor teórico de la posición del rotor, es decir, se simula dónde estaría el rotor de una turbina para generar dicho voltaje, y mediante un algoritmo se consiguen imitar las corrientes estatóricas de un generador, comandando el inversor trifásico (normalmente compuesto por transistores IGBT). Inicialmente se utiliza la transformada de Park, para expresar los voltajes trifásicos de referencia en los ejes ortonormales *dq*, que permiten pasar de dinámico a estático.

Los parámetros de control del VISMA son análogos a los de una central eléctrica tradicional: Según los generadores clásicos, la generación de potencia activa puede ajustarse mediante el par mecánico aplicado al generador, mientras que la potencia reactiva puede controlarse con la corriente que circula por el devanado de excitación. Los parámetros de control del VISMA son el torque virtual y el voltaje de red. Si el torque se fija en un valor positivo, el VISMA actúa como un generador, inyectando potencia en la red; sin embargo, para un valor negativo, se comporta como un motor. [16]

Si el voltaje de red cae, mientras el voltaje virtual de excitación se mantiene, el VISMA comenzará a dar potencia reactiva capacitiva a la red. [15]

Una de las principales ventajas del VISMA es que su masa virtual puede fijarse en un valor concreto. La figura 9 presenta el esquema de funcionamiento del VISMA, por un lado, está la fuente de tensión continua (*DG generation units*) y el inversor compuesto por IGBT, y por otro la red eléctrica trifásica y la parte de medida y procesamiento compuesto por una DSP o similar, en la que se ha modelado el VISMA.

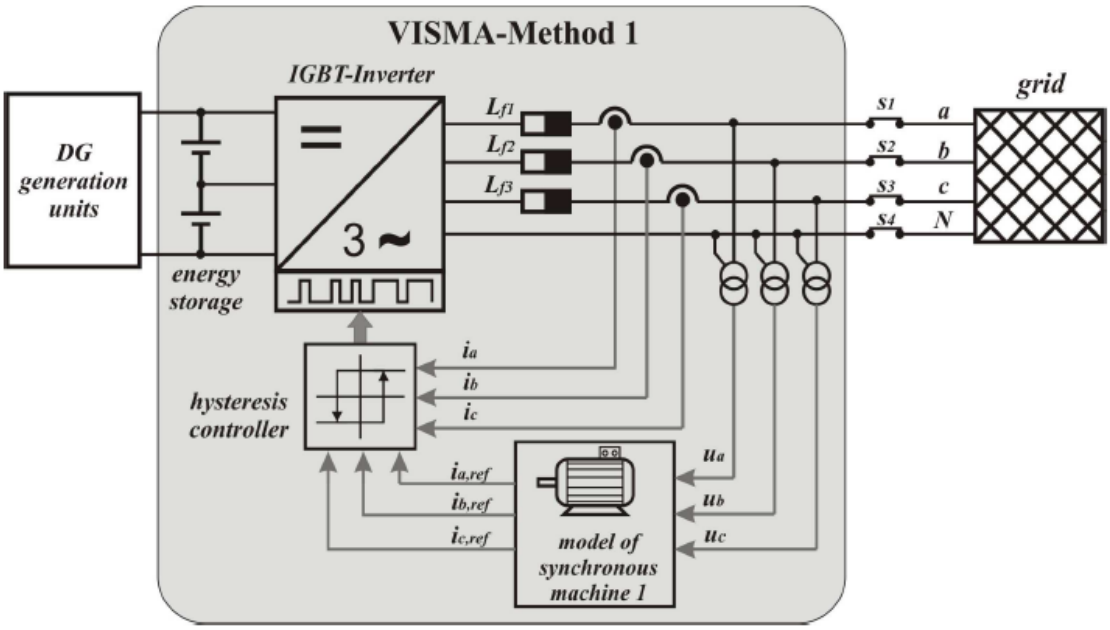

Figura 9: Esquema de VISMA [20]

Como se ha mencionado anteriormente el aspecto más importante del hardware usado para modelar el VISMA es el tiempo de reacción del mismo, ya que se necesitan tiempos de reacción bajos para poder reaccionar correctamente a las variaciones de potencia.

Los requerimientos de tiempo de respuesta para el sistema de almacenamiento y conversor se han fijado en 10 milisegundos. El análisis de dicho tiempo de respuesta se ha llevado a cabo en el siguiente apartado del trabajo.

3 Tiempo de respuesta

La definición del tiempo de respuesta del sistema se ha establecido como el tiempo invertido desde que el usuario comanda el sistema con una potencia de consigna que puede ser tanto positiva como negativa, hasta que la salida AC alcanza dicho valor de potencia, o un porcentaje preestablecido del mismo.

El algoritmo propuesto en este trabajo para la medida del tiempo de respuesta tiene por misión establecer una comunicación Modbus TCP/IP con el sistema de baterías y gestionar una conexión USB con el osciloscopio.

El software debe ser capaz de arrancar el sistema baterías-conversor, consultar y borrar los posibles errores y comandar la cesión o absorción de potencia a sus valores máximos, a la vez que realiza el disparo del osciloscopio vía puerto USB.

La configuración usada para la realización de la medida es el siguiente:

Como sistema de medida se ha elegido el osciloscopio *Agilent Technologies DSO-X 2004*, el cual permite su disparo a través del puerto USB que incorpora. En cuanto a las sondas de medida se han usado: una sonda diferencial de voltaje con capacidad de atenuación x100 y x1000, y una pinza amperimétrica con capacidad de atenuación x10 y x100. El osciloscopio se ha colocado en la salida AC del conversor trifásico(BAT50).

La disposición del aparato de medida se representa en las figuras 10 y 11:

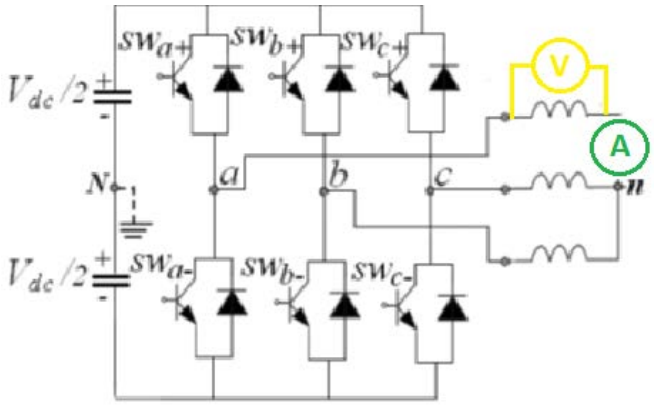

Figura 10: Esquema de medida

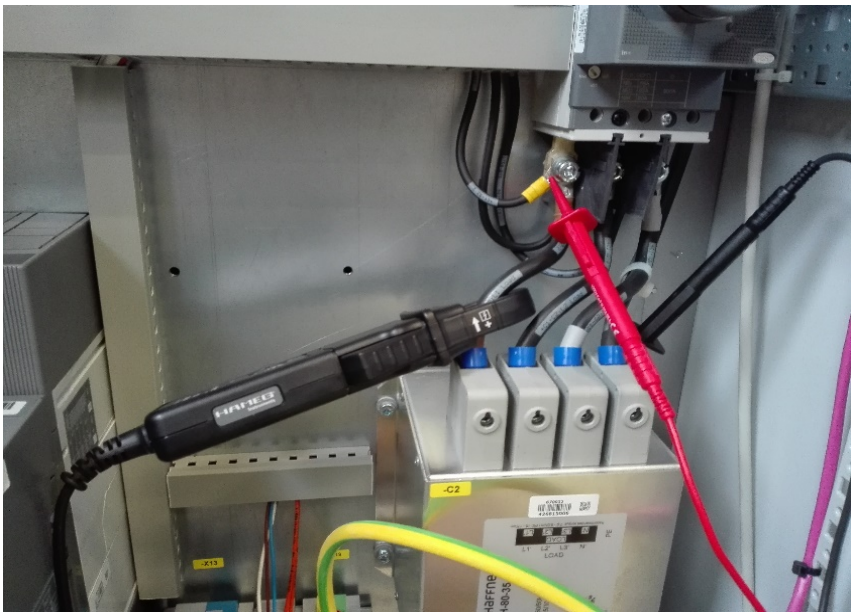

Figura 11: Disposición de las sondas de medida

La pinza amperimétrica se ha colocado en el cable de fase 1, y la sonda de voltaje se dispuso entre el cable de fase 1 y neutro. Ambos dispositivos han sido dispuestos antes del filtro de armónicos de salida por ser la zona más accesible para realizar la medida. Aunque la salida alterna posee ya la suficiente calidad de forma de onda.

Todas las medidas realizadas por el osciloscopio se guardan en una unidad de almacenamiento USB conectada a dicho aparato.

Con esta disposición, obtenemos una salida en el osciloscopio que tiene la siguiente forma. En la figura 12 se muestra la medida realizada desde 0 a -30 kW, es decir, se comanda el sistema para comenzar la carga de baterías a máxima potencia.

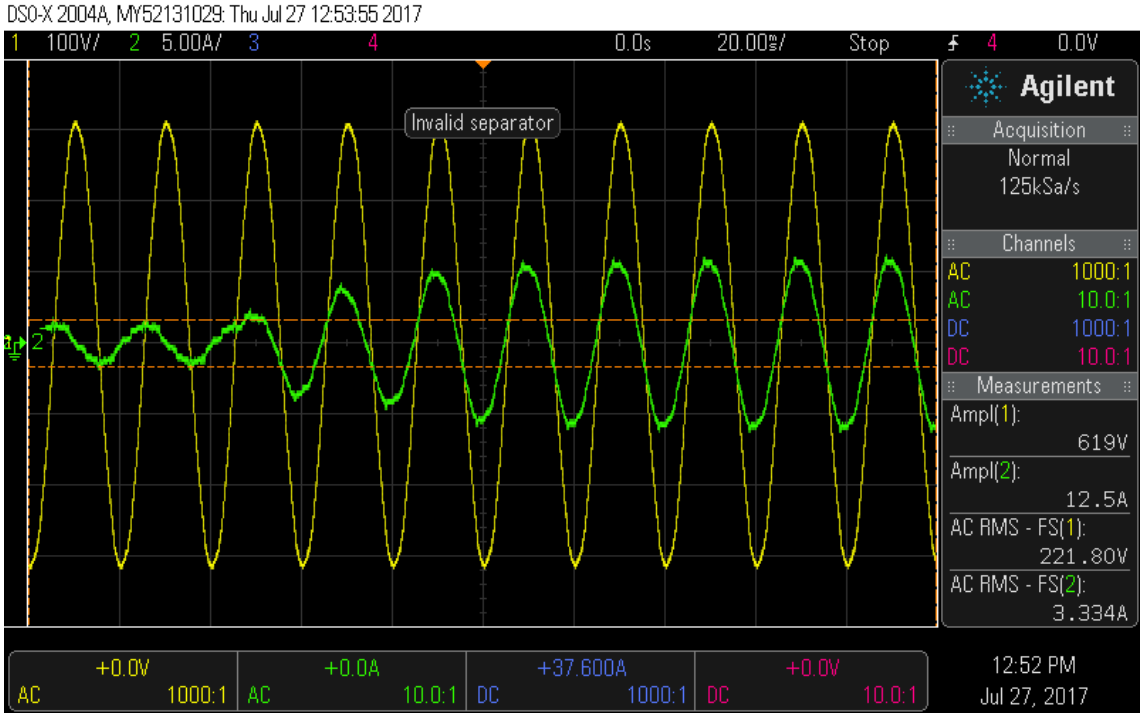

Figura 12: Señales de las sondas. Medida de 0 a -30 kW.

En la siguiente figura se comanda el sistema para la descarga de baterías:

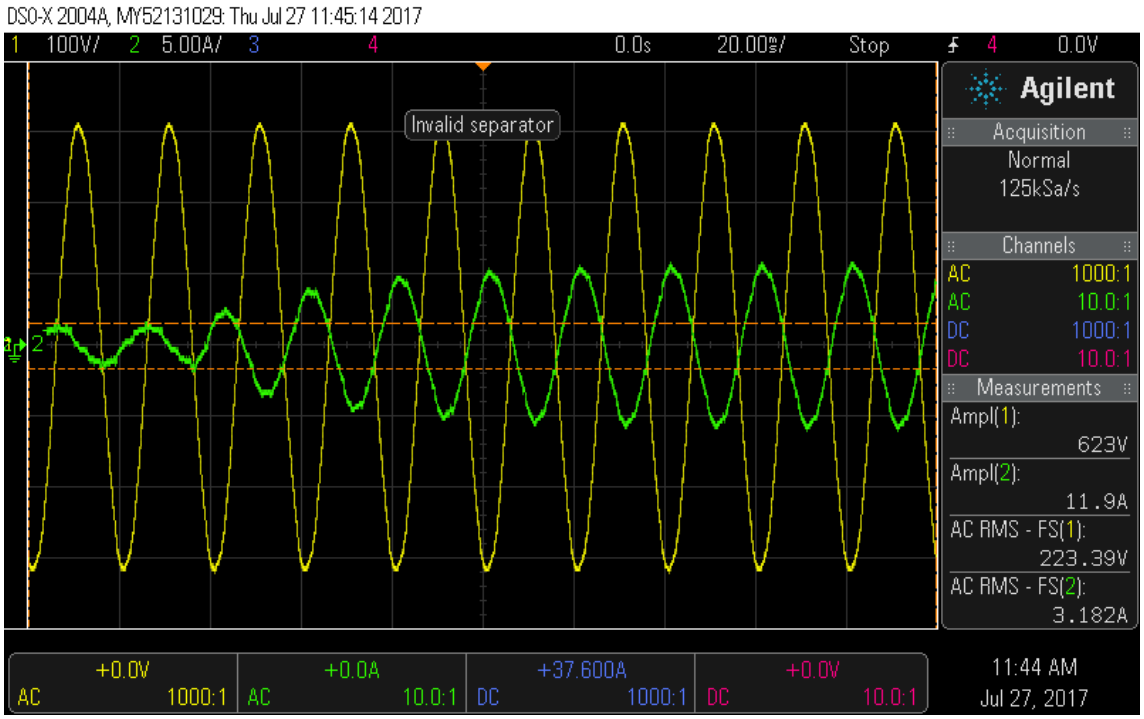

Figura 13: Señales de las sondas. Medida de 0 a 30 kW

La señal amarilla es la onda de voltaje y la verde es la señal de corriente. Se aprecia como hacia los 50 ms, el sistema se pone en marcha y se modifica el valor de la corriente y comienza a absorber potencia de la red en la figura 10, mientras que en la figura 11 se produce la cesión de potencia activa.

3.1 Definición

Ya que estamos ante un sistema de cesión y absorción de energía, se deben abordar los siguientes casos de funcionamiento:

En primer lugar, los tiempos de reacción para carga (absorción de potencia) y descarga (cesión de potencia) del sistema de baterías, empezando desde situación de reposo, es decir, sistema funcionando a 0 kW.

En segundo lugar, se deben analizar los tiempos de respuesta para la parada del sistema, es decir, desde una cesión o absorción de potencia hasta alcanzar los 0 kW.

Por último, se deben estudiar las interacciones entre procesos de carga y descarga, es decir, pasar de cesión a absorción de potencia y viceversa.

Con todo lo anterior, las medidas han sido divididas en 6 grupos que engloban todos los casos citados:

a) Desde 0 kW a cesión de potencia (descarga de baterías): “0 to P”
b) Desde 0 kW a absorción de potencia (carga de baterías): “0 to –P”
c) Desde cesión de potencia a 0 kW: “P to 0”
d) Desde absorción a 0 kW: “-P to 0”
e) Desde cesión a absorción de potencia: “P to –P”
f) Desde absorción a cesión de potencia: “-P to P”

El signo de la potencia se ha considerado acorde al que se debe introducir en el sistema de baterías.

Cabe destacar que en este análisis primario sólo se ha considerado la cesión y absorción de potencia activa. Pero el software propuesto, incluye la medida de potencia reactiva.

3.2 Aplicación

La aplicación propuesta en este Trabajo Fin de Grado debe ser capaz de realizar las siguientes especificaciones:

a) Establecer una conexión Modbus TCP/IP con el sistema de baterías.
b) Crear una conexión USB con el osciloscopio.
c) Consultar y borrar los errores en ambos sistemas.
d) Arrancar el sistema de baterías.
e) Comandar el sistema de baterías a valores máximos de funcionamiento.
f) Disparar y guardar datos del osciloscopio.
g) Medir el tiempo de ejecución del proceso de medida.
h) Apagar sistema de baterías.

