INGENIERÍA EN TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

Universidad Zaragoza

TRABAJO FIN DE GRADO

REGULADOR DE POTENCIA EN PLANTAS FOTOVOLTAICAS

POWER REGULATOR IN PHOTOVOLTAIC PLANTS

Rubén Grávalos Ruiz

25/06/2020

Director: Juan Calvo Medel
Ponente: Isidro Urriza Parroqué

RESUMEN

Este trabajo de fin de grado se ha desarrollado en un entorno industrial y se ha apoyado concretamente desde el departamento de ingeniería de Ríos Renovables Group, quienes han facilitado documentación, dispositivos y cursos para que haya sido posible llevarlo a cabo.

En el presente documento se recoge tanto motivación como proceso, pruebas, resultados, función y aspectos técnicos de la programación de un controlador de potencia para plantas fotovoltaicas.

El trabajo ha consistido en la elaboración de un controlador de la potencia activa que los inversores de los paneles solares inyectan a la red. Existe un límite máximo para el total de la potencia producida por la planta, el cual es enviado constantemente vía Modbus TCP al autómata, para que este lo interprete e indique a los inversores el rendimiento al que tienen que trabajar. Por otro lado, toda esta información de potencia en tiempo real, además de la capacidad de añadir o eliminar inversores y manipular las direcciones de red del sistema, se brindan al usuario por parte del mismo autómata trabajando a modo de servidor web.

ABSTRACT

This final bachelor project has been developed in an industrial environment and has been specifically supported by the engineering department of Ríos Renovables Group, who have provided documentation, devices and courses so that it has been possible to carry it out.

This document includes both motivation and process, tests, results, function and technical aspects of programming a power controller for photovoltaic plants.

The work has consisted of developing an active power controller that the solar panel inverters inject into the grid. There is a maximum limit for the total power produced by the plant, which is constantly sent via Modbus TCP to the automaton, so that it can interpret it and indicate to the inverters the performance at which they have to work. On the other hand, all this power information in real time, in addition to the ability to add or remove inverters and manipulate the network addresses of the system, are provided to the user by the same controller working as a web server.

ÍNDICE GENERAL

CAPÍTULO 1

INTRODUCCIÓN

1.1. Marco

La demanda de energía en la actualidad, en un momento en el que máquinas y tecnología son el foco de la mayor parte de los sectores, alcanza límites muy elevados. Quizá por esa elevada demanda, el agotamiento de algunas fuentes de energía no renovables y el impacto que a lo largo de los años se ha podido observar en la naturaleza, toma una elevada importancia la búsqueda de energías de carácter renovable.

Concretamente la energía solar se sitúa en una muy buena posición al ser inagotable, ya que es el sol la fuente de energía. Casi al nivel de importancia del hecho de ser inagotable, encontramos la ventaja frente a una de las mayores preocupaciones medioambientales, ya que es una fuente libre de emisiones de CO_2 [1]. "Nuestra estrella, llamada Sol, irradia una potencia enorme: una emisión constante de $3{,}84*10^{23}$ kilovatios (KW)" [2]. La irradiación solar media típica de Europa central es de unos 1100 KWh/(m^2·año), pudiendo llegar en zonas próximas a los trópicos incluso a los 2300 KWh/(m^2·año), lo que en base a la demanda mundial de energía anual, supone unas diez mil veces más.

Es en 1921 cuando Albert Einstein recibe el Premio Nobel de Física por explicar el efecto fotovoltaico [3], basado en la absorción de cuantos de energía por parte de los electrones de manera directamente proporcional a la frecuencia de la fuente lumínica. Así pues, la encargada de realizar el trabajo de manipulación para captar la radiación del sol y transformarla en energía eléctrica utilizable en nuestros elementos tecnológicos es la célula fotovoltaica. La esencia de este tipo de célula es que está formada por dos materiales semiconductores, cada uno cargado con cargas opuestas. El fotón incidente sobre un material libera un electrón que va en busca de un hueco que llenar en el otro material semiconductor. Éste fenómeno produce una diferencia de potencial, y por lo tanto, energía eléctrica. De este modo, se incrementa la energía eléctrica producida en función del acoplamiento de más células fotovoltaicas a la misma estructura.

1.2. Motivación y objetivos

Teniendo en cuenta los datos anteriores, se entiende claramente que se demanda muchísima menos energía solar de la que el sol oferta. Parece que según esto, podríamos generar energía eléctrica infinita, reinventar la tecnología para que fuese este el combustible de todos los artefactos e incluso almacenarla para un uso futuro. De no haber ningún tipo de control, el sistema de las placas fotovoltaicas sería algo más sencillo: convertir de manera automática toda la radiación solar en energía eléctrica y pincharla a la red para que llegue a los centros de distribución eléctricos. Pero la realidad es diferente.

El problema aparece cuando se descubre que no es posible almacenar grandes cantidades de energía eléctrica como tal debido a su naturaleza [4], teniendo pues que convertirla en otro tipo de energía, como la mecánica o la química, lo cual dificulta y encarece el proceso teniendo en cuenta el carácter infinitamente inagotable de la fuente. Por lo tanto, para que el ejercicio resulte eficiente suponiendo que se consume todo lo que se produce, habría que establecer un equilibrio preciso y continuamente actualizado de consumo y producción. En el caso de haber un desequilibrio, se traduciría en desvío de frecuencia respecto del valor nominal de 50 Hz [5]. El organismo público encargado de asegurar el cumplimiento de lo anterior, además de estimar la evolución del consumo en el territorio es conocido como Red Eléctrica de España (REE). El modo en el que opera es el siguiente: envía de manera frecuente consignas a los centros de producción para ajustar su producción y así mantener un margen de generación suficiente para poder subsanar posibles contratiempos en cuanto al consumo previsto.

Para hacer más ilustrativa la explicación de los objetivos del trabajo, resulta útil partir de la comprensión de los elementos que intervienen, mostrados en la figura 1, las funciones que desempeñan y el orden en que se comunican unos con otros.

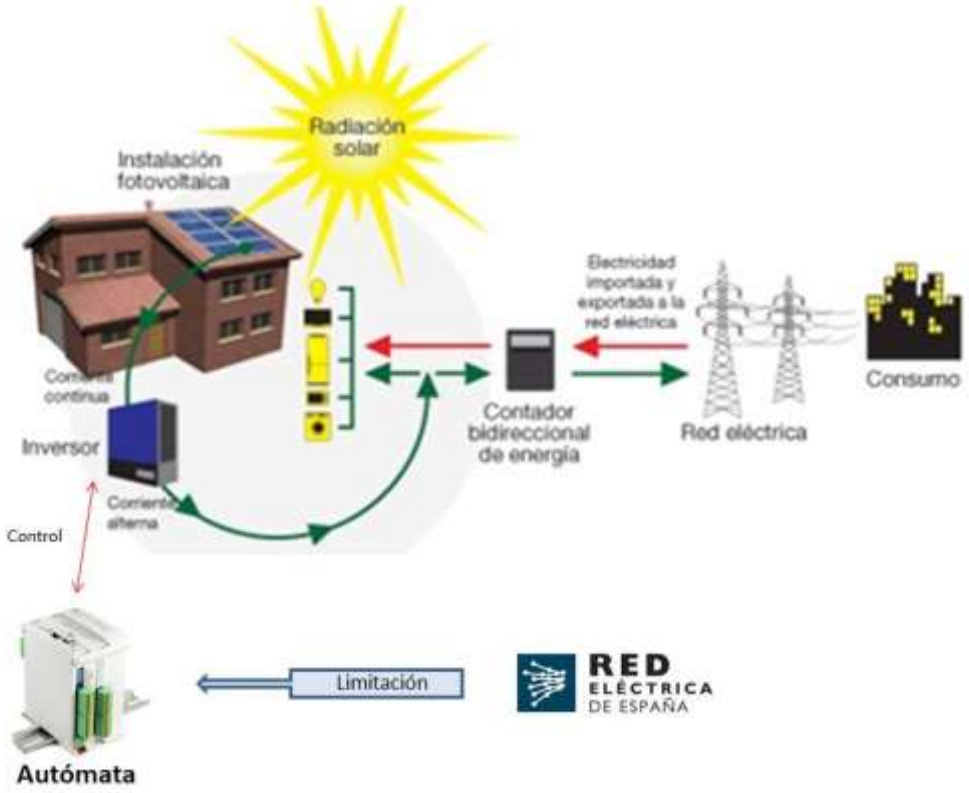


Figura 1: estructura de una planta fotovoltaica

De acuerdo a lo anterior, el objetivo del trabajo ha sido diseñar un sistema que se encargue de asegurar el cumplimiento de las limitaciones impuestas por Red Eléctrica España, controlando el modo de operación de los inversores de potencia situados en los paneles solares, los cuales son los encargados de convertir la tensión continua que procede de los módulos de las placas solares en tensión alterna senoidal, con las características de la que proporciona la red eléctrica convencional. Además, también se ha buscado brindar la posibilidad al usuario la posibilidad de realizar un seguimiento y una gestión cómoda de los parámetros característicos de los inversores, diseñando un servidor web.

Una producción total de una planta por encima del valor que le ha limitado Red Eléctrica podría suponer una multa de una cuantía muy significativa, acorde al problema que podría acarrear comentado anteriormente. Por consiguiente, la tarea más importante del sistema, para la empresa, será asegurar que en cada una de sus plantas se produce el máximo de potencia que se le asigna, lo que supone el máximo beneficio económico.

Para mantener informada en todo momento a la empresa propietaria, en lo que respecta a cambios en la limitación de potencia de una planta o a posibles fallos de los inversores, será interesante mantener informado al personal que se ocupe de la gestión y supervisión de la planta.

Además, para el usuario resulta muy útil monitorizar todos los valores de potencia en cuestión, y muy útil añadir o eliminar inversores, o simplemente modificar sus parámetros, así como los del propio sistema.

1.3. Alcance

Este trabajo consiste en el diseño y la implementación de una versión autónoma de sistema capaz de controlar la cantidad de potencia eléctrica que produce cada uno de los inversores del área mediante comunicación directa vía Modbus TCP para así poder producir el máximo con el total de la planta fotovoltaica sin rebasar el límite que marca Red Eléctrica España, que ha sido realizado fundamentándose conceptualmente en las indicaciones y el apoyo del director, vía telemática, quien ha tomado como referencia el sistema que tienen actualmente operando en las plantas fotovoltaicas de la empresa. Esto ha consistido en familiarizarse con protocolos y manejo de programas complementarios necesarios para comprobar el correcto funcionamiento del sistema diseñado, puesto que por circunstancias imprevistas debido a la crisis del COVID-19, no ha sido posible personarse en un parque fotovoltaico en funcionamiento para someter a prueba el trabajo.

La parte física del sistema consta de un controlador lógico programable (PLC), o autómata, como nos referiremos en muchas de las ocasiones a él, de la marca Industrial Shields que lleva en su interior una placa Arduino Mega. El PLC está continuamente escuchando las limitaciones de potencia que le llegan vía protocolo Modbus TCP por el puerto 502 por parte de REE, donde encuentra el dato de potencia máxima que puede generarse en ese momento en esa planta. Una vez recibido el dato de potencia, chequea todos los inversores de los que dispone para asegurar cuántos de ellos se encuentran en correcto funcionamiento, divide el dato de potencia máxima entre el número de inversores operativos y anota el porcentaje al que cada uno de ellos debe rendir en función de sus potencias nominales, que no tienen por qué ser las mismas. Esta comunicación también se hace a través del protocolo Modbus TCP y el dato escrito en los inversores va en tanto por mil (es decir, pedir al inversor que trabaje al 100% de su potencia nominal equivaldría a escribir el dato 1000).

Por otro lado, el autómata funciona a modo de servidor web, disponiendo de páginas escritas en HTML almacenadas en una tarjeta micro SD. Entre ellas, se puede encontrar la de inicio de sesión, con un nombre de usuario y una contraseña, que permiten el paso a la página principal, en la que se da la opción de elegir entre 3 departamentos. Uno de ellos, es el de gestión de las direcciones del sistema (del propio autómata), donde se pueden modificar las direcciones IP, DNS, GATEWAY, SUBNET MASK y EMAIL (al que enviar cualquier tipo de información relevante que suceda en el parque, como una nueva limitación de potencia o el mal funcionamiento de un inversor, por ejemplo. Estas direcciones pueden guardarse para la próxima sesión pulsando el botón que así lo indica, y que hace que vayan a parar a memoria EEPROM. Otro de los departamentos a

los que se tiene la opción de acceder es al de gestión de inversores, donde se ofrece la posibilidad de añadir inversores indicando el nombre, la dirección IP, el puerto para la comunicación con el autómata y el valor de potencia nominal. Además, aparece una tabla en la parte inferior de la página donde se muestran los inversores con los que cuenta el sistema, donde se indican todos los parámetros anteriores además del estado de conexión en el que se encuentra cada uno de ellos, y un botón que ofrece la posibilidad de eliminarlo. Cualquier modificación en la lista puede salvarse para la próxima sesión pulsando un botón en la parte inferior de la página, que escribe los datos en la memoria micro SD. Por último, se puede acceder al departamento en el que se sigue en tiempo real la potencia activa producida por cada uno de los inversores a modo de tabla, la potencia máxima permitida por REE y la total producida por la planta. Por defecto, la página se refresca cada 5 segundos, pero cabe la posibilidad de desactivar el refresco y volverlo a activar cuando se desee a través de un botón de tipo “check” en la parte superior izquierda de la pantalla.

1.4. Metodología

El trabajo consta de varias partes, para las cuales es empleado el mismo procedimiento, formado por un proceso de instrucción/formación, en donde familiarizarse con los aspectos pertinentes, otro de prueba y error, en donde asegurar un correcto manejo y desarrollo de lo aprendido conceptualmente, y por último un proceso de perfeccionamiento y retocado para el cumplimiento de la tarea de interés, ajustando la presentación en torno al acabado deseado.