Con todo lo anterior se diseña un algoritmo cuyos diagramas de flujo se presentan en el punto siguiente 3.2.1.

Aunque el código se encuentra en el Anexo 1, los rasgos principales del programa son:

Al arrancar el programa se crea un directorio en el que se guardan los archivos de tiempo de ejecución del programa.

Así mismo, debido a los problemas surgidos con la conexión Modbus con el sistema de baterías, se ha habilitado un hilo que comprueba la conexión cada segundo para evitar que se cierre el *socket*, ya que se pueden generar problemas de seguridad en el caso en que el *socket* se cierre mientras el sistema está funcionando a pleno rendimiento.

El almacenamiento de los datos producidos por el osciloscopio se lleva a cabo en un dispositivo de almacenamiento USB conectado al osciloscopio, los nombres son definidos por el usuario y el programa añade automáticamente el tipo de conexión establecida con la batería y los valores de potencia a los que se ha llevado a cabo la medida. Además, por cada medida se guardan 3 ficheros de resultados: en primer lugar, un fichero CSV de datos en el que se representan las señales captadas por el osciloscopio; un segundo fichero de texto en el que se muestra la configuración del osciloscopio en cuanto a atenuación; y por último una captura de pantalla del osciloscopio de modo que las señales sean fácilmente reconocibles.

Así mismo, el algoritmo incluye medida del tiempo de ejecución del proceso de medida, que será guardado en un fichero que se ubica dentro de una carpeta cuyo nombre es el modo de conexión (DIRECTA o MODBUS) y la fecha de la realización de la medida.

3.2.1 Diagramas de flujo de programa

En la siguiente imagen se muestra un diagrama de flujo de la función *main* del programa, que permite seleccionar la acción a realizar:

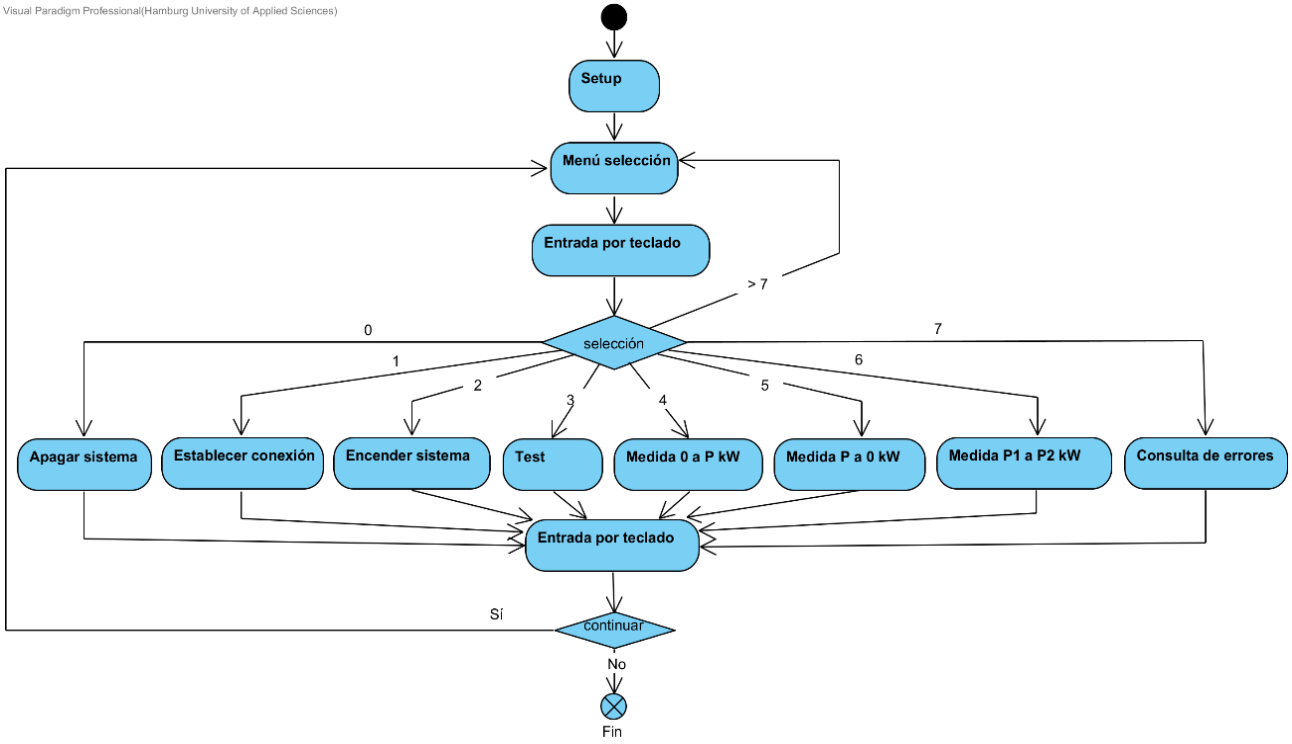

Figura 14: Diagrama de flujo de función Main

La función *main* se ocupa de la selección de función principal de programa, mientras que las selecciones de apagado y encendido se encuentran en funciones separadas.

En la figura 15 se muestra el diagrama correspondiente a la función de encendido de sistema, el orden de selección sigue el orden en el que se debe poner en marcha el sistema.

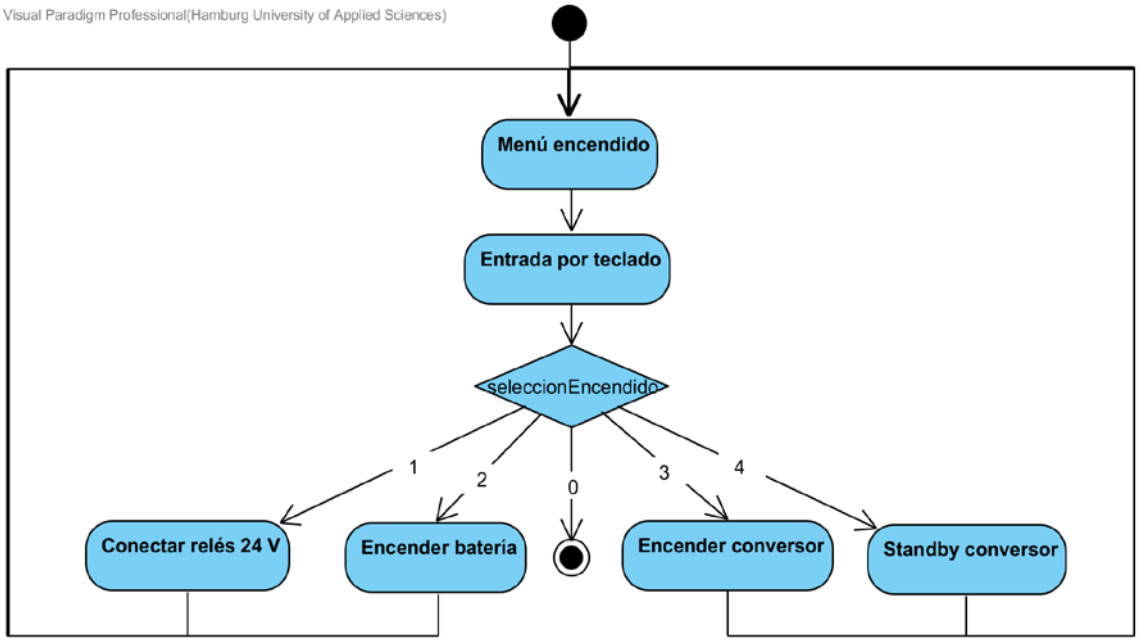

Figura 15: Diagrama de flujo de encendido

En la figura 17 se muestra el diagrama de flujo del apagado de sistema, de nuevo se sigue el orden establecido por la empresa para el apagado del hardware:

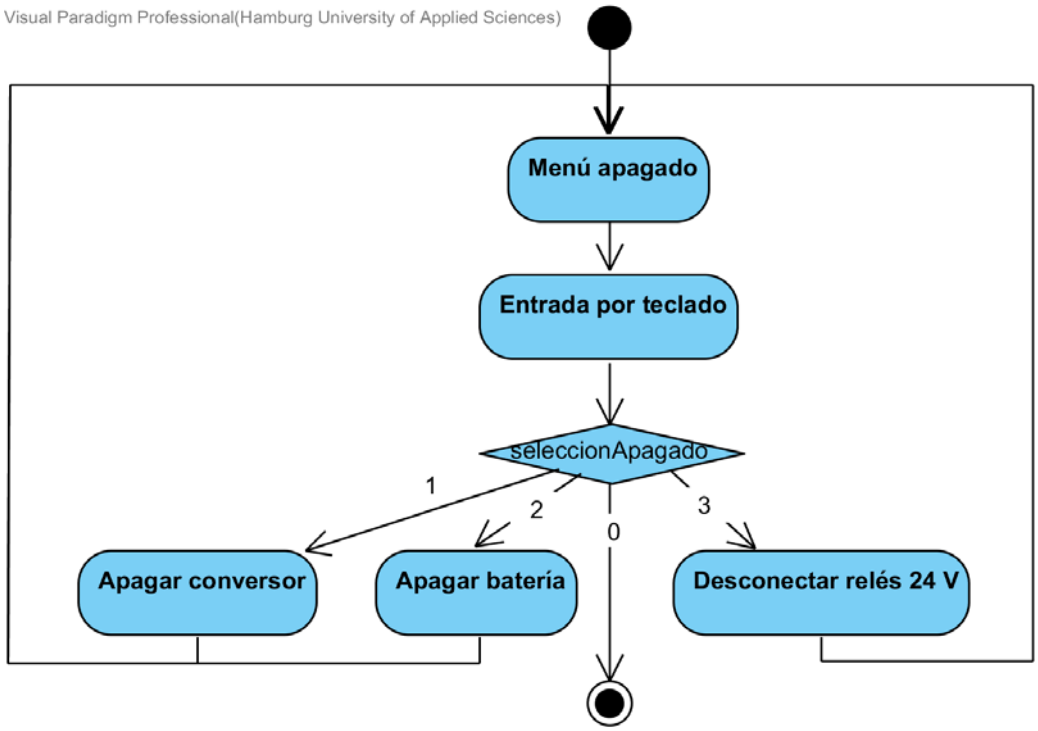

Figura 16: Diagrama de flujo de apagado.

En las figuras 17 y 18 se muestran los diagramas de flujo correspondientes al hilo que mantiene la conexión con la batería (izquierda) y a la función que realiza la medida (derecha), bloqueando el hilo cuando es necesario:

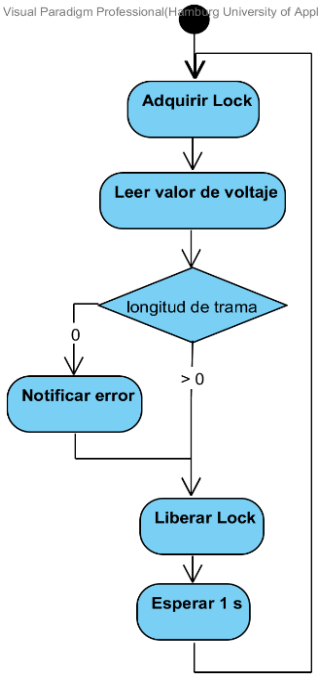

Figura 17: Diagrama de flujo de hilo de conexión

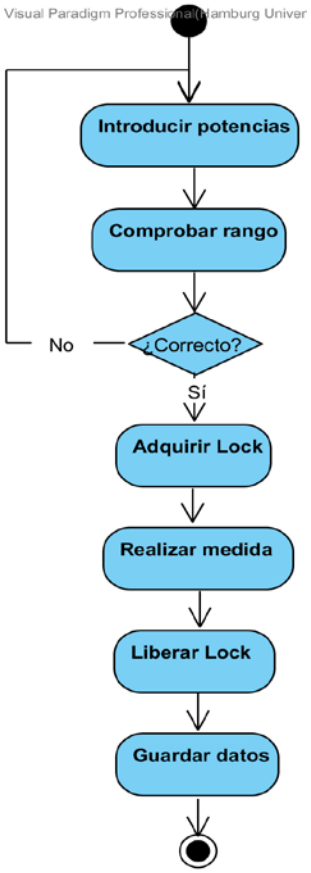

Figura 18: Diagrama de flujo de proceso de medida

Durante el proceso de medida es crítico el tiempo de ejecución, por ello se produce el bloqueo del hilo de conexión para no introducir retardos indeseados.

El proceso de medida parte de la comprobación del rango de las potencias introducidas por el usuario para asegurar que no se superan los valores máximos de funcionamiento, una vez hecho esto el programa envía el primer valor de potencia, tras lo que se produce una espera de 2 segundos para estabilizar la potencia, luego se adquiere el *Lock* que impedirá la ejecución del hilo, y se toma medida del instante temporal para calcular el tiempo de ejecución y por último, se envía el valor de potencia final y se dispara el osciloscopio. A continuación, se registra de nuevo el instante de tiempo y se

libera el *Lock*. Para terminar, se comanda de nuevo la batería a su estado de reposo.

Por último, se le pide al usuario que introduzca un nombre para los ficheros de datos generados y se envía al osciloscopio y se guarda el tiempo de ejecución asociado al nombre de dicha medida en un fichero, con lo que finaliza el proceso de medida.

3.3 Codificación

Para la codificación se ha usado el lenguaje de alto nivel *Python*, ya que dispone de librerías muy sencillas para establecer conexiones TCP/IP y acceso al puerto USB.

Así mismo, se ha elegido *Python* debido a su facilidad de sintaxis y la rápida curva de aprendizaje que posee.

Python es un lenguaje de programación interpretado, es decir se compila durante la ejecución.

Cabe reseñar que, a pesar de haberse intentado, no ha sido posible la implementación de una función para el ensamblado de todos los *frames* para enviar al sistema; debido a los problemas derivados de su implementación, se han definido los distintos *frames* como listas. A pesar de esto, se han implementado dos funciones para la construcción de los mensajes de potencia activa y reactiva.

3.4 Librerías utilizadas

3.4.1 Librería *socket*

Usada para establecer la conexión TCP con el sistema de baterías, sobre el puerto registrado 502, que corresponde al protocolo Modbus.

El *socket* es el método de comunicación entre una aplicación cliente y una aplicación servidor. El *socket* se define como el punto final de la conexión. [17]

Las funciones utilizadas de esta librería se exponen a continuación:

Función	Declaración	Descripción
socket()	self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)	Crea un nuevo *socket* de conexión
connect_ex()	estadoConexion = sock.connect_ex(self.server_address)	Conecta a la dirección indicada y devuelve el estado de conexión
sendall()	self.sock.sendall(s_string)	Envía trama a través del *socket*.
recv()	data= self.sock.recv(33)	Recibe datos a través del *socket*
close()	self.sock.close()	Cierra el *socket*

Tabla 4: Uso de librería socket.py

3.4.2 Librería *threading*

El *thread* o hilo, es un subproceso que permite la ejecución en paralelo de varios procesos dentro de la misma aplicación. [18]

La librería *threading* permite el uso de hilos. En este algoritmo se ha usado para configurar un hilo de conexión que impida la cancelación del *socket* de comunicación.

Las funciones utilizadas de esta librería se exponen en la tabla 5.

Función	Declaración	Descripción
Thread()	self.comprobarConexion_= threading.Thread(target= self.comprobarConexionHilo,args = (lock,))	Crea un nuevo *thread* cuyo objeto es la función comprobarConexionHilo y le pasa el argumento *lock*
RLock()	lock = threading.RLock()	Crea el *lock*
acquire()	lock.acquire()	Adquiere el lock para impedir que pueda ser usado
release()	lock.release()	Liberar el *lock* para permitir que otra parte del programa lo use
isAlive()	self.comprobarConexion_.isAlive()	Devuelve *True* si el hilo está activo

Tabla 5: Uso de librería threading.py

3.4.3 Librería *os*

Librería para manejo de directorios y archivos en el ordenador.