En primer lugar es muy útil comenzar siguiendo unos cursos online impartidos por Industrial Shields, formados por 10 capítulos en los que le aportan al usuario unos conocimientos básicos necesarios para la programación en Arduino en entornos industriales. Comienzan indicando el material que podría ayudarle al usuario a empezar un proyecto industrial, como librerías y ejemplos ya creados, y el modo en el que se debe conectar la placa Arduino al PC. Continúan con las entradas y salidas de las que dispone la placa, el voltaje al que trabajan y el modo de utilizarlas, incluyendo precauciones que deben tomarse antes de lanzarse al conexionado para evitar fallos que en ocasiones pueden ser irreversibles. Dos de los capítulos están dedicados a la utilización de variables, y los 4 últimos se centran en las comunicaciones, con todo lo que es necesario saber para la utilización del protocolo Ethernet, así como Serial TTL, RS-232 y RS-485. Concretamente en el apartado dedicado al control remoto de entradas-salidas usando Ethernet, se analiza un ejemplo de comunicación Modbus TCP entre un Master y un Slave, que sirve de gran ayuda para entender y probar el funcionamiento del protocolo en el que se comunica el autómata con los inversores.

Una vez recibido el curso y con la ayuda de la referencia que se puede encontrar en la página web de Arduino para funciones, variables y estructuras, es de gran ayuda comprender cada uno de los ejemplos relacionados con los aspectos que van a tratarse

en el proyecto. Estos ejemplos vienen integrados en programa al descargarlo, y algunos pueden descargarse aparte desde GitHub, como son los de la librería Tools40 cuya descarga está recomendada por el mismo curso mencionado anteriormente. De entre todos los ejemplos que ofrece el software, son de especial interés los relacionados con SD, en donde aprender a crear, abrir, eliminar, cerrar y eliminar un fichero, además de leer y escribir en él; con la memoria EEPROM, en la que se aprende a acceder a espacios de memoria con el fin de leer, escribir y actualizar datos; con servidor web utilizando Ethernet, gracias al cual se entiende a la perfección el funcionamiento mediante un ejemplo que consta de una sola página en la que se muestra simplemente el valor leído en una serie de entradas analógicas de la placa, a partir del cual se puede conseguir mostrar la página que se desee editando el código del ejemplo; y como ya se ha comentado anteriormente, con los de la librería Tools40, en donde se encuentran ejemplos de todo tipo en cuanto al empleo de las diferentes funciones de las que se dispone para la solicitud de lectura y escritura de diferentes datos (analógicos y digitales) entre Master y Slaves.

Llegado al punto de haber adquirido los conocimientos básicos e interiorizado el funcionamiento de funciones y protocolos necesarios para llevar a cabo el proyecto en sí, cobra gran importancia el tema del servidor web y la estructura de árbol que tiene que ser diseñada para mostrar las páginas del modo deseado.

El resto se compone de completar y darle el formato deseado a las páginas, añadir atributos adicionales y realizar pruebas con programas que actúan como Slaves y Master, para simular el comportamiento de los inversores y de REE y de esta manera comprobar el correcto funcionamiento del sistema.

1.5. Herramientas

En cuanto al hardware utilizado, se compone principalmente del autómata industrial de la casa Industrial Shields, un M-DUINO PLC Arduino Ethernet 21 I/Os Analog/Digital PLUS [6] como el de la figura 1, proporcionado por Ríos Renovables e impulsado por un una fuente de alimentación de la marca Leader Electronics Inc., la cual se conecta a la red eléctrica doméstica y proporciona al circuito del autómata 12 voltios a 1000 mA.

En la figura 1 se muestra la parte física del autómata.

Figura 2: M-DUINO PLC Arduino Ethernet 21 I/Os Analog/Digital PLUS.

Además, un cable Ethernet de 20 metros de longitud lo conecta al router de casa, y otro cable USB de tipo B es el que lo conecta con el PC desde el que se realiza toda la programación y en el que reside todo el software necesario para someter el sistema a prueba. El autómata está diseñado para su utilización en un entorno profesional, y cuenta con 21 E/S1 y diferentes modos de comunicación. Además, dispone de la capacidad de expandirse con 127 módulos mediante el bus I2C, lo que hace que en modo de maestro-esclavo (cliente-servidor, respectivamente), pueda gobernar hasta 6604 E/S. El autómata lleva interna una placa Arduino Mega 2560 (figura 2) cuyo microcontrolador es un ATmega2560 que trabaja a la velocidad de 16 MHz. Las memorias con las que cuenta la placa son Flash, con 256 KB de capacidad de los cuales 8 KB son utilizados por el bootloader; una memoria SRAM con capacidad para 8 KB; y una EEPROM con una capacidad de 8 KB.

Figura 3: placa Arduino Mega 2560.

En la figura 4 se muestra el conexionado de todos los elementos citados anteriormente.

Figura 4: conexión de los elementos en el entorno de trabajo.

Deberían de haber formado parte de esta sección de hardware también los inversores de los paneles solares de alguna planta en funcionamiento en la que nos habríamos personado de haber sido posible para testear la correcta operación del proyecto, pero las medidas tomadas como consecuencia del COVID-19 no lo han permitido.

Es software empleado está formado, entre otros y como bien se ha comentado, por la plataforma Arduino IDE [7], un entorno de desarrollo integrado open-source escrito en Java y basado en Processing [8], entre otros softwares de tipo open-source. Puede utilizarse en Windows, Mac OS y Linux, y admite los lenguajes C y C ++ utilizando reglas especiales de estructuración de códigos. Es utilizado para escribir y cargar programas en las placas compatibles con Arduino.

Otra de las aplicaciones utilizadas es Modbus Slave [9], de modbus tools, que permite simular hasta 32 dispositivos de tipo Slave. Permite ejecutar y someter a prueba el programa desarrollado en PLCs mediante un periodo de prueba de 30 días, a partir de los cuales pasa a ser de pago.

La sustituta de la anterior se llama Easy Modbus Server Simulator [10], del paquete Easy Modbus TCP .NET, un software tipo open-source utilizado y probado en varias industrias (Barcode Scanner Application, IoT / Industry 4.0, Data storage of PLC-data in SQL-Database, Custom made Scada applications, Energy data acquisition, etc). Permite acceso rápido y seguro a muchos sistemas PLC desde un PC. En este mismo paquete va incluido el Easy Modbus Client Simulator, que desempeña la función de Master y sería el encargado de actuar a modo de REE, pero Ríos Renovables

proporcionó una versión pirata de Modbus Poll, de la misma casa que Modbus Slave. La decisión de emplear este software se tomó teniendo en cuenta que permite realizar una configuración más completa, y que es la herramienta con la que la empresa estaba acostumbrada a trabajar.

1.6. Plan del trabajo

Para disponer de una buena organización de las tareas que se han seguido de manera secuencial durante la realización del proyecto, se elaboró un boceto con la ayuda del software de administración de proyectos de la casa Microsoft, MS Project. Conforme se superaban los diferentes pasos del plan inicial y se añadían nuevos que iban surgiendo, se modificaba o se añadía a la línea base el tiempo que había costado llevar a cabo la tarea, respectivamente. Este software ofrece una planificación de proyectos muy ilustrativa y ordenada, y fue elegido para la realización de este Trabajo de Fin de Grado ya que durante la realización del Erasmus en el primer semestre en el Politécnico di Torino, adquirí los conocimientos suficientes para su utilización cursando la asignatura Project Management.

En la figura 5 se muestra el planning definitivo del proyecto, con las tareas superadas y en los casos en los que es necesario, sus tareas precedentes.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		**TRABAJO DE FIN DE GRADO**	**78 días**	**jue 05/03/20**	**sáb 20/06/20**
2		ESTUDIO DE LA DOCUMENTACIÓN DEL PLC	7 días	jue 05/03/20	vie 13/03/20
3		ESTUDIO DEL PROTOCOLO MODBUS TCP	2 días	lun 16/03/20	mar 17/03/20
4		CURSO DE INICIACIÓN	7 días	mié 18/03/20	jue 26/03/20
5		EJEMPLOS DE ARDUINO IDE	6 días	vie 27/03/20	vie 03/04/20
6		ESCRITURA POR PANTALLA	1 día	vie 27/03/20	vie 27/03/20
7		SENTENCIAS BÁSICAS	1 día	vie 27/03/20	vie 27/03/20
8		SD READ/WRITE	2 días	lun 30/03/20	mar 31/03/20
9		EEPROM	2 días	lun 30/03/20	mar 31/03/20
10		MODBUS TCP	2 días	mar 31/03/20	mié 01/04/20
11		SERVIDOR WEB	2 días	jue 02/04/20	vie 03/04/20
12		COMUNICACIÓN MODBUS TCP	7 días	lun 06/04/20	mar 14/04/20
13		ESTUDIO DE LA DOCUMENTACIÓN DE LOS INVERSORES	1 día	lun 06/04/20	lun 06/04/20
14		APRENDIZAJE DE MODBUS TCP TOOLS	2 días	mar 07/04/20	mié 08/04/20
15		PRUEBAS READ/WRITE	4 días	jue 09/04/20	mar 14/04/20
16		HTML	4 sem.	mié 15/04/20	mar 12/05/20
17		APRENDIZAJE DE CONCEPTOS BÁSICOS	3 días	mié 15/04/20	vie 17/04/20
18		ELABORACIÓN DE PRIMERAS PÁGINAS	7 días	lun 20/04/20	mar 28/04/20
19		DISEÑO DE PÁGINAS DEFINITIVAS	11 días	mar 28/04/20	mar 12/05/20
20		PRUEBAS DE ROBUSTEZ	3 días	mié 13/05/20	vie 15/05/20
21		ÚLTIMOS ACABADOS	3 días	lun 18/05/20	mié 20/05/20
22		PRUEBAS DEFINITIVAS CON INVERSORES	3 días	jue 21/05/20	lun 25/05/20
23		ELABORACIÓN DE LA MEMORIA	15 días	mar 26/05/20	lun 15/06/20
24		ACABADOS DE LA MEMORIA	5 días	mar 16/06/20	sáb 20/06/20

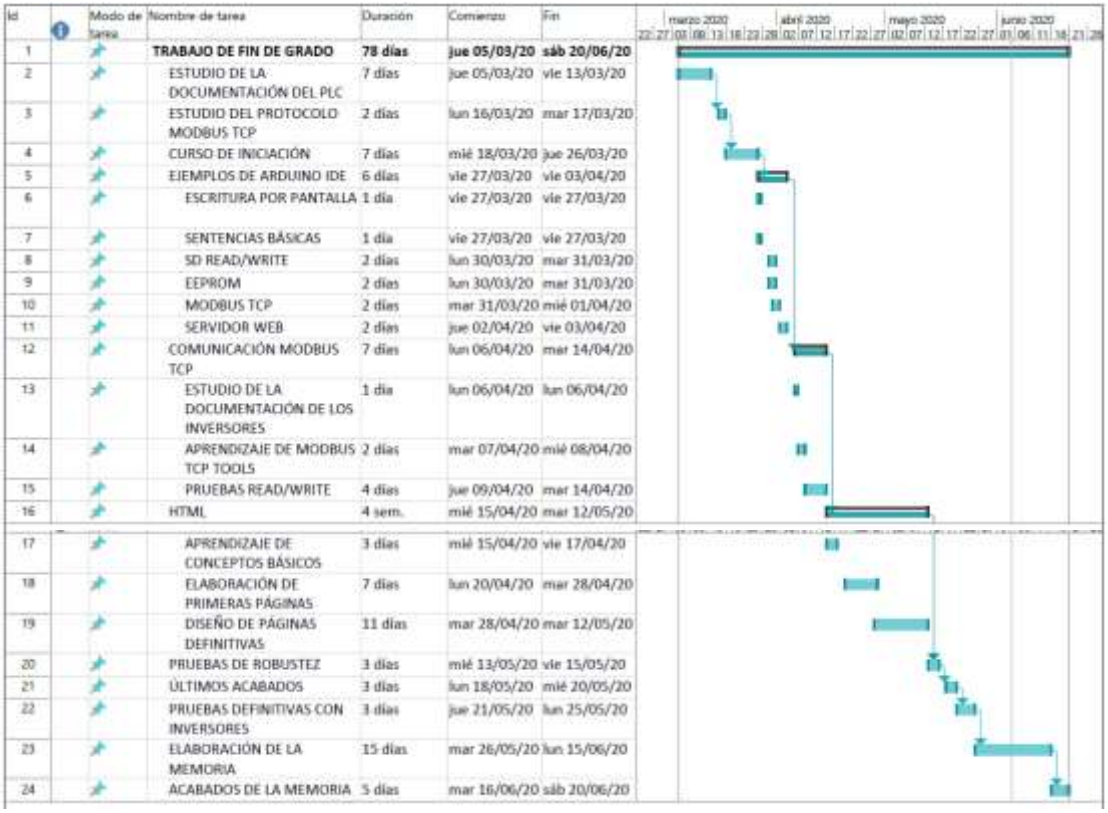


Figura 5: planificación del trabajo con diagrama de Gantt.

CAPÍTULO 2

ESTADO DEL ARTE

El Power Plant Controller (PPC) es un aparato cuya función principal es monitorizar y controlar los datos de interés de un parque fotovoltaico al completo. Entre ellos, pueden encontrarse parámetros de los dispositivos implicados en la producción de energía o todo tipo de flujo total o parcial de potencia del área para el que se configure. Algunos de los principales son tensión en planta, control de frecuencia, limitación de la producción, factor de potencia y potencias activa y reactiva. A partir de aquí, y teniendo en cuenta el nivel y la competencia de las casas que desarrollan los dispositivos en cuestión, además de la cada vez más importante explotación de energías renovables, se puede encontrar PPCs de lo más vanguardistas.

Gestionar la energía de manera inteligente y digitalizar las centrales es el punto de partida, pero inmersos en un clima en el que parece que la tecnología tiende a la convergencia, cobra un papel muy protagonista la interconexión de diferentes tipos de energía en la misma plataforma.

A día de hoy, los PPC que encontramos en el mercado ofrecen una alta precisión en el manejo de todos los parámetros de red durante el estado de operación.