Las funciones utilizadas son las siguientes:

Función	Declaración	Descripción
gtcwd()	os.getcwd()	Obtiene el directorio de trabajo actual
listdir()	os.listdir(currentDirectory)	Crea una lista de los subdirectorios del directorio actual
mkedir()	os.mkdir(base_path)	Crea el nuevo directorio “base_path”
chdir()	os.chdir(base_path)	Cambia el directorio de trabajo al introducido.

Tabla 6: Uso de librería os.py

3.4.4 Librería *usbtmc*

Librería para la conexión USB con dispositivos categorizados como "tmc": *Test and Measurement Class*. En este caso para el osciloscopio.

Las funciones utilizadas de esta librería se exponen a continuación:

Función	Declaración	Descripción
instrument()	self.instr = usbtmc.instrument(id1, id2)	Crea un nuevo instrumento con la dirección especificada. Busca el dispositivo
ask()	self.instr.ask("*idn?")	Pregunta al dispositivo su dirección específica y la imprime por pantalla.
read_stb()	self.instr.read_stb()	Consulta los bits de estado del dispositivo.
write()	self.instr.write("*TRG")	Envía "*TRG" al dispositivo, lo que provoca su disparo

Tabla 7: Uso de librería usbtmc.py

3.4.5 Librería *time*

Librería para la gestión de tiempo. Usada para incluir fecha y hora en los archivos, medidas de tiempo de ejecución e incorporación de retardos y esperas en el algoritmo.

Función	Declaración	Descripción
asctime()	time.asctime()	Obtiene hora actual en formato: *Día semana, mes, nº mes, horas:minutos:segundos*
sleep()	time.sleep(1)	Detiene la ejecución del proceso durante el tiempo especificado.
time()	tiempo_final = time.time()	Obtiene los segundos transcurridos desde el 1 de enero de 1970.
strftime()	time.strftime("%h-%m-%s")	Obtiene hora actual en formato *horas-minutos-segundos*

Tabla 8: Uso de librería time.py

3.4.6 Librería auxiliar *errores*

Esta librería ha sido definida por el usuario y contiene todos los errores que pueden surgir en el sistema, asociados a su correspondiente número de identificación.

También se han incorporado los avisos o *warnings*, también con sus números de identificación. Aunque la aplicación utiliza únicamente los errores. El código de la librería se encuentra disponible en Anexo 2.

3.5 Clases de la aplicación

Se ha dividido la programación en dos clases: clase batería y clase osciloscopio:

a) **Clase batería**: representa el sistema de almacenamiento energético (batería y conversor), e incluye todas las funciones asociadas al mismo. Por ejemplo, el manejo del *socket*, *thread* de conexión y todas las consultas y escrituras de datos.

b) **Clase osciloscopio:** se ocupa de manejar dicho dispositivo, incluyendo su conexión a través del puerto USB y las operaciones de disparo y guardado de datos.

4 Resultados y medidas

Una vez realizadas las medidas usando el software propuesto (disponible en Anexo 1), se obtienen ficheros de datos CSV(*Comma Separated Value*) procedentes del osciloscopio. Cada fichero incluye la referencia temporal y todas las señales acopladas a los canales del osciloscopio. En este caso, consta de: valor temporal, voltaje y corriente a la salida del sistema.

Se han divido en dos tipos de medidas:

1) Medidas llevadas a cabo mediante la conexión directa entre ordenador y sistema.
2) Medidas tomadas mediante la conexión del sistema y ordenador a la red Modbus.

De este modo pueden apreciarse las diferencias en tiempo de respuesta entre una conexión directa al sistema, y una conexión a través de una red Modbus.

Dentro de cada uno de estos dos grupos, se han realizado las siguientes medidas:

a) Desde 0 kW a potencia de descarga de batería.
b) Desde 0 kW a potencia de carga.
c) Desde descarga a 0 kW.
d) Desde carga a 0 kW.
e) Desde descarga a carga.
f) Desde carga a descarga.

Todas las medidas se han realizado con una escala temporal de medida de 200 milisegundos.

Con estos datos se lleva a cabo un análisis, usando el programa Excel, ya que incorpora numerosas funciones de tratamiento de datos.

4.1 Procedimiento del análisis

En primer lugar, se han importado los ficheros CSV a ficheros Excel para hacer más sencillo su manejo.

Primero se ha calculado la potencia aparente instantánea como producto de voltaje y corriente. Luego se ha llevado a cabo una media móvil de orden 100, es decir de periodo 10 milisegundos, sobre este valor de potencia aparente para hallar la potencia activa. En último lugar, se compara dicha potencia activa con su valor nominal final y se aplica un algoritmo de búsqueda que encuentra el instante temporal en el que la potencia activa llega a un porcentaje establecido de la potencia de referencia. Se han establecido los porcentajes de referencia en 80, 85, 90 y 95.

En la siguiente figura se presenta el análisis llevado a cabo de forma gráfica:

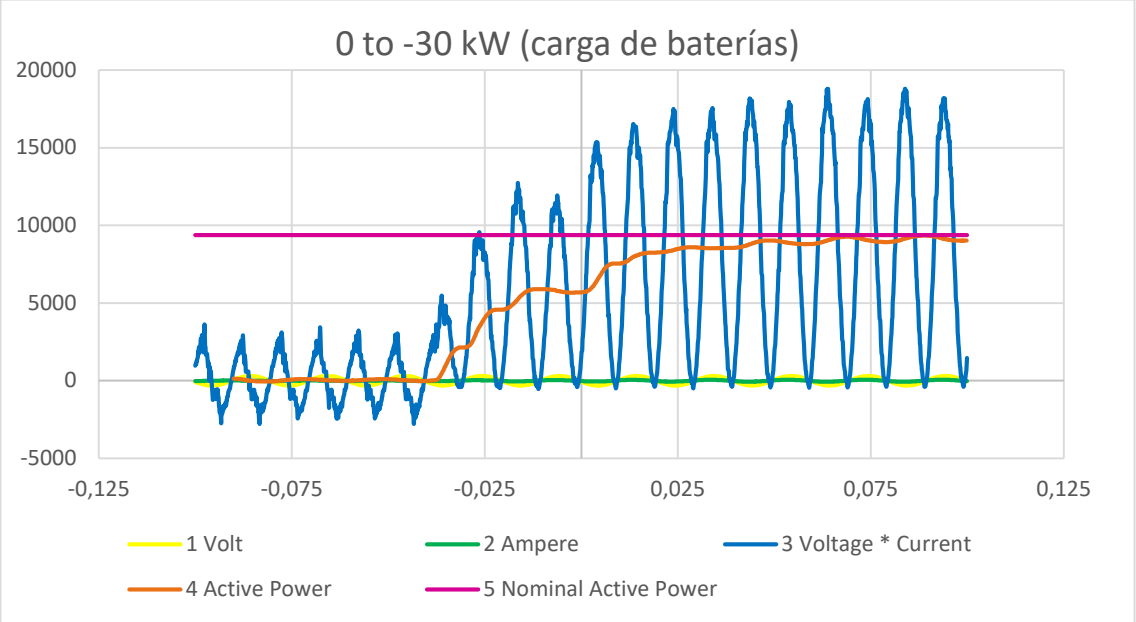

Figura 19: Análisis gráfico de tiempo de respuesta

Como se aprecia en la figura 19, la señal azul (3) es la potencia aparente instantánea, mientras que la naranja (4), es la media móvil, es decir, la potencia activa. Por último, la señal morada (5) se corresponde con la potencia nominal: el valor final de potencia activa al que debe llegar el sistema.

Al ser un sistema trifásico, la medida de la potencia suministrada a una de las líneas es un tercio de la potencia de consigna, en este caso al comandar el sistema con 30 kW, la potencia por cada línea será de 10 kW. Así mismo, aunque numéricamente la potencia es positiva, el signo de la potencia de consigna transferida al controlador debe ser negativo, de ahí el título.

En la siguiente tabla se presentan los valores de tiempo de respuesta del sistema para la carga de baterías desde 0 a 30 kW del gráfico anterior, para los diferentes porcentajes de la potencia de referencia establecidos: 80, 85, 90 y 95%:

BDIREC_0_to_-30_kw	80%	85%	90%	95%
Tiempo de respuesta (ms)	102,7	108,7	125,4	184,2

Tabla 9: Ejemplo de tiempos de respuesta para carga de baterías (ms)

Por último, se realiza un análisis estadístico de los datos conforme a una distribución normal.

Como se puede apreciar los tiempos de respuesta obtenidos en este caso son muy superiores al criterio de tiempo de respuesta establecido (10 milisegundos), sin embargo, en los siguientes puntos se procederá al análisis estadístico de los mismos para obtener resultados consistentes.

4.2 Tiempo de respuesta para carga de baterías

Este grupo de medidas se ha llevado a cabo desde el estado de reposo (0 kW) hasta la carga de baterías.

4.2.1 Conexión directa

En primer lugar, se ha analizado el tiempo de respuesta de la conexión directa a la batería, obteniéndose los resultados mostrados en tabla:

DIRECTA	80%	85%	90%	95%
DIREC_0_to_-5_kw	80,1	96,9	99,6	138,2
DIREC_0_to_-10_kw	71,8	75,8	93,8	137,1
DIREC_0_to_-15_kw	95,7	99,2	118,8	178
DIREC_0_to_-20_kw	121,3	139,2	159,9	>200
DIREC_0_to_-25_kw	136,3	154	174,6	>200
DIREC_0_to_-30_kw	59,3	61,3	81,7	123,5
DIREC_0_to_-30_kw_2	119,8	128,9	150,8	194,1
DIREC_0_to_-30_kw_3	103,1	105,9	134,5	177,2
DIREC_0_to_-30_kw_4	102,7	108,7	125,4	184,2

Tabla 10: Tiempos de carga, conexión directa (ms)

Como se observa, para cada medida existen 4 valores de tiempo de respuesta, que corresponden cada uno a un porcentaje de potencia final. Por ejemplo 80 % corresponde al tiempo de respuesta para que el sistema llegue al 80 % de la potencia final requerida.

El tiempo de ejecución del programa ha sido desechado en comparación con el tiempo de respuesta total, la justificación se encuentra en el apartado 4.9.

También es preciso resaltar que las medidas se llevaron a cabo con distintos valores de potencia final, comenzando en 5 kW, hasta el máximo de 30 kW.

4.2.2 Conexión a través de red Modbus

En la siguiente tabla se muestran los tiempos de respuesta para la conexión a través de la red Modbus:

MODBUS	80%	85%	90%	95%
MODB_0_to_-15_kw	87,3	104,5	124,7	146,9
MODB_0_to_-20_kw	109,1	112,1	130,7	151,8
MODB_0_to_-25_kw	42,8	46	63,8	84,6
MODB_0_to_-25_kw_2	57,6	71,2	79	99,2
MODB_0_to_-30_kw	67	69,4	74,5	101
MODB_0_to_-30_kw_2	107,6	113,5	124,4	145,2
MODB_0_to_-30_kw_3	138,4	146,5	165,7	187,4

Tabla 11: Tiempos de carga conexión Modbus (ms)

Se puede apreciar que no existen grandes diferencias entre las medidas realizadas con conexión directa y mediante la red Modbus, por lo que se pueden agrupar para extraer el tiempo de respuesta final.

4.2.3 Comparativa

En el siguiente paso, se seleccionan las medidas del funcionamiento máximo del sistema: cargar las baterías a 30 kW, y se extrae la media y varianza, entre ambos tipos de medidas: Modbus y conexión directa.

COMPARATIVA	80%	85%	90%	95%
DIREC_0_to_-30_kw	59,3	61,3	81,7	123,5
DIREC_0_to_-30_kw_2	119,8	128,9	150,8	194,1
DIREC_0_to_-30_kw_3	103,1	105,9	134,5	177,2
DIREC_0_to_-30_kw_4	102,7	108,7	125,4	184,2
MODB_0_to_-30_kw	67	69,4	74,5	101
MODB_0_to_-30_kw_2	107,6	113,5	124,4	145,2
MODB_0_to_-30_kw_3	138,4	146,5	165,7	187,4
MEDIA	99,70	104,89	122,43	158,94
DESVIACIÓN	27,95	30,43	33,64	36,10

Tabla 12: Comparativa de tiempos de carga (ms)

Una vez hecho esto, se puede proceder a realizar una distribución normal con dicha media y varianza, lo que resulta en la probabilidad de ocurrencia de un tiempo de respuesta, para un determinado porcentaje de la potencia máxima.

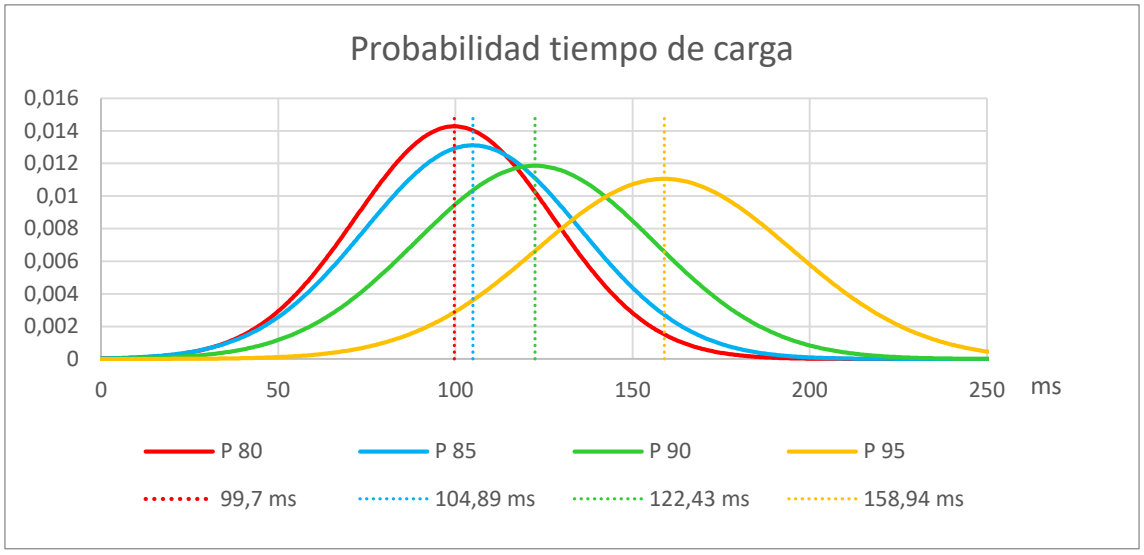

Figura 20: Probabilidad de tiempo de carga

Como vemos en la figura 20, para el 80% la mayor probabilidad de ocurrencia recae en un tiempo de respuesta de 99,7 ms. A medida que aumenta el porcentaje de potencia final, la varianza es mayor, por lo que la probabilidad de ocurrencia se distribuye más.