Flexibilidad, modulación y escalabilidad son unos de los atributos que más se premian de cara a la integración fiable del dispositivo en las diferentes plantas para las que puede ser interesante su uso, no resultando un problema la ampliación del abanico de dispositivos contenidos en la planta.

La independencia del fabricante y la adaptación a diferentes protocolos utilizados en distintos dispositivos son de gran importancia para cumplir con lo anterior, como también la posibilidad de actualización periódica, para mantener la compatibilidad del sistema y tenerlo preparado para el futuro.

El cliente busca realizar ajustes y visualizar datos de manera rápida y concisa, por lo que los PPCs intentan ofrecer una interfaz gráfica al usuario de poca complejidad y con estructura intuitiva, para que su utilización sea sencilla y no requiera conocimientos de programación, sino una simple guía de usuario bien estructurada.

Además de los datos en tiempo real, resulta igual de importante o incluso más disponer de datos pasados almacenados, con el fin de hacer seguimiento de evolución del sistema a largo plazo. Una opción muy de moda es la de incluir un servicio de almacenamiento en una base de datos en línea “cloud”, ofreciendo también la posibilidad de exportación de datos.

La vulnerabilidad no puede ser una característica del sistema, por lo que la administración del acceso de usuarios y el cifrado extremo a extremo en configuración y transferencia de datos es uno de los puntos en los que más se trabaja, en un momento en el que el uso de internet por parte de empresas o entidades de interés para malhechores está en completo auge.

Por último, la transparencia desde el primer momento es algo muy buscado por los clientes, de manera que muchos de los PPCs disponen de modelos de simulación que aseguran la integración a la red de la planta desde el primer momento, minimizándose el riesgo a una mala operatividad desde la fase de planificación.

CAPÍTULO 3

DISEÑO DE LA APLICACIÓN

3.1. Lenguaje de programación del autómata

Como se ha comentado anteriormente, la programación del sistema se ha llevado a cabo en Arduino IDE, en donde se permite la programación en lenguaje C, el cual ha sido elegido por la comodidad que transmite al haberlo utilizado lo suficiente a lo largo de la titulación. Al abrir el programa por primera vez, se muestra un sketch en blanco en donde se diferencian dos partes en las que puede escribirse código. Una de ellas, llamada setup, es la que primero se ejecuta, y solamente lo hace una vez, por lo que en su interior se escribe el código dedicado a la configuración de funcionalidades que lo precisen para un correcto funcionamiento cuando posteriormente se les invoque en la parte recurrente. Por otro lado, tenemos la parte llamada loop, en cuyo interior se escribe el código que se pretende que se ejecute repetidamente [11]. El trabajo consta de unas 1300 líneas de código escritas de manera autónoma.

Durante la programación no se ha utilizado ningún tipo de librería particular descargada de Github, sino solamente las que vienen integradas en el software y las de la librería Tools40 (recomendada por Industrial Shields para el empleo de sus dispositivos en entornos industriales):

```
#include <SPI.h>
```

Esta biblioteca permite la comunicación con dispositivos SPI, con Arduino como dispositivo maestro. Las siglas SPI significan interfaz periférica en serie y dan nombre a un protocolo de datos en serie síncrono utilizado por microcontroladores permitiendo una rápida comunicación con uno o más dispositivos periféricos en distancias cortas.

```
#include <EEPROM.h>
```

Es la biblioteca con la que se controla la memoria de la que dispone el microcontrolador de la placa. Los datos guardados en memoria EEPROM se mantienen aun cuando la placa está apagada. El espacio de almacenamiento del que dispone es de 4 KB, estructurado de manera en que cada dirección de memoria, de la 0 a la 4095, alberga 1 Byte de información. Este tipo de memoria no volátil se caracteriza por tener una velocidad de acceso a lectura y escritura inferior a otro tipo de memorias como SRAM y Flash [12]. En memoria EEPROM, el tiempo empleado para escritura es de unos 3,3 ms

y además dispone de un número limitado de accesos borrar/reescribir que en la hoja de especificaciones de nuestra placa indica que es 100.000 para cada dirección de memoria. Las funciones utilizadas para acceso a lectura y escritura son las que se muestran a continuación, respectivamente:

```
EEPROM.read(address)
```

```
EEPROM.write(address, value)
```

Teniendo en cuenta los dos aspectos comentados sobre este tipo de memoria, es de especial interés la sustitución de la función propia de escritura por una función que funciona del mismo modo pero que al ser llamada, solo escribe si en la dirección de memoria encuentra un valor que difiere del indicado en el argumento. De esta manera se consigue un ahorro en tiempo de acceso y se evita el consumo de ciclos de escritura. La llamada a la función tiene la siguiente sintaxis:

```
EEPROM.update(address, value)
```

Para el sistema solo se han utilizado 22 Bytes de la memoria EEPROM, de la dirección 0 a la 21, donde se guardan las direcciones de red propias del sistema que han de estar disponibles al ponerlo en marcha, para configurarse en la red. El mapa de memoria se distribuye gráficamente de la manera en la que se muestra en la figura 6.

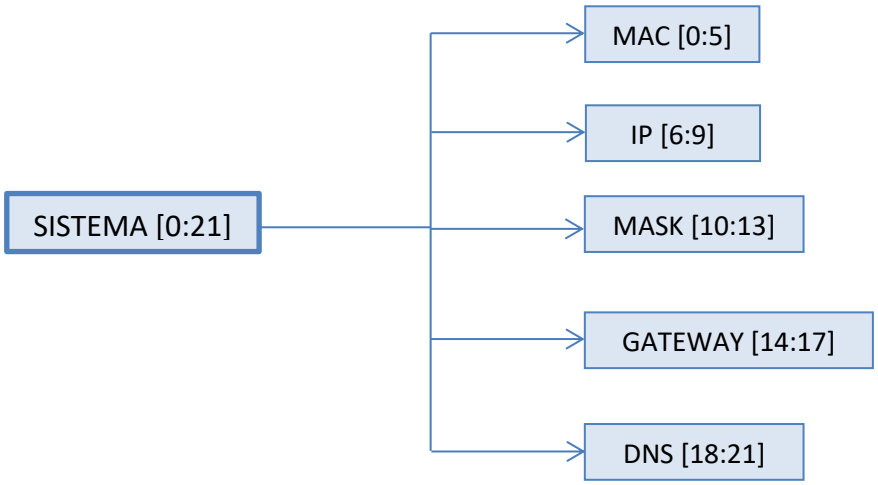


Figura 6: distribución de direcciones en memoria EEPROM.

```
#include <SD.h>
```

Esta es la librería que permite leer y escribir en tarjetas SD. Para la comunicación entre el microcontrolador y la SD se utiliza SPI, que en el caso de nuestra tarjeta Arduino Mega se realiza en los pines 50, 51 y 52. Además, para seleccionar la tarjeta se utiliza un pin SS del hardware, que en nuestro caso es el 53, y se especifica en la llamada a

SD.begin(). Es indispensable dejar este último pin como salida ya que se lo contrario la biblioteca SD no funciona [13].

Una vez iniciada la tarjeta, las funciones utilizadas para abrir y cerrar ficheros son las siguientes:

```
SD.open(filepath, mode)
```

En donde en filepath se especifica la ruta hasta el fichero deseado y en mode se especifica entre si la apertura del fichero es con intención de leer o escribir contenido. En este lugar, se debe escribir FILE_READ o FILE_WRITE, aunque si no se especifica, la apertura por defecto es en modo lectura. Además de estos dos modos, existe otro interesante que se especifica mediante FILE_WRITE | O_TRUNC, donde lo que hace la función es abrir el fichero indicado pero habiendo borrado el contenido, en el caso de haberlo. Esta función es utilizada para casos en los que el usuario decide editar desde la web la lista de inversores salvando los cambios para la próxima sesión. En cualquier caso, lo que devuelve la función es un objeto de tipo File, que se utiliza al llamar a las funciones encargadas de leer y escribir.

```
file.close()
```

Al terminar de realizar las acciones que se desean en el fichero en cuestión, es conveniente cerrar el objeto de tipo File llamando a esta función, y de este modo se asegura el que los datos escritos se guarden físicamente en la tarjeta SD. La función no devuelve ningún valor.

Para leer y escribir en el fichero, las funciones que proporciona esta librería son las siguientes, respectivamente:

```
file.read()
```

Donde se lee Byte a Byte el contenido del fichero, devolviendo el siguiente Byte (o carácter) en el caso de haber disponibles, o -1 si no hay más que leer.

```
file.write(buf, len)
```

Con la cual se escribe un arreglo de caracteres o Bytes de longitud len. La función devuelve el número de Bytes escritos.

```
file.print(data)
```

Lo que hace esta función es escribir datos en el archivo, los cuales pueden ser de tipo char, byte, int, long o string.

```
#include <Ethernet.h>
```

La función de esta librería es habilitar a la placa la conexión a Internet, pudiendo funcionar como cliente que realiza conexiones salientes o como servidor que acepta conexiones entrantes. En el caso de este sistema, la función que realiza la placa es la de servidor. En la comunicación, la placa utiliza el bus SPI, donde de la misma manera que se ha explicado para la librería anterior, el pin SS de nuestra placa (Mega), pin de hardware número 53, ha de estar configurado como salida o la biblioteca no funcionará [14].

Como nuestro dispositivo va a funcionar a modo de servidor, tenemos que configurarlo como tal, y comenzamos por definir un objeto de la clase EthernetServer indicando que las peticiones que va a recibir vendrán por el puerto 80, puesto que es el puerto por defecto para HTTP:

```
EthernetServer server(80);
```

Una vez definido el objeto de la clase servidor, tenemos que inicializar la biblioteca en cuestión y realizar la configuración de red. Ambas tareas se llevan a cabo de manera conjunta al llamar a la siguiente función, donde se indica por argumento las direcciones de interés:

```
Ethernet.begin(mac, ip, dns, gateway, subnet);
```

Una vez hecho lo anterior, el servidor ya está en disposición de recibir peticiones. Comienza a escuchar conexiones una vez se llama a la función:

```
server.begin()
```

En este punto, el servidor se encuentra escuchando conexiones entrantes, pero falta por definir un objeto de la clase EthernetClient, que será el cliente cuya conexión reciba y peticiones atienda nuestro servidor. La manera de hacerlo es la siguiente:

```
EthernetClient client = server.available();
```

Además, otro objeto de la clase EthernetClient ha de ser definido, que es el objeto que nuestro sistema utiliza para comunicarse con los inversores, es decir, con los esclavos. A través de este objeto, se les indica a los inversores al rendimiento al que deben funcionar, además de solicitárseles la potencia activa que están inyectando a la red. De ahí que al objeto se le defina con el siguiente nombre:

```
EthernetClient slave;
```

Y la manera de conectarlo a los inversores es la siguiente, donde por parámetro se especifica la dirección IP y puerto del dispositivo esclavo:

```
slave.connect(ip, port);
```

```
#include <ModbusTCPMaster.h>
```

Esta librería, al igual que la siguiente, está configurada únicamente para funcionar en dispositivos de Industrial Shields, pero pueden ser utilizadas en otros dispositivos Arduino si se modifican sus ficheros. Con ella podemos tanto leer como escribir en los dispositivos Slave varios tipos de datos a través de hasta 8 funciones que se pueden utilizar sobre un objeto de este tipo (Master) [15]. Las de interés en nuestro sistema se reducen a 2, que se muestran en la siguiente figura y cuyo funcionamiento se explica a continuación:

```
writeSingleRegister(client, slave_address, address, value);
```

```
readInputRegisters(client, slave_address, address, quantity);
```

La primera es utilizada para escribir en el dispositivo esclavo, indicado por el objeto *client* de la clase EthernetClient que previamente ha tenido que ser conectado a él mediante la función explicada en la anterior librería. En el argumento se indica también la dirección CPU del esclavo, que por lo general y sobre todo en nuestro sistema es siempre 1. Por último, en *address* se indica la bobina, entrada digital, registro de retención o dirección de registro de entrada. Por lo general, esta dirección es la bobina, entrada digital, registro de retención o número de registro de entrada menos 1, es decir, que el número de registro de retención 40009 tiene la dirección 8; y en *value*, la cifra que se quiere escribir. Este último argumento admite varios tipos de datos, pero en nuestro caso, escribimos un entero sin signo de 16 bits, puesto que los valores escritos van de 0 a no más de 2000, lo que correspondería a solicitar al inversor un rendimiento del 200% de su potencia nominal. En cuanto a la dirección en la que se escribe el valor anterior, viene dada en la hoja de especificaciones de los inversores y en el caso de los que utilizamos es la 42242, por lo que en el lugar de *address* se coloca 2241.

En cuanto a la segunda, la función que realiza es la de solicitar al dispositivo esclavo la lectura de los valores que contienen un número determinado de direcciones de memoria consecutivos. La dirección en la que está el primer dato que se pretende leer se indica en *address*, y la cantidad, en *quantity*. En nuestro caso, al utilizar la función para leer el nivel de potencia activa que inyecta el inversor a la red en el momento exacto de la solicitud de lectura, la dirección que alberga el dato de interés es la 42498, por lo que se en el argumento se escribirá el valor 2497, la cantidad de direcciones leídas será una única y el dato que la función devolverá será un entero con signo de 16 bits, lo que corresponderá a la potencia en KW.

```
#include <ModbusTCPSlave.h>
```

Nuestro dispositivo, además de funcionar como Master para los inversores, funciona también como Slave para Red Eléctrica España, de manera que hace falta definir

también un objeto esclavo para que sea capaz de recibir el dato de limitación de potencia. Al definir el objeto Slave es necesario indicar el puerto que se dedica a esa conexión, y ha de ser también conocido por REE. En nuestro caso, la manera de definirlo es la siguiente:

```
ModbusTCPSlave REE(masterPort);
```

Es necesario iniciarlo y posteriormente especificar la estructura en la que se quieren guardar los datos recibidos y el número de ellos. En el caso de nuestro sistema, la estructura es un entero sin signo de 16 bits y el número de elementos es 1, ya que el dato que esperamos recibir va de 0 a no más de 50000 (correspondería a 50000 KW de limitación). El lugar de realizar lo anterior es la sección de setup, y el modo de llamar a las funciones encargadas de hacerlo se muestra a continuación:

```
REE.begin();
REE.setHoldingRegisters(limit, 1);
```