4.3 Tiempo de respuesta para detención de carga de sistema

Este grupo de medidas se ha llevado a cabo desde el sistema cargando las baterías hasta la detención de la carga

4.3.1 Conexión directa

DIRECTA	80%	85%	90%	95%
DIREC_-5_to_0_kw	51,9	55,8	57,1	58,4
DIREC_-10_to_0_kw	22	22,7	27,9	29,5
DIREC_-15_to_0_kw	43,6	44,7	46,2	65,1
DIREC_-20_to_0_kw	110,2	112,8	120,8	130,2
DIREC_-25_to_0_kw	140,6	142,1	157,4	160,7
DIREC_-30_to_0_kw	47,5	49,1	64	69,5
DIREC_-30_to_0_kw_2	68,5	70,7	79,1	88,4
DIREC_-30_to_0_kw_3	75,5	76,8	91,4	95,8

Tabla 13: Tiempos de detención de carga conexión directa (ms)

4.3.2 Conexión a través de red Modbus

MODBUS	80%	85%	90%	95%
MODB_-10_to_0_kw	58,8	59,3	60,1	60,9
MODB_-15_to_0_kw	102	110,8	113,3	122,9
MODB_-20_to_0_kw	81,8	82,8	84,1	101,1
MODB_-25_to_0_kw	101,7	109,7	117,6	121,9
MODB_-30_to_0_kw	84,3	85,7	99,4	104

Tabla 14: Tiempos de detención de carga conexión Modbus (ms)

4.3.3 Comparativa

Al no haber grandes diferencias entre las medidas directas y a través de la red Modbus, de nuevo se agrupan ambas medidas, seleccionando únicamente las llevadas a cabo con el sistema a pleno rendimiento (30 kW):

COMPARATIVA	80%	85%	90%	95%
DIREC_-30_to_0_kw	47,5	49,1	64	69,5
DIREC_-30_to_0_kw_2	68,5	70,7	79,1	88,4
DIREC_-30_to_0_kw_3	75,5	76,8	91,4	95,8
MODB_-30_to_0_kw	84,3	85,7	99,4	104
MEDIA	68,95	70,58	83,48	89,43
DESVIACION	15,69	15,59	15,44	14,73

Tabla 15: Comparativa de tiempos de detención de carga (ms)

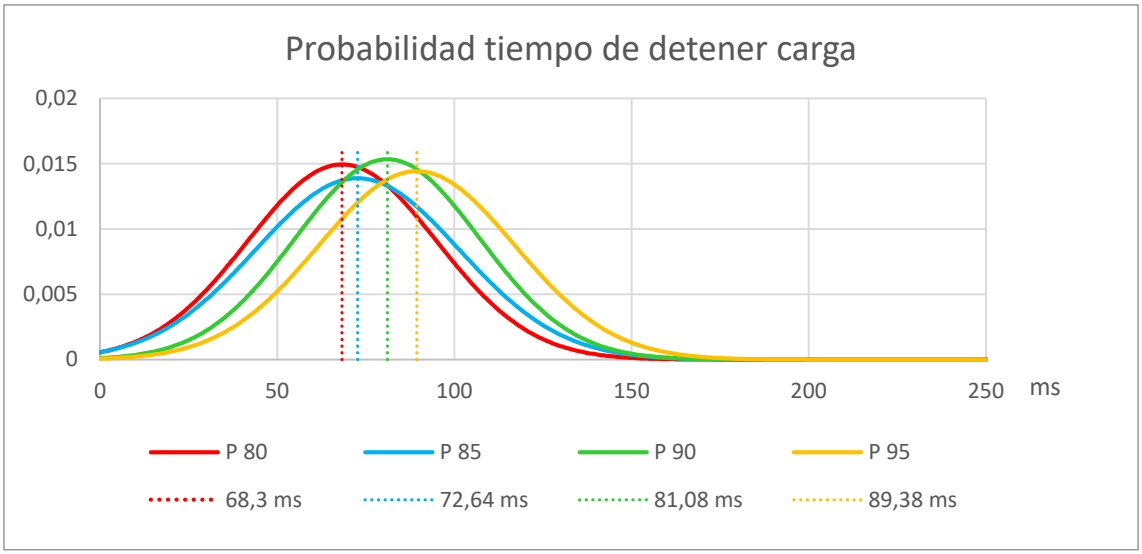

Figura 21: Probabilidad de tiempo de detención de carga

Como se aprecia en la figura 21, la detención de carga del sistema es significativamente más rápida que la carga, obteniéndose una probabilidad máxima al 80 % en 68,95 milisegundos.

4.4 Tiempo de respuesta para descarga de baterías

Este grupo de medidas se ha llevado a cabo comandando el sistema para que inicie la descarga desde una situación de reposo (0 kW).

4.4.1 Conexión directa

DIRECTA	80%	85%	90%	95%
DIREC_0_to_10_kw	138,7	158,7	180,4	198,7
DIREC_0_to_20_kw	14,8	32,8	55	114,2
DIREC_0_to_25_kw	16	34,2	56,4	107,1
DIREC_0_to_30_kw	92,3	110,6	133,1	192,2
DIREC_0_to_30_kw_2	72,6	90,2	117,7	181

Tabla 16: Tiempos de descarga conexión directa (ms)

4.4.2 Conexión a través de red Modbus

MODBUS	80%	85%	90%	95%
MODB_0_to_15_kw	92	96,9	115,5	135,1
MODB_0_to_20_kw	55,8	65,5	82,1	102,7
MODB_0_to_25_kw	63,9	73,9	86	105,3
MODB_0_to_30_kw	114,6	129	141,3	167,7
MODB_0_to_30_kw_2	70,7	81,1	92,5	119,7
MODB_0_to_30_kw_3	100,6	115,3	126,7	148,4

Tabla 17: Tiempos de descarga conexión Modbus (ms)

4.4.3 Comparativa

De nuevo para la comparativa se agrupan las medidas de potencia máxima:

COMPARATIVA	80%	85%	90%	95%
DIREC_0_to_35_kw	92,3	110,6	133,1	192,2
DIREC_0_to_40_kw	72,6	90,2	117,7	181
MODB_0_to_30_kw	114,6	129	141,3	167,7
MODB_0_to_30_kw_2	70,7	81,1	92,5	119,7
MODB_0_to_30_kw_3	100,6	115,3	126,7	148,4
MEDIA	90,16	105,24	122,26	161,8
DESVIACION	18,69	19,39	18,75	28,63

Tabla 18: Comparativa de tiempos de descarga (ms)

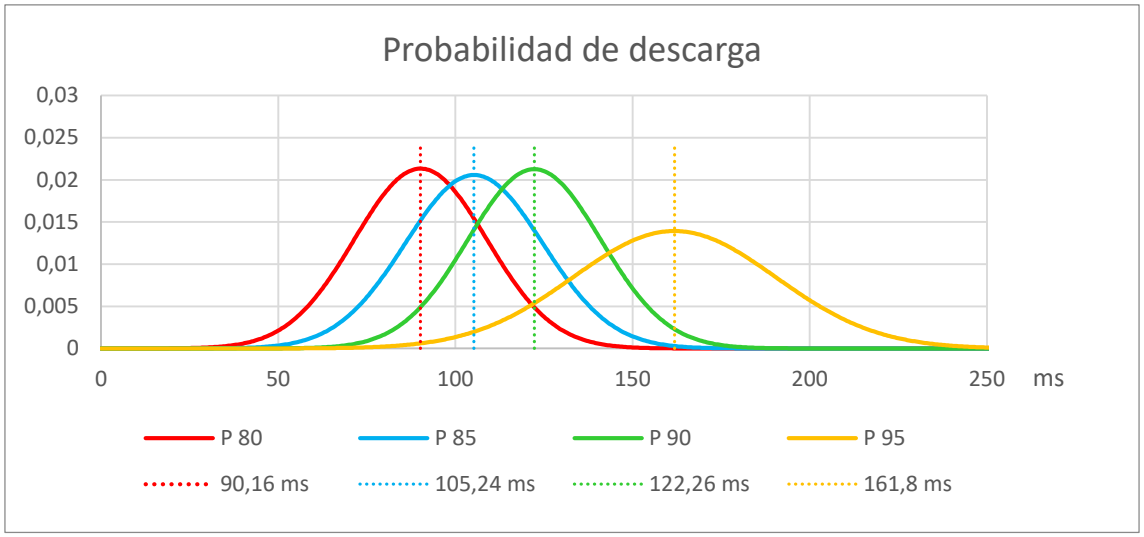

Figura 22: Probabilidad de tiempos de descarga

En la figura 22 se aprecia que el tiempo de respuesta de descarga es ligeramente superior al de carga, aunque las probabilidades se encuentran más distribuidas. También se muestra que el sistema tarda más tiempo en alcanzar el 95 % de la potencia final.

4.5 Tiempo de respuesta para detención de descarga del sistema

Este grupo de medidas se obtiene comandando el sistema para detener la descarga de baterías.

4.5.1 Conexión directa

DIRECTA	80%	85%	90%	95%
DIREC_5_to_0_kw	114,2	115,2	116,1	116,9
DIREC_15_to_0_kw	76,1	77,2	78,7	96,5
DIREC_15_to_0_kw	166,7	167,9	169,3	187
DIREC_20_to_0_kw	94,6	96,9	103,5	108,6
DIREC_30_to_0_kw	45,7	47,9	56	65,2
DIREC_30_to_0_kw_2	95,1	97,6	105,7	114,9
DIREC_30_to_0_kw_3	34,1	35,5	50,1	54,3

Tabla 19: Tiempos de detención de descarga conexión directa (ms)

4.5.2 Conexión a través de red Modbus

MODBUS	80%	85%	90%	95%
MODB_15_to_0_kw	122,3	123,5	140,7	143,2
MODB_20_to_0_kw	55,5	56,7	72,9	75,8
MODB_25_to_0_kw	48,9	50,2	65,6	69,1
MODB_30_to_0_kw	82,6	92,5	99,5	103,1
MODB_30_to_0_kw_2	84	89,7	94,1	109,4

Tabla 20: Tiempos de detención de descarga conexión Modbus (ms)

4.5.3 Comparativa

COMPARATIVA	80%	85%	90%	95%
DIREC_30_to_0_kw	45,7	47,9	56	65,2
DIREC_30_to_0_kw_2	95,1	97,6	105,7	114,9
DIREC_30_to_0_kw_3	34,1	35,5	50,1	54,3
MODB_30_to_0_kw	82,6	92,5	99,5	103,1
MODB_30_to_0_kw_2	84	89,7	94,1	109,4
Media	68,30	72,64	81,08	89,38
Desviación	26,69	28,72	26,00	27,64

Tabla 21: Comparativa tiempos de detención de descarga (ms)

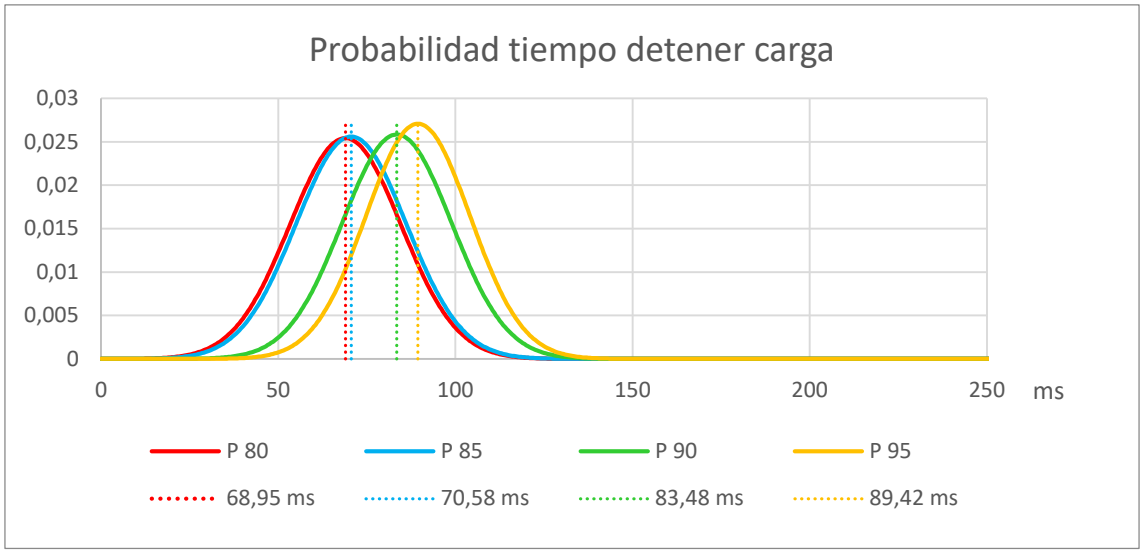

Figura 23: Probabilidad de tiempo de detención de descarga

La figura 23 muestra que de nuevo el tiempo de detención es inferior al de arranque. También se aprecia que los tiempos de respuesta están agrupados.

4.6 Tiempo de respuesta procesos de carga a descarga

Este grupo de medidas se han llevado a cabo comandando el sistema para alcanzar desde carga, una potencia determinada de descarga.

4.6.1 Conexión directa

DIRECTA	80%	85%	90%	95%
DIREC_-10_to_10_kw	95,6	101,5	160,5	198,8
FDIREC_-20_to_20_kw	90	108,9	147,6	190,9
DIREC_-25_to_25_kw	84,7	96,8	132,5	193,3
FDIREC_-25_to_25_kw	174,6	188,3	196,1	>200
DIREC_-30_to_30_kw	169,1	185,5	198,2	>200
DIREC_-30_to_30_kw_2	144,7	158,8	194,4	>200
DIREC_-30_to_30_kw_3	170,9	189,1	199,6	>200

Tabla 22: Tiempos de carga a descarga conexión directa (ms)

En este caso, el tiempo de respuesta a potencia máxima al 95 % es mayor a la escala temporal del osciloscopio (200 ms).

4.6.2 Conexión a través de red Modbus

MODBUS	80%	85%	90%	95%
MODB_-10_to_10_kw	144,7	160,7	165,4	>200
MODB_-20_to_20_kw	189,3	198,7	>200	>200
MODB_-25_to_25_kw	82,8	92,3	108,6	131,3
FMODB_-25_to_25_kw	76,1	85,4	97,5	117,7
MODB_-30_to_30_kw	139,5	149,8	164,9	189,3
MODB_-30_to_30_kw_2	122,2	131,1	148,5	171,1
MODB_-30_to_30_kw_3	104,4	112	128,9	151,1

Tabla 23: Tiempos de carga a descarga conexión Modbus (ms)

4.6.3 Comparativa

COMPARATIVA	80%	85%	90%	95%
DIREC_-30_to_30_kw	169,1	185,5	198,2	>200
DIREC_-30_to_30_kw_2	144,7	158,8	194,4	>200
DIREC_-30_to_30_kw_3	170,9	189,1	199,6	>200
MODB_-30_to_30_kw	139,5	149,8	164,9	189,3
MODB_-35_to_35_kw	122,2	131,1	148,5	171,1
MODB_-40_to_40_kw	104,4	112	128,9	151,1
MEDIA	141,80	154,38	172,42	170,50
DESVIACION	24,97	29,11	30,05	19,11

Tabla 24: Comparativa tiempos de carga a descarga (ms)

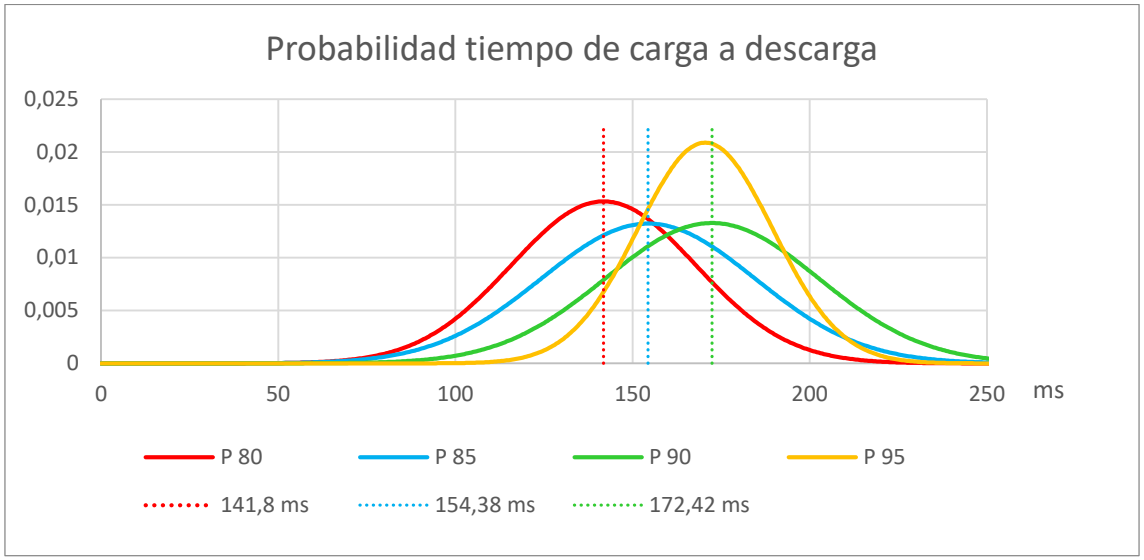

Figura 24: Probabilidad de tiempo de carga a descarga

En la figura 24 se observa que la interacción entre procesos de carga y descarga, aumenta significativamente el tiempo de respuesta.

4.7 Tiempo de respuesta de procesos de descarga a carga

Este grupo de medidas se han obtenido comandando el sistema para alcanzar desde descarga, una potencia determinada de carga.

4.7.1 Conexión directa

DIRECTA	80%	85%	90%	95%
DIREC_5_to_-5_kw	153,6	173,9	191,4	194,6
DIREC_10_to_-10_kw	177,7	178,2	178,8	179,5
DIREC_15_to_-15_kw	133,5	134,8	140	141,6
DIREC_20_to_-20_kw	66,1	67,8	69,6	75,2
DIREC_30_to_-30_kw	184,9	186	187,7	192,6
DIREC_30_to_-30_kw_2	175,3	179,7	182	183,7

Tabla 25: Tiempos de descarga a carga conexión directa (ms)

4.7.2 Conexión a través de red Modbus

MODBUS	80%	85%	90%	95%
MODB_10_to_-10_kw	143,6	144	144,4	145
BMODB_10_to_-10_kw	158,2	158,7	159,1	159,7
CMODB_15_to_-15_kw	90,9	92	93,2	97,9
MODB_15_to_-15_kw	119,1	120,4	125,1	127,1
MODB_30_to_-30_kw	179,9	181	182,5	187,6
MODB_30_to_-30_kw_2	109,4	114,2	115,4	116,7
MODB_30_to_-30_kw_3	74,3	80,4	82,7	83,9

Tabla 26: Tiempos de descarga a carga conexión Modbus (ms)

4.7.3 Comparativa

COMPARATIVA	80%	85%	90%	95%
DIREC_30_to_-30_kw	184,9	186	187,7	192,6
DIREC_30_to_-30_kw_2	175,3	179,7	182	183,7
MODB_30_to_-30_kw	179,9	181	182,5	187,6
MODB_30_to_-30_kw_2	109,4	114,2	115,4	116,7
MODB_30_to_-30_kw_3	74,3	80,4	82,7	83,9
Media	144,76	148,26	150,06	152,90
Desviación	49,98	48,09	48,03	49,50

Tabla 27: Comparativa de tiempos de descarga a carga (ms)

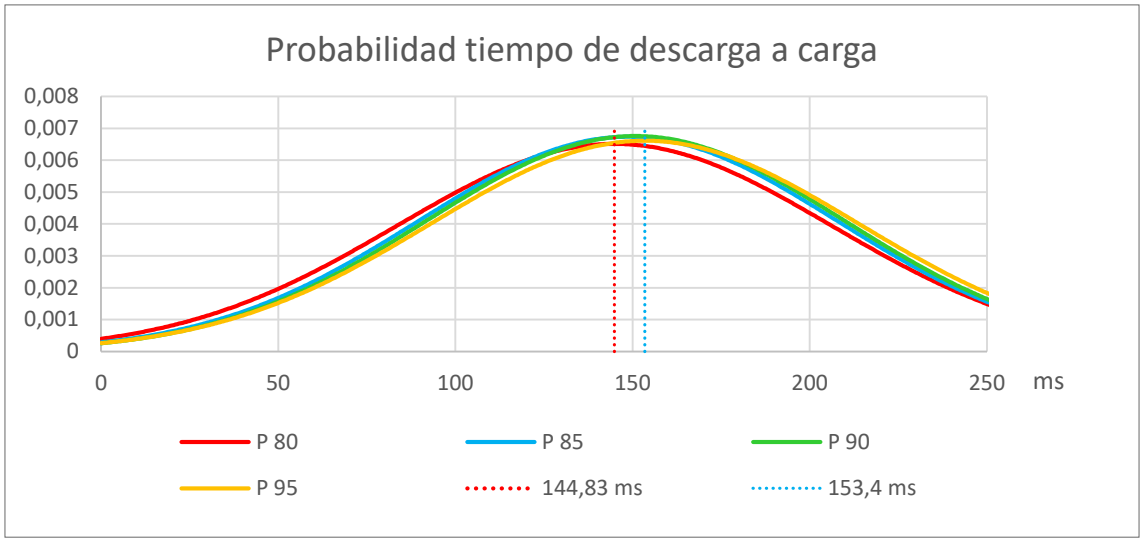

Figura 25: Probabilidad de tiempo de descarga a carga

En la figura 25 se aprecia que las probabilidades de este grupo de medidas se encuentran agrupadas y poseen una gran desviación.

4.8 Comparativa de procesos

En el siguiente gráfico se muestra la comparativa entre los tiempos de respuesta de todos los procesos explicados previamente.

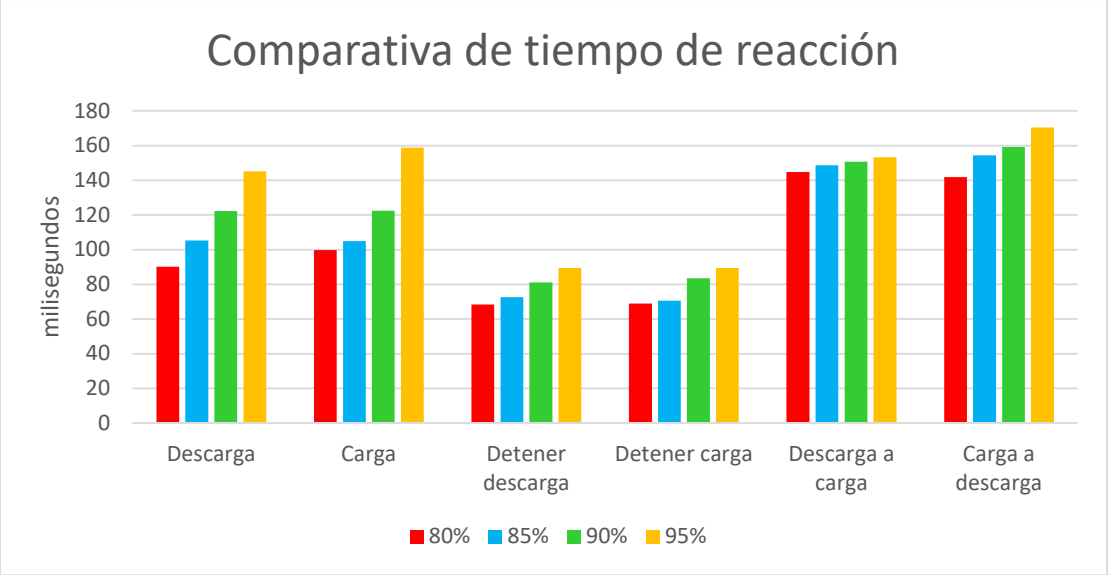

Figura 26: Comparativa de tiempos de reacción entre procesos

Se puede concluir que los procesos más rápidos son los de detención de carga y descarga, con un tiempo medio inferior a 90 milisegundos. Mientras que los procesos de descarga a carga y carga a descarga son los más lentos con tiempos medios superiores a 150 milisegundos.

Por último, los procesos de iniciar carga e iniciar descarga marcan tiempos de respuesta inferiores a los 160 milisegundos.

4.9 Fuentes de error en el proceso

Las fuentes de error durante el proceso de medida son:

4.9.1 Tiempo de ejecución

Es el tiempo de ejecución de los comandos de enviar una determinada potencia al sistema y disparar el osciloscopio.

Este tiempo ha sido despreciado al realizar el análisis, ya que su magnitud es de 0,7 milisegundos. El análisis del tiempo de ejecución se encuentra disponible en el Anexo 3.

4.9.2 Tiempo de respuesta del osciloscopio

Se ha definido como el tiempo que invierte el ordenador en ejecutar la orden de disparar el osciloscopio. Este tiempo es de unos 0,6 milisegundos. De nuevo es matemáticamente despreciable en comparación con el tiempo de respuesta del sistema. Su análisis se encuentra en el Anexo 4.

4.9.3 Periodo de la media móvil

Al realizar el análisis de las medidas se ha realizado una media de orden 100 sobre la potencia aparente para obtener la potencia activa, esto supone una media móvil de 10 milisegundos de periodo, pudiendo introducir un retardo máximo de 5 milisegundos en el análisis.

4.9.4 Número de muestras

Debido a errores durante el proceso de medida, un gran número de muestras han tenido que ser desechadas. Esto da lugar a que el tamaño de muestra sea bajo, por lo que, para conseguir un resultado más sólido, se deberían realizar más medidas.

5 Conclusiones

En primer lugar, el análisis realizado en el presente Trabajo Fin de Grado, demuestra que la comunicación con el sistema a través de la red Modbus o la conexión directa no presentan diferencias apreciables en el tiempo de respuesta del mismo.

En segundo lugar, el objetivo principal del trabajo era la comprobación de las prestaciones del sistema de baterías y conversor para el futuro modelado de la máquina síncrona virtual (VISMA). Para implementar el VISMA, se necesita un sistema de almacenamiento con un tiempo de respuesta inferior a 10 milisegundos.

Como se ha demostrado, el proceso de carga tiene un tiempo de reacción comprendido en un rango de 99 a 160 milisegundos. El proceso de descarga posee un tiempo de reacción comprendido entre 90 y 160 milisegundos. Por último, el proceso de carga a descarga se encuentra en el rango de 141 a 170 milisegundos; y el proceso de descarga a carga en el rango de 144 a 153 milisegundos.

Por tanto, el sistema compuesto por la batería VUCAB52 y el conversor BAT50, no provee un tiempo de reacción lo bastante bajo para cumplir con las especificaciones.

Se puede concluir que sería necesario un cambio de *hardware* del sistema para poder cumplir los requisitos de modelado del sistema de estabilización de red.

A pesar de todo lo anterior, el sistema cumple holgadamente con los requerimientos de almacén energético, y gracias a la comunicación Modbus, puede ser comandado desde un servidor con los valores de cesión y absorción de potencia que mejor convengan a la situación actual de generación y consumo.

A nivel personal, la realización de este Trabajo Fin de Grado, ha supuesto un acercamiento a los sistemas de comunicación industrial modernos, y a los sistemas de almacenamiento energético y su utilización para solventar los problemas derivados de la integración de energías renovables.

Por otro lado, el trabajo me ha permitido poner a prueba las habilidades adquiridas durante mis estudios, debido al alcance del mismo y a su dificultad técnica; así como los inconvenientes idiomáticos. De cualquier modo, la realización del presente trabajo ha sido realmente gratificante debido a su modernidad, alcance y temática.

6 Bibliografía

[1] Endesa, «Endesa Educa,» [En línea]. Available: https://www.endesaeduca.com/Endesa_educa/recursos-interactivos/smart-city/smart-grid. [Último acceso: 2017].

[2] «Grupo Novelec,» 2017. [En línea]. Available: http://blog.gruponovelec.com/electricidad/como-funciona-smart-grid/.

[3] Xi Fang, Satyajayant Misra, Guoliang Xue, Dejun Yang, «Smart Grid - The New and Improved Power Grid: A Survey,» *IEEE,* vol. 14, nº 4, pp. 944-964, 2012.

[4] J. C. G. Galarza, *Diseño e implementación del sistema de control de un inversor multinivel de fuentes independientes, por medio de modulación vectorial espacial,* Cuenca, 2012.

[5] López Mesa Diana Jimena, Camacho Muñoz Guillermo Alberto, Díaz Chávez, *MODULACIÓN PWM APLICADA A INVERSORES TRIFÁSICOS DENTRO DEL ESQUEMA DE ACCIONAMIENTOS ELÉCTRICOS AC.,* 2007.

[6] L. Hassaine, *Implementación de un control digital de potencia activa y reactiva para inversores,* Madrid, 2010.

[7] «Battery Balancing,» Wikipedia, [En línea]. Available: https://en.wikipedia.org/wiki/Battery balancing.

[8] «Battery management system,» Wikipedia, [En línea]. Available: https://en.wikipedia.org/wiki/Battery_management_system.

[9] A. F. R. Olaya, «Implementación de una red Modbus/TCP,» Santiago de Cali, 2002.

[10] P. Ledesma, «Regulación de frecuencia y potencia,» Madrid, 2008.

[11] Pablo F. Frack, Pedro E. Mercado, Marcelo G. Molina, «Extending the VISMA Concept to Improve the Frequency Stability in Microgrids».

[12] Miguel Torres, Luis AC Lopes, «An Optimal Virtual Inertia Controller to Support Frequency Regulation in Autonomous Diesel Power Systems with High Penetration of Renewables,» Montreal, 2011.

[13] L. Beuschausen, R. Benger, J. Gollenstede, B. Werther, H. P. Beck, «Dynamic Requierements on LFP Batteries used for Providing Virtual Inertia,» de *NEIS Conference*, Hamburg, 2016.

[14] Yong Chen, Ralf Hesse, Dirk Turschner and Hans-Peter Beck, «Improving the Grid Power Quality Using Virtual Synchronous Machines,» de *Proceedings of the 2011 International Conference on Power Engineering, Energy and Electrical Drives*, Málaga, 2011.

[15] Hans-Peter Beck, Ralf Hesse;, «Virtual Synchronous Machine,» *IEEE,* pp. 1-6, 2007.

[16] Yong Chen, Ralf Hesse, Dirk Turschner and Hans-Peter Beck, «Comparison of methods for implementing virtual synchronous machine on inverters,» de *International Conference on Renewable Energies and Power Quality*, 2012.

[17] «Masadelante,» [En línea]. Available: http://www.masadelante.com/faqs/socket. [Último acceso: Enero 2018].

[18] J. C. R. Pacheco, «Microsoft,» [En línea]. Available: https://msdn.microsoft.com/es-es/communitydocs/win-dev/os/manejador-de-procesos-los-threads. [Último acceso: Enero 2018].

[19] C. Núñez–Gutiérrez, «http://www.scielo.org.mx,» Enero 2009. [En línea]. Available: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-77432009000100006.

[20] Weiyi Zhang, Daniel Remon, Joan Rocabert, Alvaro Luna1, J. Ignacio Candela, Pedro Rodriguez, «Frequency Support Properties of the Synchronous Power Control for Grid-Connected Converters,» *IEEE,* 2016.

[21] FeCon GmbH, «Interface Description Modbus TCP and EtherCat,» 2016.

[22] WSTECH, «BAT 50,» [En línea]. Available: http://www.wstech.com/en/products/indoor-central-grid-inverters-battery-chargers/bat-50. [Último acceso: 2017].

[23] P. Innovation, «VUCAB-52 Power Innovation,» [En línea]. Available: https://www.powerinnovation.com/datasheets/Speicher_Serie_Eng2017.pdf. [Último acceso: 2017].