Una vez hecho esto, será ya en la sección loop en la que se actualizará continuamente el objeto esclavo para poder comprobar si se ha escrito una nueva limitación de potencia por REE o continúa tomándose como válida la última escrita. La actualización tiene lugar al llamar a la siguiente función:

```
REE.update();
```

3.1. Elaboración de las páginas web

HTML son las siglas de “HyperText Markup Language”, y como su nombre indica, es un lenguaje de marcado de hipertexto que interpreta el navegador web para brindar al usuario el diseño de sitios y aplicaciones web al que estamos acostumbrados. La utilidad de este lenguaje es la de describir la semántica de un documento, y el modo en el que este está estructurado. La característica principal de HTML es que es un lenguaje de etiquetas, a través de las cuales el programador da formato al documento que pretende crear y distribuye los contenidos multimedia incluidos en el documento [16].

Para la programación en HTML solo es necesario un editor de texto básico como puede ser el bloc de notas, que es el que se ha utilizado en este trabajo. Para la elaboración una a una de las páginas web de las que dispone el sistema ha sido necesaria la iniciación a este lenguaje de manera autónoma y empezando desde cero, puesto que se trata de un lenguaje que no se ha visto a lo largo del grado. Para ello, fue necesario comprender la estructura básica de un documento HTML mediante la visualización de una serie de documentos sencillos que a la vez contenían etiquetas de elementos básicos de una página web.

La estructura básica de un documento HTML se muestra en la figura 7.

```
<html>


<head></head>


<body>


</body>


</html>
```

Figura 7: estructura básica de un fichero HTML.

Donde las etiquetas que indican el comienzo y el final de lo que será la página web son <html> y </html>, respectivamente. En su interior, se diferencian claramente dos estructuras. La primera es <head> </head>, en cuyo interior se escribe toda la información correspondiente a la cabecera de la página, en donde tiene cabida el título de la página, mediante <title> </title>, además de elementos como metadatos, estilos, código javascript, etc. La segunda de las estructuras es la que abarca todo el cuerpo de la página, es decir, todos los párrafos, figuras, formularios, tablas, fotografías y demás elementos que se precisen [17].

Cabe destacar en este trabajo la utilización de etiquetas como <table> </table> (que estructura el contenido que se encuentre en su interior en forma de tabla, con las especificaciones de los elementos básicos de una tabla (filas, columnas, grosor de marco, dimensiones de celda, etc) indicados en primer lugar. Otra de las etiquetas más utilizadas es la que define los formularios, <form> </form>, gracias a la cual el usuario introduce información en los campos habilitados para ello y se la hace llegar al servidor para que este la procese. Dentro de esta estructura pueden incluirse botones, campos de texto, listas, celdas de “check”, etc.

Para comprender mejor la estructura de una de las páginas del sistema se adjunta en la figura 8 el código HTML correspondiente a la que permite la gestión de los inversores del parque. En ella, aparece en las primeras líneas la cabecera típica del código de respuesta HTTP, el tipo de contenido de la respuesta y el tipo de conexión que espera mantener el servidor, respectivamente. A continuación, se introduce el texto HTML recogido entre <html> y <\html>. En su interior, se diferencia en color azul turquesa la estructura de la cabecera, en naranja la del cuerpo, de color amarillo se distinguen los formularios y de color morado aparece la estructura de una tabla, la cual también cuenta

con cabecera y cuerpo, resaltados en color verde oscuro y claro, respectivamente. Por último, en la línea en la que está indicado que se encuentra el cursor (localizada con una línea azul horizontal), iría la parte del fichero que contiene información dinámica, es decir, dependiente de variables que almacena y modifica el programa. Esta parte de código se escribiría directamente desde una función definida en el sketch.

Figura 8: fichero HTML de la página de gestión de inversores.

La manera en la que se hace llegar este texto a su destino para que el usuario web vea la página solicitada es la de escribirle el contenido del código anterior mediante la llamada a la función *write()* aplicada sobre el objeto de la clase EthernetClient que se ha creado al recibir la solicitud. Este código está almacenado en ficheros contenidos en la micro SD, de manera que son extraídos mediante la función *read()* aplicada sobre el objeto de la clase File que se ha creado al abrir el documento en modo lectura, de la manera que se explica en secciones anteriores. El contenido de estos ficheros es permanente, de manera que para mostrar en la página web de manera actualizada la información que se precisa, se vuelca la parte de código que corresponde a la estructura fija de sitio hasta la línea en la que tienen que aparecer los nuevos datos solicitados. Estos datos se escriben al usuario desde el propio código del sketch de manera explícita. Al terminar esta parte, vuelve a volcarse el resto del contenido del primer fichero, desde donde se había quedado el cursor, que termina de estructurar la parte fija de la página.

El proceso que sigue el sistema para escribir el fichero en HTML al usuario se entiende de manera ilustrativa en el diagrama de flujo de la figura 9, donde la parte estática de la

página es la parte que no varía, y la dinámica es la que incluye contenido de variables que almacena el sistema.

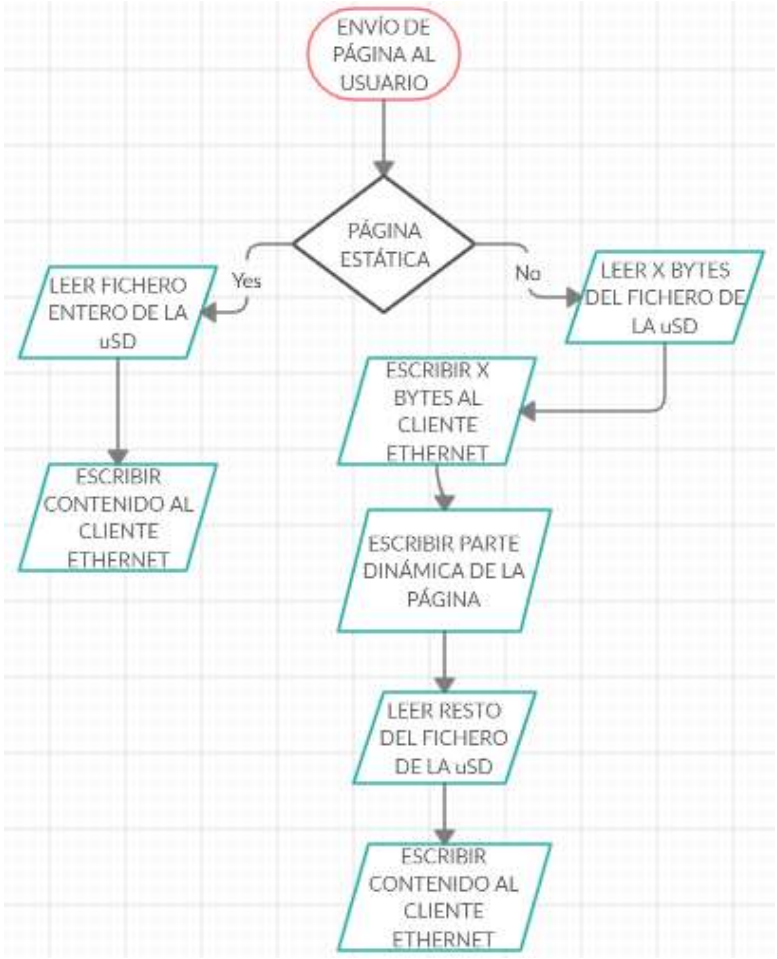


Figura 9: diagrama de flujo para mandar la página al usuario web.