ANEXO 1: Código de la aplicación en *Python*

```
#########################################################################
#                                                                       #
#   ESCUELA UNIVERSITARIA POLITECNICA DE TERUEL                         #
#   GRADO EN INGENIERIA ELECTRONICA Y AUTOMATICA                        #
#                                                                       #
#   TRABAJO DE FIN DE GRADO                                             #
#   Curso 2017/18                                                       #
#                                                                       #
#   Autor: Miguel Torres Hornero                                        #
#   Fecha: Febrero de 2018                                              #
#                                                                       #
#   Fichero: tiempoRespuestaBateria.py                                  #
#                                                                       #
#   Breve descripcion: Medida de tiempo de respuesta de sistema de      #
#   baterias VUCAB52 y conversor BAT50. Comandado del sistema.          #
#   Disparo de osciloscopio, guardado de datos y medida de tiempos      #
#   de ejecucion.                                                       #
#                                                                       #
#   Acentos intencionadamente eliminados.                               #
#                                                                       #
#########################################################################


# Import
import socket                       # Sockets
import threading                    # Hilos
import os                           # Archivos y directorios
import usbtmc                       # Puertos USB
import time                         # Manejo de tiempo
from errores import errorText       # Errores del sistema


#Definicion de colores para terminal-> solo para LINUX
BLUE = '\033[;94m'
GREEN = '\033[;92m'
YELLOW = '\033[;93m'
WHITE = '\033[;37m'
RED = '\033[;31m'


#Menus
MENUSELECCION = '0-> APAGADO \t1->Conectar a bateria \t2->Encender
sistema \t3-> Test \t4->0 W to P \t5 ->P to 0 W \t6-> P to P2 \t7->
Consultar errores '
MENUAPAGADO = YELLOW + "\n0-> CANCELAR \t1-> OFF Fecon \t2-> OFF LICON
\t3-> OFF 24 Vdc  : "
MENUENCENDIDO = GREEN + "\n 0-> CANCELAR \t1-> 24 V ON \t2-> Run Licon
\t3-> Run FeCon \t4-> Standby Fecon "
FINPROGRAMA = RED +"\nPulse 'f' para terminar el programa: "

#Introducir potencias
INTRODUZCAPOTENCIAINCIAL = "Introduzca potencia inicial deseada dentro
del rango -30 a 30: "
INTRODUZCAPOTENCIA  = "Introduzca potencia final deseada dentro del
rango -30 a 30: "
```

```
# Menu de selecciones
SELECCIONES = {
    0: "Apagado",
    1: "Seleccionado conectar a bateria",
    2: "Seleccionado Encender sistema",
    3: "Seleccionado Test de sistema",
    4: "Seleccionado desde 0 W ir a Potencia",
    5: "Seleccionado desde Potencia ir a 0 W",
    6: "Seleccionado desde Potencia 1 ir a Potencia 2",
    7: "Seleccionado show errors",
    8: "Seleccionado apagar FECON",
    9: "Selecionado apagar LICON",
    10: "Seleccionado apagar 24 VDC",
    11: "Seleccionado encender los reles",
    12: "Selecionado Run LICON",
    13: "Seleccionado Run FECON",
    14: "Seleccionado Standby FECON",
 }


#-----------------------------------------------------------------------

'''
    Clase oscilloscope, Agilent Technologies DSO-X 2004A incorpora
    las operaciones de disparo y guardado de fichero de medida

    Variables de entrada: Identificadores de dispositivo USB TMC
'''
class oscilloscope():

    def __init__(self,id1, id2):

        print("Direccion de osciloscopio : " + str(id1) +" , " +
               str(id2) + "\n")
        try:
            instr = usbtmc.Instrument(id1, id2)

        except usbtmc.usbtmc.UsbtmcException:
            print(RED + "\n Osciloscopio no detectado \n" + WHITE)

    # Establecer conexion con osciloscopio
    def conectar(self):

        print(instr.ask("*IDN?"))
        print("Conexion establecida")
        print(instr.ask(":SYST:ERR?\n"))
        print(instr.ask(":SYST:ERR?\n"))
        print(instr.ask(":SYST:ERR?\n"))
        print(instr.ask(":SYST:ERR?\n"))
        print(instr.ask(":SYST:ERR?\n"))

    # Consultar de errores
    def consultarErrores(self):

        print(instr.ask(":SYST:ERR?\n"))

    # Consultar estatus osciloscopio
    def consultaStatus(self):

        instr.read_stb()
```

```
    # Disparar osciloscopio
    def trigger(self):

        instr.write("*TRG")

    # Run de osciloscopio
    def run_OSCL(self):

        instr.write(":RUN")

    # Guardar fichero de datos csv
    def save_csv( self,name):

        instr.write('":SAVE:PWD "\usb"')
        instr.write(':SAVE:FILename "\usb"')
        instr.write(':SAVE:WAVeform:FORMat CSV')
        nombreCSV = ':SAVE:WAVeform:STARt "' + name + '" \n'
        instr.write(nombreCSV)

    # Guardar screenshot de la medida realizada
    def save_image(self, name):

        instr.write('":SAVE:PWD "\usb"')
        instr.write(':SAVE:IMAGe:FACTors 1')
        instr.write(':SAVE:IMAGe:FORMat PNG')
        nombreIMG = ':SAVE:IMAGe:STARt "' + name + '" \n'
        instr.write(nombreIMG)
```

```
#-----------------------------------------------------------------------
'''
    Clase battery: comunicacion con sistema de baterias VU-CAB52 e
    inversor trifasico BAT50.

    Variables de entrada: Direccion IP de bateria
                          Lock para sincronizacion de medida
'''
class battery():

    def __init__(self, direccionIP, lock) :
        print("Direccion ip de bateria : " + str(direccionIP))

        server_addr = direccionIP
        socket_timeouts = 0
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        data = ''
        self.estadoConexion = 12345
        socket_timeouts = 0
        self.server_address = (server_addr, 502)
        # Crear hilo de conexion
        self.comprobarConexion_= threading.Thread(
                    target=self.comprobarConexionHilo, args = (lock,))
        self.continuarHilo = True
```

```
# Conectar con sistema de baterias e inversor
def conectar(self):
    print("Estableciendo conexion")
    data = ''
    self.estadoConexion= self.sock.connect_ex(self.server_address)
    self.sock.settimeout(3)  #Timeout set para 3 segundos

    if(self.estadoConexion == 0):
        print("Connection OK")

    else: # Reapertura de socket
        print(RED + "No connection to LiCon -> must reconnect")
        self.sock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)

    # Arrancado de hilo de conexion
    if(self.comprobarConexion_.is_alive()):
        print("Thread activo")

    else:
        print(BLUE + "Arrancando hilo" + WHITE)
        self.comprobarConexion_.start()

    return(self.estadoConexion)

# Hilo de comprobacion de conexion con sistema
def comprobarConexionHilo(self, lock):

    # Definicion de frame a enviar
    frameVoltajeL3 = [0x00,0x02,0x00,0x00,0x00,0x06,0x01,0x04,
                      0x00,0x02,0x00,0x01]
    s_string = ''.join(str(chr(e)) for e in frameVoltajeL3)
    noConexion = 0

    while(self.continuarHilo):
        lock.acquire() # Bloqueo para no interferir con la medida
        try:
            self.sock.sendall(s_string)
            data= self.sock.recv(33)
            l = len(data)
            if l == 0: # Si recibe una trama vacia: error
                print(RED + "Conexion perdida -> reconectar")

            else:
                noConexion = 1
        except socket.error:
            if (noConexion == 0): # Notificar una sola vez
                print("\n" + RED + "ERROR DE CONEXION CON BATERIA"
                        + WHITE)
                noConexion = 1
        lock.release()
        time.sleep(1)

    print(RED + "Hilo terminado")

# Terminar hilo de conexion
def terminarHilo(self, lock):
    if (self.comprobarConexion_.isAlive()):
        print("Hilo activo -> matando hilo \n")
        lock.acquire()
        self.continuarHilo = False # Termina hilo
```

```
        lock.release()
# Enviar frame modbus
def enviarFrame(self, frame):

    if(self.estadoConexion == 0):
        try:
            self.sock.sendall(frame)
        except socket.error:
            print(RED + "\nError en el envio\n" + WHITE)
    else:
        print(RED + "No conexion")
        print(WHITE + "\n")


# Recibir de tramas Modbus
def recibirFrame(self, frame):
    if(self.estadoConexion == 0):
        dato = self.sock.recv(33)
    else:
        dato = 0
    return dato

# Consultar errores de sistema de almacenamiento
def consultaErrores(self):
    # Vector de errores 1
    request_frame_Error_0 = [0x0b, 0xb8, 0x00, 0x00, 0x00, 0x06,
                             0x01, 0x04, 0x0b, 0xb8, 0x00, 0x01]
    string = ''.join(str(chr(e)) for e in request_frame_Error_0)
    self.enviarFrame(string)
    data= self.recibirFrame()

    if(len(data) > 0):

        error  =convertirValorC(data)

        if error == 0:
            print("No Errors")

        else:
            print(RED +"Error:" + errorText[error] + ":" + error)

        # Vector de errores 2
        request_frame_Error_2=[0x0b, 0xb9, 0x00, 0x00, 0x00, 0x06,
                               0x01, 0x04, 0x0b, 0xb9, 0x00, 0x01]
        string=''.join(str(chr(e)) for e in request_frame_Error_2)
        self.enviarFrame(string)
        data= self.recibirFrame()
        error  =  convertirValorC(data)
        if (error != 0):
            print(RED +"Vector 2: "+errorText[error] +":" + error)

        # Vector de errores 3
        request_frame_Error_3=[0x0b, 0xba, 0x00, 0x00, 0x00, 0x06,
                               0x01, 0x04, 0x0b, 0xba, 0x00, 0x01]
        string= ''.join(str(chr(e))for e in request_frame_Error_3)
        self.enviarFrame(string)
        data = self.recibirFrame()
        error  =  convertirValorC(data)
        if (error != 0):
            print(RED +"Vector 3: "+errorText[error] +":" + error)
```

```
        # Vector de errores 4
        request_frame_Error_4=[0x0b, 0xbb, 0x00, 0x00, 0x00, 0x06,
                               0x01, 0x04, 0x0b, 0xbb, 0x00, 0x01]
        string=''.join(str(chr(e)) for e in request_frame_Error_4)
        self.enviarFrame(string)
        data= self.recibirFrame()
        error  =  convertirValorC(data)
        if (error != 0):
            print(RED +"Vector 4: "+errorText[error]+ ":" + error)

        # Vector de errores 5
        request_frame_Error_5=[0x0b, 0xbc, 0x00, 0x00, 0x00, 0x06,
                               0x01, 0x04, 0x0b, 0xbc, 0x00, 0x01]
        string=''.join(str(chr(e)) for e in request_frame_Error_5)
        self.enviarFrame(string)
        data= self.recibirFrame()
        error  =  convertirValorC(data)
        if (error != 0):
            print(RED +"Vector 5: "+errorText[error]+ ":" + error)

        # Vector de errores 6
        request_frame_Error_6=[0x0b, 0xbd, 0x00, 0x00, 0x00, 0x06,
                               0x01, 0x04, 0x0b, 0xbd, 0x00, 0x01]
        string=''.join(str(chr(e)) for e in request_frame_Error_6)
        self.enviarFrame(string)
        data= self.recibirFrame()
        error  =  convertirValorC(data)
        if (error != 0):
            print(RED +"Vector 6: "+errorText[error]+ ":" + error)

        # Vector de errores 7
        request_frame_Error_7=[0x0b, 0xbe, 0x00, 0x00, 0x00, 0x06,
                               0x01, 0x04, 0x0b, 0xbe, 0x00, 0x01]
        string=''.join(str(chr(e)) for e in request_frame_Error_7)
        self.enviarFrame(string)
        data= self.recibirFrame()
        error  =  convertirValorC(data)
        if (error != 0):
            print(RED +"Vector 7: "+errorText[error]+ ":" + error)

        # Vector de errores 8
        request_frame_Error_8=[0x0b, 0xbf, 0x00, 0x00, 0x00, 0x06,
                               0x01, 0x04, 0x0b, 0xbf, 0x00, 0x01]
        string=''.join(str(chr(e)) for e in request_frame_Error_8)
        self.enviarFrame(string)
        data= self.recibirFrame()
        error  = convertirValorC(data)
        if (error != 0):
            print(RED +"Vector 8: "+errorText[error] +":" + error)

        tiempo_final = time.time()


    else:
        print(RED +"No connection")
```

```
# Encender los reles que conectan las celdas de baterias
def turn_24V_ON(self):

    frame_system_status= [0x05,0xa2,0x00,0x00,0x00,0x06,0x02,0x04,
                          0x05,0xa2,0x00,0x01]
    s_string_system_status = ''.join(str(chr(e)) for e in
                               frame_system_status)
    request_frame_24V_ON = [0x05,0xa3,0x00,0x00,0x00,0x0b,0x02,
                            0x10,0x05,0xa3,0x00, 0x02,0x04, 0x00]
    s_string_24V_ON= ''.join(str(chr(e)) for e in
                         request_frame_24V_ON)
    self.enviarFrame(s_string_24V_ON)
    time.sleep(4)

    self.enviarFrame(s_string_system_status)
    data= self.recibirFrame()
    print("LiCon no está listo, espere mientras se cierran reles")

    # Esperar hasta que todos los reles se conecten
    while(ord(data[10]) !=7):

        self.enviarFrame(s_string_system_status)
        data= self.recibirFrame()
        time.sleep(0.5)

    print("\nLiCon listo") # Sistema listo para arrancar

# Borrar registros de errores de bateria
def reset_errors_LiCon(self):

    #CLEAR FSP (erros in LICON) reg 1600
    request_frame_CLEAR_FSP = [0x06,0x40,0x00,0x00,0x00,0x0b,0x02,
                0x10,0x06,0x40,0x00,0x02,0x04,0x00,0x00,0x00,0x01]
    s_string_CLEAR_FSP = ''.join(str(chr(e)) for e in
                         request_frame_CLEAR_FSP)
    self.enviarFrame(s_string_CLEAR_FSP)

    print("Reset errors completado")

# Poner en marcha de sistema de baterias : conversor DC-DC
def run_LiCon(self):
    #Run on licon
    request_frame_RUN_ON = [0x05,0xa0,0x00,0x00,0x00,0x0b,0x02,
                0x10,0x05,0xa0,0x00,0x02,0x04,0x00,0x00,0x00,0x01]
    s_string_RUN_ON= ''.join(str(chr(e)) for e in
                         request_frame_RUN_ON)
    self.enviarFrame(s_string_RUN_ON)

# Apagar sistema baterias -> conversor DC-DC
def turnOffLicon(self):

    request_frame_RUN_OFF_LICON = [0x05,0xa1,0x00,0x00,0x00,0x0b,
         0x02,0x10,0x05,0xa1,0x00,0x02,0x04,0x00,0x00,0x00,0x01]
    s_string_RUN_OFF_LICON= ''.join(str(chr(e)) for e in
                               request_frame_RUN_OFF_LICON)

    self.enviarFrame(s_string_RUN_OFF_LICON)
    print("Licon apagado")
```

```
# Desconectar los reles de las celdas de baterias
def turnOff24Vdc(self):

    request_frame_24V_OFF = [0X05,0Xa4,0x00,0x00,0x00,0x0b,0x02,
                             0x10,0x05,0xa4,0x00,0x02,0x04,0x00]
    s_string_24V_OFF= ''.join(str(chr(e)) for e in
                              request_frame_24V_OFF)

    self.enviarFrame(s_string_24V_OFF)
    print("24 VDC apagado")

# Borrar registro de errores en conversor AC-DC
def reset_errors_FeCon(self):
    #CLEAR ERROS FECON reg 2200
    request_frame_CLEAR_ERROR_FECON =[0x08,0x98,0x00,0x00,0x00,
                0x06,0x01,0x05,0x08,0x98,0xff,0x00]
    s_string_CLEAR_ERROR_FECON = ''.join(str(chr(e)) for e in
                         request_frame_CLEAR_ERROR_FECON)
    self.enviarFrame(s_string_CLEAR_ERROR_FECON)

# Poner en Standby conversor AC-DC
def standbyFeCon(self):
    #RUN ON-STANDBY-OFF  FECON   reg 3102
    request_frame_STANDBYFECON =[0x0c,0x1e,0x00,0x00,0x00,0x06,
                    0x01,0x06,0x0c,0x1e,0x00,0x14] # Data 20
    s_string_STANDBYFECON = ''.join(str(chr(e)) for e in
                         request_frame_STANDBYFECON)
    self.enviarFrame(s_string_STANDBYFECON)

# Poner en marcha del conversor AC-DC
def run_FeCon(self):
    #RUN ON-STANDBY-OFF  FECON   reg 3102
    request_frame_RUN_ON_FECON =[0x0c,0x1e,0x00,0x00,0x00,0x06,
                         0x01,0x06,0x0c,0x1e,0x02,0xbc] # Data 700
    s_string_RUN_ON_FECON = ''.join(str(chr(e)) for e in
                         request_frame_RUN_ON_FECON)
    self.enviarFrame(s_string_RUN_ON_FECON)

# Apagar conversor AC-DC
def turnOffFecon(self):

    #Transferir 0 kw a bateria
    request_frame_0kw= self.ensamblarFramePkW(0)
    self.enviarFrame(request_frame_0kw)
    time.sleep(0.5)
    request_frame_OFF_FECON = [0x0c,0x1e,0x00,0x00,0x00,0x06,0x01,
                    0x06,0x0c,0x1e,0x00,0x00]  # Data 0
    s_string_OFF_FECON = ''.join(str(chr(e)) for e in
                         request_frame_OFF_FECON)

    self.enviarFrame(s_string_OFF_FECON)
    print("Fecon Apagado")

# Ensamblar frame para potencia ACTIVA
def ensamblarFramePkW(self, value):
    print(value)
    if(-51 < value and value < 51):
        value  = value * 100
        byteAlto =(value >> 8 ) & 0xff
        byteBajo = (value & 0xff)
```

```
        # Registro 5801
        request_frame_kw = [0x16,0xa9,0x00,0x00,0x00,0x06,0x01,
                            0x06,0x16,0xa9,byteAlto, byteBajo]
        print("request frame", value, request_frame_kw)
        frameFinal =''.join(str(chr(e)) for e in request_frame_kw)
        return(frameFinal)

    else:
        print("Number out of range")
        return 0

# Ensamblar frame para potencia REACTIVA
def ensamblarFrameQkw(self, value):

    if(-51 < value and value < 51):
        value  = value * 100
        byteAlto =(value >> 8 ) & 0xff
        byteBajo = (value & 0xff)
        # Registro 5802
        request_frame_kw = [0x16,0xaa,0x00,0x00,0x00,0x06,0x01,
                            0x06,0x16,0xaa,byteAlto, byteBajo]
        frameFinal =''.join(str(chr(e)) for e in request_frame_kw)
        return(frameFinal)

    else:
        print("Number out of range")
        return 0

# Convertir valor Modbus en valor decimal
def convertirValorA(self, dato):

    resultado = (ord(dato[9]) << 8) + (ord(dato[10]))
    resultado = float(resultado) / 10
    return resultado

# Convertir dato recibido por Modbus en 4 bytes
def convertirValorB(self, dato):
    valor =( (ord(dato[9]) << 24) + (ord(dato[10]) << 16) +
            (ord(dato[11]) << 8) + (ord(dato[12])))
    valor = float(valor)/1000
    return valor

# Convertir dato recibido por Modbus en 2 bytes
def convertirValorC(self, dato):
    valor  =int(ord(dato[9])*256 + ord(dato[10]))
    return valor

# Consultar voltaje de la linea 1 (AC)
def voltajeL1(self):

    frameVoltajeL1 = [0x00,0x00,0x00,0x00,0x00,0x06,0x01,0x04,
                      0x00,0x00,0x00,0x01]
    s_string = ''.join(str(chr(e)) for e in frameVoltajeL1 )
    self.enviarFrame(s_string)
    data= self.recibirFrame()
    l = len(data)

    if l != 0:
        valor = self.convertirValorA(data)
        print "Voltaje VL1: " + str(valor) + " V"
```

```
        else:
            print("No dato, fallo en conexion")

    # Consultar voltaje linea 2 (AC)
    def voltajeL2(self):
        frameVoltajeL2 = [0x00,0x01,0x00,0x00,0x00,0x06,0x01,0x04,
                          0x00,0x01,0x00,0x01]
        s_string = ''.join(str(chr(e)) for e in frameVoltajeL2)
        self.enviarFrame(s_string)
        data= self.recibirFrame()
        l = len(data)

        if l != 0:
            valor = self.convertirValorA(data)
            print "Voltaje VL2: " + str(valor) + " V"

        else:
            print("No dato, fallo en conexion")

    # Consultar voltaje linea 3 (AC)
    def voltajeL3(self):
        frameVoltajeL3 = [0x00,0x02,0x00,0x00,0x00,0x06,0x01,0x04,
                          0x00,0x02,0x00,0x01]
        s_string = ''.join(str(chr(e)) for e in frameVoltajeL3)
        self.enviarFrame(s_string)
        data= self.recibirFrame()
        l = len(data)

        if l != 0:
            valor = self.convertirValorA(data)
            print "Voltaje VL3: " + str(valor) + " V"

        else:
            print("No dato, fallo en conexion")

    # Consultar frecuencia de red
    def frecuenciaRed(self):
        frameFrecuencia = [0x00, 0x0a, 0x00, 0x00, 0x00, 0x06, 0x01,
                           0x04, 0x00, 0xa, 0x00, 0x01]  # Grid frequency
        s_string = ''.join(str(chr(e)) for e in frameFrecuencia)
        self.enviarFrame(s_string)
        data= self.recibirFrame()
        l = len(data)
        if(l != 0):
            valor = self.convertirValorA(data)
            valor = valor / 100
            print("Frecuencia de red " + str(valor) + " Hz")
        else:
            print("Error en consulta de frecuencia. Fallo Conexion")

    # Consultar voltaje del bus DC
    def voltajeBusDC(self):

        frameVbusDC = [0x00, 0xc8, 0x00, 0x00, 0x00, 0x06, 0x01, 0x04,
                       0x00, 0xc8, 0x00, 0x01] # Voltaje bus DC
        s_string_VDCbus= ''.join(str(chr(e)) for e in frameVbusDC)
        self.enviarFrame(s_string_VDCbus)
        data= self.recibirFrame()
        l = len(data)
```

```
        if l != 0:
            valor = self.convertirValorA(data)
            print "Voltaje V bus DC: " + str(valor) + " V DC"

        else:
            print("No dato, fallo en conexion")

    # Consultar los voltajes en las celdas de almacenamiento
    def voltajeCeldas(self):

        # Bloque 1 = 55 -> 0x37
        request_frame_V_C1 = [0x00,0x37,0x00,0x00,0x00,0x06,0x02,0x04,
                              0x00,0x37,0x00,0x02]
        s_string_Vc1 =''.join(str(chr(e)) for e in request_frame_V_C1)
        self.enviarFrame(s_string_Vc1)
        data= self.recibirFrame()
        voltaje1 = self.convertirValorB(data)

        # Bloque 2 = 57 -> 0x39
        request_frame_V_C2 = [0x00,0x39,0x00,0x00,0x00,0x06,0x02,0x04,
                              0x00,0x39,0x00,0x02]
        s_string_Vc2 =''.join(str(chr(e)) for e in request_frame_V_C2)
        self.enviarFrame(s_string_Vc2)
        data = self.recibirFrame()
        voltaje2 = self.convertirValorB(data)

        # Bloque 3 = 59 -> 0x3b
        request_frame_V_C3 = [0x00,0x3b,0x00,0x00,0x00,0x06,0x02,0x04,
                              0x00,0x3b,0x00,0x02]
        s_string_Vc3 =''.join(str(chr(e)) for e in request_frame_V_C3)
        self.enviarFrame(s_string_Vc3)
        data = self.recibirFrame()
        voltaje3 = self.convertirValorB(data)

        # Bloque 4 = 61 -> 0x3d
        request_frame_V_C4 = [0x00,0x3d,0x00,0x00,0x00,0x06,0x02,0x04,
                              0x00,0x3d,0x00,0x02]
        s_string_Vc4 =''.join(str(chr(e)) for e in request_frame_V_C4)
        self.enviarFrame(s_string_Vc4)
        data= self.recibirFrame()
        voltaje4 = self.convertirValorB(data)

        # Bloque 5 = 63 -> 0x3f
        request_frame_V_C5 = [0x00,0x3f,0x00,0x00,0x00,0x06,0x02,0x04,
                              0x00,0x3f,0x00,0x02]
        s_string_Vc5 =''.join(str(chr(e)) for e in request_frame_V_C5)
        self.enviarFrame(s_string_Vc5)
        data= self.recibirFrame()
        voltaje5 = self.convertirValorB(data)

        print("Voltaje celdas: C1: " + str(voltaje1) + " C2: "+
            str(voltaje2) + " C3: " + str(voltaje3) + " C4: " +
            str(voltaje4) + " C5: " + str(voltaje5))

    # Cerrar socket
    def cerrarSocket(self):

        print("Cerrando socket")
        self.sock.close()
```

```
#---------------------------------------------------------------------

# Confirmar por teclado la seleccion
def confirmacionSeleccion(seleccionado) :
    print SELECCIONES[seleccionado]
    confir = raw_input("Para confirmar teclee 's' y enter: ")
    if(confir == 's' or confir == 'S'):
        return True
    else:
        return False

# Comprobar la potencia introducida por usuario dentro de rango
def comprobarRango(potencia):
    # De -30 kW a + 30 kW
    if(potencia <= 30) and (potencia >= -30):
        return True
    else:
        return False


# Encender sistema de baterias y conversor
def encenderSistema(bateria):

    print(GREEN + "\n Menu de encendido de sistema")
    seleccionEncendido = 6
    while(seleccionEncendido != 0):

        try:
            seleccionEncendido = int(input(MENUENCENDIDO))
            if(seleccionEncendido == 1): # 24 V on -> reles

                if(confirmacionSeleccion(11)):
                    print("Confirmado")
                    bateria.reset_errors_LiCon()
                    bateria.turn_24V_ON()

            elif(seleccionEncendido == 2): # Run LiCon

                if(confirmacionSeleccion(12)):
                    print("Confirmado")
                    bateria.run_LiCon()

            elif(seleccionEncendido == 3): # Run FeCon

                if(confirmacionSeleccion(13)):
                    print("Confirmado")
                    bateria.reset_errors_FeCon()
                    bateria.run_FeCon()


            elif(seleccionEncendido == 4): # Standby FeCon

                if(confirmacionSeleccion(14)):
                    print("Confirmado")
                    bateria.reset_errors_FeCon()
                    bateria.standbyFeCon()

        except:
            print("Repita Seleccion")
    print("Fin de encendido")
    return
```

```
# Test de sistema de baterias y osciloscopio
def testSistemaCompleto(bateria, osciloscopio):

    osciloscopio.run_OSCL()
    print("Test voltaje linea 1")
    bateria.voltajeL1()
    print("Test voltaje linea 2")
    bateria.voltajeL2()
    print("Test voltaje linea 3")
    bateria.voltajeL3()
    print("Test frecuencia de red")
    bateria.frecuenciaRed()
    print("Test voltaje del bus DC")
    bateria.voltajeBusDC()
    print("Test voltaje celdas: ")
    bateria.voltajeCeldas()
    #Test de osciloscopio
    print("Test osciloscopio -> estado de equipo:")
    osciloscopio.consultarErrores()
    print("Test trigger")
    osciloscopio.trigger()
    return


# Medir tiempo de respuesta
def medidaP1_to_P2(bateria, lock, osciloscopio, potencia1,
potencia2,modoConexion, ficheroTempEjecucion):

    #Ensamblar frames de potencia reposo, inicial y final
    frameCeroPotencia = bateria.ensamblarFramePkW(0)
    framePrimeraPotencia = bateria.ensamblarFramePkW(potencia1)
    frameSegundaPotencia = bateria.ensamblarFramePkW(potencia2)

    #Enviar primera potencia
    bateria.enviarFrame(framePrimeraPotencia)
    print("Enviada 1ª P. Espera de 2 s para estabilizar potencia")
    time.sleep(2) # Espera de 2 segundos para estabilizar

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    lock.acquire() # Bloqueo de thread
    tiempoInicial = time.time() # Medida del tiempo de ejecucion
    bateria.enviarFrame(frameSegundaPotencia)#Enviar segunda potencia
    osciloscopio.trigger() # Disparo de osciloscopio
    tiempoFinal = time.time() # Medida del tiempo final
    lock.release() # Desbloqueo de thread

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    print("Segunda potencia enviada, espera de 0.5 segundos")
    time.sleep(0.5)
    print("Fin de medida ->  envio de 0 kw")
    bateria.enviarFrame(frameCeroPotencia) # Bateria a 0 kW

    # Usuario desea guardar los datos
    confirmacionGuardado = raw_input("Medida realizada, para guardar
datos pulse 's' y enter: ")
    if( confirmacionGuardado == "s" or confirmacionGuardado == "S"):

        # Consultar al usuario del nombre del fichero que desea
```

```
        nombre = raw_input("Introduzca un nombre para fichero :")
        nombreFichero = (str(nombre) + modoConexion + "_" +
                     str(potencia1) + "_to_" +str(potencia2) + "_kw")
        print(nombreFichero)
        #Guardar nombre de fichero en fichero.txt con tiempo ejecucion
        osciloscopio.save_csv(nombreFichero)    # Guardar fichero CSV
        osciloscopio.save_image(nombreFichero)  # Guardar screenshot
        print("Fichero guardado")
        intervalo = tiempoFinal - tiempoInicial
        afichero = ("\nTiempo de ejecucion de " + str(nombreFichero)
                    +"  " + str(intervalo) + " s")
        print(afichero)
        ficheroTempEjecucion.write(afichero)# Guardar tiempo ejecucion
        print(BLUE + "Resultados guardados" + WHITE)

    else:
        print(BLUE + "Resultados NO guardados" + WHITE)
    return


# Apagado de sistema de baterias
def apagadoSistema(bateria):
    seleccionApagado = 6
    print(YELLOW + "\n Menu de apagado de sistema\n")
    while(seleccionApagado != 0):

        try:
            seleccionApagado = int(input(MENUAPAGADO))

            if(seleccionApagado == 1): # Apagar FeCon

                if(confirmacionSeleccion(8)):
                    print("Confirmado")
                    bateria.turnOffFecon()

            elif(seleccionApagado == 2): # Apagar LiCon

                if(confirmacionSeleccion(9)):
                    print("Confirmado")
                    bateria.turnOffLicon()

            elif(seleccionApagado == 3): # Desconectar 24 Vdc reles

                if(confirmacionSeleccion(10)):
                    print("Confirmado")
                    bateria.turnOff24Vdc()

        except:
            print("Repita Seleccion")

    print("Fin de apagado")
    return

# Seleccionar modo de conexion
def selecionModoConexion():

    selecConexion = 0
    while(int(selecConexion) != 1 and int(selecConexion) != 2):
        selecConexion = raw_input("Seleccione modo de conexion: 1->
Directa     2-> Red Modbus: ")
```

```
    if (int(selecConexion) == 1):
        modoDeConexion = "DIRECTA"
    else:
        modoDeConexion = "MODBUS"

    return modoDeConexion


#Crear fichero de tiempo de respuesta y el directorio correspondiente
def crearFicheroTempEjecucion(modoConexion):
    nombreFichero = ("conexion" + " " + modoConexion + " " +
                time.strftime("%H-%M-%S") + '.txt')
    base_path = ('Tiempo de ejecucion ' + modoConexion +
                  time.strftime("%d_%m_%y"))

    if not base_path in os.listdir(os.getcwd()):
        os.mkdir(base_path)
    os.chdir(base_path)
    ficheroTiempoEjecucion = open(nombreFichero,'a')
    aFichero =("Medida de tiempo de ejecucion conexion  "+modoConexion
                + " " + time.asctime())
    ficheroTiempoEjecucion.write(aFichero)
    ficheroTiempoEjecucion.write("\n\n")
    return ficheroTiempoEjecucion

# Configuracion del osciloscopio
def setup(osciloscopio):

    osciloscopio.conectar()
    osciloscopio.consultarErrores()
    osciloscopio.trigger() # Disparo de prueba
    osciloscopio.run_OSCL()

    return

#-----------------------------------------------------------------------

# Funcion principal
def main():

    print(BLUE + time.asctime() + WHITE)

    modoConexion = selecionModoConexion()
    ficheroTempEjecucion = crearFicheroTempEjecucion(modoConexion)

    lock = threading.RLock() # Lock para bloquear el hilo de conexion

    licon = battery('192.168.241.44', lock) # Instanciar bateria
    osciloscopio = oscilloscope(2391, 6042) # Instanciar osciloscopio

    setup(osciloscopio) # Configuracion de osciloscopio

    continuar = True

    while(continuar): #Bucle principal: seleccion de funcion

        print(WHITE+"\nSeleccion de medida: ")
```

```
try:
    seleccion = int(input((MENUSELECCION)))

    if((0 <= seleccion <= 7) and
        (confirmacionSeleccion(seleccion))):


        if seleccion == 0: # Apagar sistema
            print("Apagado confirmado")
            apagadoSistema(licon)

            if(raw_input(FINPROGRAMA) == 'f') :
                continuar = False # Finalizar programa

        if seleccion == 1: # Conectar con bateria

            estadoDeConexion = licon.conectar()
            print("Estado conexion: "+str(estadoDeConexion))

        elif seleccion == 2: # Encender de sistema
            encenderSistema(licon)

        elif seleccion == 3: # Test
            testSistemaCompleto(licon,osciloscopio)

        elif seleccion == 4: # Medida desde 0 W a una potencia
            potencia0ToP  = int(input(INTRODUZCAPOTENCIA))
             # Comprobacion del rango introducido
            if comprobarRango(potencia0ToP):
                medidaP1_to_P2(licon,lock, osciloscopio,0,
                    potencia0ToP,modoConexion,
                    ficheroTempEjecucion)

            else:
                print(RED+"Number out of range. Repeat")

        elif seleccion == 5: #Medida desde una potencia a 0 kW
            potencia1  = int(input(INTRODUZCAPOTENCIA))

            if comprobarRango(potencia1):
                medidaP1_to_P2(licon,lock,osciloscopio,
                    potencia1, 0,modoConexion,
                    ficheroTempEjecucion)

            else:
                print(RED+"Number out of range. Repeat")
        # Medida desde una potencia ir a otra
        elif seleccion == 6:
            potencia1  = int(input(INTRODUZCAPOTENCIAINCIAL))
            potencia2  = int(input(INTRODUZCAPOTENCIA))

            if (comprobarRango(potencia1) and
                comprobarRango(potencia2)):

                medidaP1_to_P2(licon,lock,osciloscopio,
                    potencia1, potencia2,modoConexion,
                    ficheroTempEjecucion)
            else:
                print(RED+"Number out of range. Repeat")
```

```
                elif seleccion == 7: # Consultar errores

                    licon.consultaErrores()

            else:
                print(RED+"WRONG SELECTION. REPEAT")


        except KeyboardInterrupt:

            print("Interrupcion por teclado")
            continuar = False # Fin bucle principal

        except NameError:
            #No se ha seleccionado nada
            print("\n")

        except:
            print("Error no definido")

    print(WHITE +"Guardando fichero de tiempo de ejecucion")

    # Cerrar fichero de tiempo de ejecucion
    ficheroTempEjecucion.write( "\n\n Fin de medida " +
                                time.strftime("%c"))
    ficheroTempEjecucion.close()

    licon.terminarHilo(lock)    # Terminar hilo de conexion
    licon.cerrarSocket()        # Cerrar socket de conexion

    print("Fin de programa")


if __name__ == '__main__':
    import sys
    sys.exit(main())
```

ANEXO 2: Código de librería errores.py

```
#########################################################################
#                                                                       #
#   ESCUELA UNIVERSITARIA POLITECNICA DE TERUEL                         #
#   GRADO EN INGENIERIA ELECTRONICA Y AUTOMATICA                        #
#                                                                       #
#   TRABAJO DE FIN DE GRADO                                             #
#   Curso 2017/18                                                       #
#                                                                       #
#   Autor: Miguel Torres Hornero                                        #
#   Fecha: Febrero de 2018                                              #
#                                                                       #
#   Fichero: errores.py                                                 #
#                                                                       #
#   Breve descripcion: Errores de sistema de baterias VUCAB52 e         #
#   inversor BAT50.                                                     #
#                                                                       #
#   Acentos intencionadamente eliminados.                               #
#                                                                       #
#########################################################################

errorText= {\
0 : 'no error',\
11 : 'Temperature_high_L1_HW',\
12 : 'Temperature_high_L1_SW',\
13 : 'Temperature_high_L2_HW',\
14 : 'Temperature_high_L2_SW',\
15 : 'Temperature_high_L3_HW',\
16 : 'Temperature_high_L3_SW',\
17 : 'Temperature_high_PT100_X108A12',\
18 : 'Temperature_high_PT100_X108A23',\
19 : 'Temperature_high_PT100_X108B12',\
20 : 'Temperature_high_PT100_X108B34',\
21 : 'Temperature_high_board',\
22 : 'Temperature_low_L1_SW',\
23 : 'Temperature_low_L2_SW',\
24 : 'Temperature_low_L3_SW',\
25 : 'Temperature_low_PT100_X108A12',\
26 : 'Temperature_low_PT100_X108A23',\
27 : 'Temperature_low_PT100_X108B12',\
28 : 'Temperature_low_PT100_X108B34',\
29 : 'Temperature_low_board',\
31 : 'Current_Inv_L1_HW_driver',\
32 : 'Current grid : see manual',\
33 : 'Current grid : see manual',\
35 : 'Current grid : see manual',\
36 : 'Current grid : see manual',\
37 : 'Current grid : see manual',\
39 : 'Current grid : see manual',\
40 : 'Current grid : see manual',\
41 : 'Current grid : see manual',\
43 : 'Current DC: see manual',\
44 : 'Current DC: see manual',\
45 : 'Current unsysm',\
46 : 'Current grid sum',\
66 : 'Overvoltage_grid_L1,_stage_1',\
67 : 'Overvoltage_grid_L1,_stage_2',\
```

```
68 : 'Overvoltage_grid_L1,_stage_3',\
71 : 'Overvoltage_grid_L2,_stage_1',\
72 : 'Overvoltage_grid_L2,_stage_2',\
73 : 'Overvoltage_grid_L2,_stage_3',\
76 : 'Overvoltage_grid_L3,_stage_1',\
77 : 'Overvoltage_grid_L3,_stage_2',\
78 : 'Overvoltage_grid_L3,_stage_3',\
81 : 'Over voltage DC Board',\
82 : 'Over voltage DC SW',\
116 : 'Undervoltage_grid_L1,_stage_1',\
117 : 'Undervoltage_grid_L1,_stage_2',\
121 : 'Undervoltage_grid_L2,_stage_1',\
122 : 'Undervoltage_grid_L2,_stage_2',\
126 : 'Undervoltage_grid_L3,_stage_1',\
127 : 'Undervoltage_grid_L3,_stage_2',\
131 : 'Under voltage DC Board',\
132 : 'Under voltage DC SW',\
151 : 'DC_circuit_breaker_tripped',\
155 : 'AC_circuit_breaker_tripped',\
156 : 'Emergency_Stop_S1',\
157 : 'OV-Protection_/_Filter_Fuses',\
158 : 'Insulation Error',\
161 : 'Inverter_HW_Error_L1',\
162 : 'Inverter_HW_Error_L2',\
163 : 'Inverter_HW_Error_L3',\
164 : 'Port_Idc_error',\
169 : 'Precharge_DC-Link',\
171 : 'Grid_rotation',\
172 : 'Switch_on_condition failed',\
173 : 'Vector_jump',\
174 : 'Frequency_error',\
175 : 'Island_Grid_detected',\
179 : 'CPLD_Supply_Error',\
184 : 'Fehler nicht definiert!',\
186 : 'CAN_BUS_time_out',\
196 : 'Unknown_Inverter_Model',\
197 : 'Inverter_Model_Wrong',\
203 : 'Communication_ModbusTCP_failed',\
208 : 'Inverter(s) report error(s)',\
219 : 'Communication Card Update active',\
235 : 'Switch off grid fault',\
}


warningtext = {\
0 :  'no warning',\
12 : 'temperature_high_L1',\
14 : 'temperature_high_L2',\
16 : 'temperature_high_L3',\
17 : 'temperature_high_PT100_X108A12',\
18 : 'temperature_high_PT100_X108A23',\
19 : 'temperature_high_PT100_X108B12',\
20 : 'temperature_high_PT100_X108B34',\
21 : 'temperature_high_board',\
22 : 'temperature_low_L1',\
23 : 'temperature_low_L2',\
24 : 'temperature_low_L3',\
25 : 'temperature_low_PT100_X108A12',\
26 : 'temperature_low_PT100_X108A23',\
27 : 'temperature_low_PT100_X108B12',\
28 : 'temperature_low_PT100_X108B34',\
```

```
29 : 'temperature_low_board',\
164 : 'Internal CAN bus receive error',\
165 : 'Internal CAN bus message lost',\
166 : 'Internal CAN bus wrong ID',\
167 : 'Wrong calibration',\
168 : 'Synchronize error',\
169 : 'Converter overload',\
170 : 'Overvoltage protection fault',\
200 : 'CAN communication lost',\
201 : 'no_access_ext_memory',\
202 : 'no_time_sync',\
204 : 'config_locked_(counter)',\
208 : 'inverter_reports_warnings',\
}
```

ANEXO 3: Tiempos de ejecución

CONEXIÓN DIRECTA	
MEDIDA	TIEMPO(ms)
sL1-N_DIREC_0_to_5_kw	0,000714064
AL1-N_DIREC_0_to_5_kw	0,000896931
AL1-N_DIREC_0_to_10_kw	0,000828981
AL1-N_DIREC_0_to_15_kw	0,000804901
AL1-N_DIREC_0_to_20_kw	0,000957012
AL1-N_DIREC_0_to_30_kw	0,000632048
AL1-N_DIREC_0_to_25_kw	0,000657797
AAL1-N_DIREC_0_to_30_kw	0,000804186
AL1-N_DIREC_0_to_35_kw	0,000873089
AL1-N_DIREC_0_to_40_kw	0,000736952
BL1-N_DIREC_0_to_-5_kw	0,000762224
BL1-N_DIREC_0_to_-10_kw	0,000835896
BL1-N_DIREC_0_to_-15_kw	0,00074482
BL1-N_DIREC_0_to_-20_kw	0,000874996
BL1-N_DIREC_0_to_-25_kw	0,000813007
BL1-N_DIREC_0_to_-30_kw	0,000874996
BBL1-N_DIREC_0_to_-30_kw	0,000790834
BL1-N_DIREC_0_to_-35_kw	0,000841141
BBL1-N_DIREC_0_to_-35_kw	0,000586987
BL1-N_DIREC_0_to_-40_kw	0,000882149
BBL1-N_DIREC_0_to_-40_kw	0,000819206
CL1-N_DIREC_5_to_0_kw	0,000821114
CCL1-N_DIREC_5_to_0_kw	0,0007689
CL1-N_DIREC_10_to_0_kw	0,000845909
CL1-N_DIREC_15_to_0_kw	0,000718117
CCL1-N_DIREC_15_to_0_kw	0,000751019
CL1-N_DIREC_20_to_0_kw	0,00070405
CL1-N_DIREC_25_to_0_kw	0,000884056
CCL1-N_DIREC_25_to_0_kw	0,000853777
CL1-N_DIREC_30_to_0_kw	0,00061202
CCL1-N_DIREC_30_to_0_kw	0,000845909
CL1-N_DIREC_35_to_0_kw	0,000679016
CCL1-N_DIREC_35_to_0_kw	0,000861883
CL1-N_DIREC_40_to_0_kw	0,000709057
DL1-N_DIREC_-5_to_0_kw	0,00078392

CONEXIÓN MODBUS	
MEDIDA	TIEMPO(ms)
AL1-N_MODB_0_to_5_kw	0,00069904
AL1-N_MODB_0_to_10_kw	0,00063181
AL1-N_MODB_0_to_15_kw	0,0007751
AL1-N_MODB_0_to_20_kw	0,00085187
AL1-N_MODB_0_to_25_kw	0,00081801
AL1-N_MODB_0_to_30_kw	0,00084591
AL1-N_MODB_0_to_35_kw	0,00058579
AL1-N_MODB_0_to_40_kw	0,00069594
BL1-N_MODB_0_to_-5_kw	0,00078607
BL1-N_MODB_0_to_-10_kw	0,00082088
BL1-N_MODB_0_to_-15_kw	0,00081205
BL1-N_MODB_0_to_-20_kw	0,00080419
BL1-N_MODB_0_to_-25_kw	0,00073695
BBL1-N_MODB_0_to_-25_kw	0,00069189
BL1-N_MODB_0_to_-30_kw	0,00087714
BL1-N_MODB_0_to_-35_kw	0,00077987
BL1-N_MODB_0_to_-40_kw	0,00074601
BBL1-N_MODB_0_to_-40_kw	0,00079608
CL1-N_MODB_5_to_0_kw	0,00070095
CL1-N_MODB_10_to_0_kw	0,00074887
CL1-N_MODB_15_to_0_kw	0,00070095
CL1-N_MODB_20_to_0_kw	0,00083303
CL1-N_MODB_25_to_0_kw	0,00069785
CL1-N_MODB_30_to_0_kw	0,00081086
CL1-N_MODB_35_to_0_kw	0,00062585
CL1-N_MODB_35_to_0_kw	0,00070715
DL1-N_MODB_-5_to_0_kw	0,00083613
DL1-N_MODB_-10_to_0_kw	0,000875
DL1-N_MODB_-15_to_0_kw	0,00084591
DL1-N_MODB_-15_to_0_kw	0,00122213
DL1-N_MODB_-20_to_0_kw	0,00080395
DL1-N_MODB_-25_to_0_kw	0,00155687
DL1-N_MODB_-30_to_0_kw	0,00069404
DDL1-N_MODB_-30_to_0_kw	0,00081992
DL1-N_MODB_-35_to_0_kw	0,00078297

DL1-N_DIREC_-10_to_0_kw	0,000833035
DL1-N_DIREC_-15_to_0_kw	0,000667095
DL1-N_DIREC_-20_to_0_kw	0,000718832
DDL1-N_DIREC_-20_to_0_kw	0,000830173
DL1-N_DIREC_-25_to_0_kw	0,000838995
DDL1-N_DIREC_-25_to_0_kw	0,000718832
DL1-N_DIREC_-30_to_0_kw	0,000672817
DDL1-N_DIREC_-30_to_0_kw	0,000812054
DL1-N_DIREC_-35_to_0_kw	0,000722885
DL1-N_DIREC_-40_to_0_kw	0,000819206
DL1-N_DIREC_-40_to_0_kw	0,000761986
EL1-N_DIREC_5_to_-5_kw	0,000791073
EL1-N_DIREC_10_to_-10_kw	0,000603914
EL1-N_DIREC_15_to_-15_kw	0,000665903
EL1-N_DIREC_20_to_-20_kw	0,000850916
EL1-N_DIREC_25_to_-25_kw	0,000913143
EL1-N_DIREC_30_to_-30_kw	0,000830889
EEL1-N_DIREC_30_to_-30_kw	0,000721931
EL1-N_DIREC_30_to_-30_kw	0,000769138
EL1-N_DIREC_35_to_-35_kw	0,00075388
EL1-N_DIREC_40_to_-40_kw	0,000777006
FL1-N_DIREC_-5_to_5_kw	0,000830889
FFL1-N_DIREC_-5_to_5_kw	0,000692129
FL1-N_DIREC_-10_to_10_kw	0,000778198
FL1-N_DIREC_-15_to_15_kw	0,000865936
FL1-N_DIREC_-20_to_20_kw	0,000540018
FFL1-N_DIREC_-20_to_20_kw	0,00082016
FL1-N_DIREC_-25_to_25_kw	0,000825167
FFFL1-N_DIREC_-25_to_25_kw	0,000851154
FL1-N_DIREC_-30_to_30_kw	0,000806093
FL1-N_DIREC_-35_to_35_kw	0,000613213
FL1-N_DIREC_-40_to_40_kw	0,000771999
MEDIA	0,000776711
DESVIACIÓN	8,67576E-05

Tabla 29: Tiempos de ejecución de conexión directa (ms)

DDL1-N_MODB_-35_to_0_kw	0,00069404
DDDL1-N_MODB_-35_to_0_kw	0,00082588
EL1-N_MODB_5_to_5_kw	0,00068092
EEL1-N_MODB_10_to_10_kw	0,00066996
DL1-N_MODB_5_to_-5_kw	0,00082898
DL1-N_MODB_10_to_-10_kw	0,00085282
DL1-N_MODB_15_to_-15_kw	0,00078106
EL1-N_MODB_20_to_-20_kw	0,00071096
EL1-N_MODB_15_to_-15_kw	0,00076985
EL1-N_MODB_10_to_-10_kw	0,00076318
EL1-N_MODB_25_to_-25_kw	0,00068092
EL1-N_MODB_30_to_-30_kw	0,00072885
EEL1-N_MODB_30_to_-30_kw	0,00076008
EEEL1-N_MODB_30_to_-30_kw	0,000808
FL1-N_MODB_-5_to_5_kw	0,00086689
FL1-N_MODB_-10_to_10_kw	0,00077105
FL1-N_MODB_-15_to_15_kw	0,00063491
FL1-N_MODB_-20_to_20_kw	0,00077009
FL1-N_MODB_-25_to_25_kw	0,00081897
FFL1-N_MODB_-25_to_25_kw	0,00071287
FL1-N_MODB_-30_to_30_kw	0,00070596
FL1-N_MODB_-35_to_35_kw	0,00082302
FL1-N_MODB_-40_to_40_kw	0,00073695
MEDIA	0,00078299
DESVIACIÓN	0,00013813

Tabla 28: Tiempos de ejecución de conexión Modbus (ms)

ANEXO 4: Tiempo de respuesta de osciloscopio

Medida	Tiempo (ms)
1	0,000445127
2	0,000691175
3	0,000636816
4	0,000629187
5	0,000638962
6	0,000576973
7	0,000560045
8	0,000571966
9	0,00069499
10	0,00063014
11	0,000555992
12	0,00061512
13	0,000611067
14	0,000617027
15	0,000484943
16	0,000643015
17	0,000617027
18	0,000633001
19	0,000567913
20	0,000610828
21	0,000526905
22	0,00067997
23	0,000608921
24	0,000658035
25	0,000611067
26	0,000674963
27	0,000609875
28	0,000635862
29	0,000712872
30	0,000617981
Media	0,000612259
Desviación	5,87002E-05

Tabla 30: Tiempos de respuesta de osciloscopio (ms)

Tiempo de respuesta de sistema de almacenamiento e inversor bidireccional trifásico

Miguel Torres Hornero