Al escribir el código HTML en el bloc de notas, la única manera de comprobar que realmente el código escrito describía estructura de página deseada era comprobarlo desde un programa que actuaba a modo de servidor web programado en un sketch diferente al del sistema, que mostraba únicamente la página cuya estructura quería comprobarse. Así era posible intuir donde estaba el fallo y corregirlo. Cabe destacar que la elaboración de las páginas siguiendo este método ha resultado muy costosa, haciendo que conseguir la estructura deseada en todas ellas ha sido la parte que más ha costado del trabajo, siendo 14 el número total de ficheros que han sido creados para cubrir el servicio web de la manera deseada.

3.1. Protocolo Modbus TCP

A día de hoy, el protocolo Modbus es el más utilizado en entornos industriales para la comunicación que tiene lugar entre los dispositivos. Modbus TCP es un protocolo de comunicación destinado a la supervisión y control de equipamiento automatizado. Uno de los usos más comunes que se le da a este protocolo es el que nos atañe, que consiste en proporcionar a los PLCs conexión Ethernet, y en nuestro caso, permitir la comunicación entre este, REE y los inversores. Modbus actúa a nivel de aplicación, lo que le hace estar en la capa más cercana al usuario en el modelo OSI. En la figura 10 se muestran los niveles y los elementos del modelo.

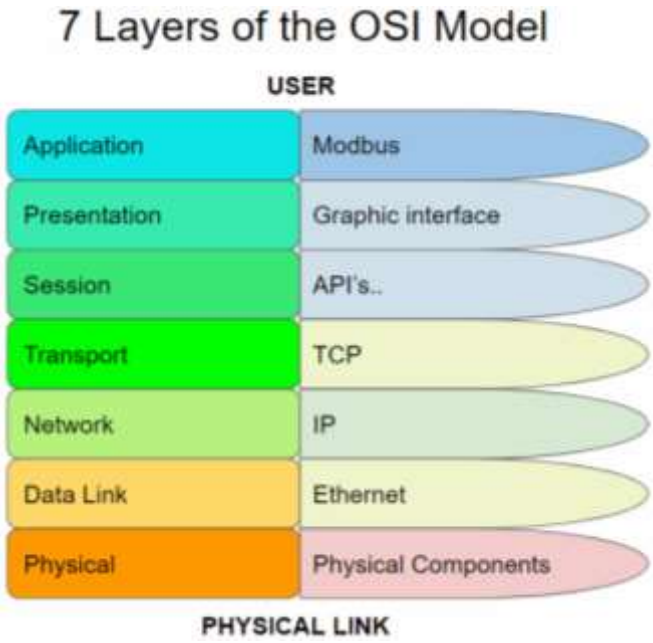


Figura 10: estructura de capas del modelo OSI [18].

La comunicación mediante este protocolo utiliza la técnica maestro-esclavo, en donde solo el maestro tiene la capacidad para iniciar una comunicación bien sea de lectura o de escritura. Una vez iniciada, los esclavos responden proporcionando al maestro los datos solicitados o adoptando las medidas indicadas por él. Los equipos que actúan como servidor suelen se aquellos equipos host que ejecutan software de una aplicación determinada, y tienen la capacidad tanto de establecer una comunicación de manera individual (con un solo esclavo) como de enviar mensajes de difusión (recibidos por todos los esclavos de la red). Por otro lado, los dispositivos esclavos se caracterizan por ser periféricos con capacidad para procesar información y enviarla al maestro. En la figura 11 se muestra claramente el tipo de mensajes propios de cada uno de los dos protagonistas de este tipo de comunicación.

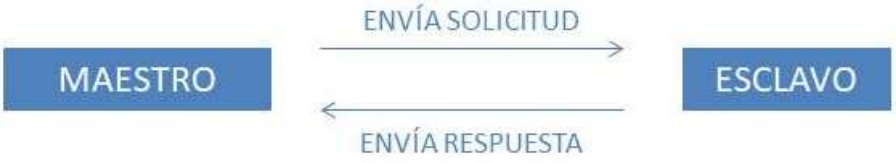


Figura 11: modo de comunicación en dispositivos vía Modbus.

Es interesante destacar el buen funcionamiento de este protocolo de comunicación frente al ruido. Además, las operaciones de programación esperan un enfoque orientado a conexión, esto hace que el protocolo utilizado sea TCP/IP ya que permite que cada una de las transacciones pueda ser fácilmente identificada y gestionada de acuerdo a la situación en la que se encuentre en cada momento, sin requerir una acción determinada por parte de las aplicaciones de maestro y esclavo. Esto hace que se adapte mejor a los cambios en el funcionamiento de la red [19].

La estructura de la trama Modbus TCP tiene la forma que se muestra en la figura 12. Como todas las tramas TCP, el primer campo es el de identificación de la transacción, utilizado para la sincronización entre mensajes de servidor y cliente, el cual dispone de una longitud de 2 bytes. Seguido va el campo de identificación del protocolo, con una longitud de 2 bytes, donde para indicar que el protocolo es Modbus TCP se localiza un 0. El tercer campo es el que contiene la longitud total de la trama expresada en número de bytes, con una longitud de 2 bytes. A continuación es donde va contenida la trama Modbus, con la estructura básica de identificador de usuario, código de función y bytes de datos. Por último, como en todas las tramas TCP, un campo de checksum dedicado a la detección de errores, como se muestra en la figura 12 [20].

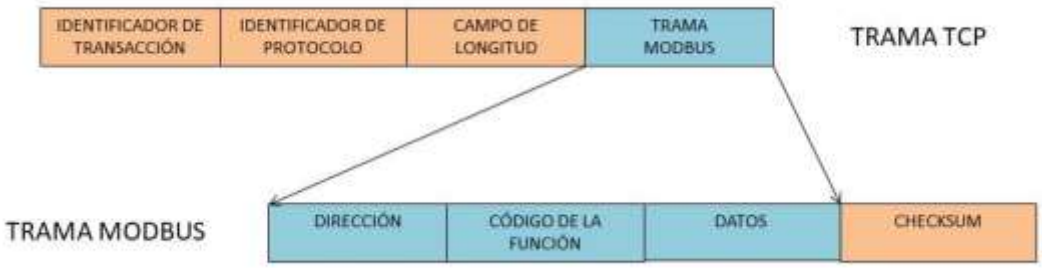


Figura 12: estructura de las tramas TCP y Modbus.

Con este protocolo, el maestro tiene acceso en el extremo del usuario a diferentes tipos de datos distribuidos en diferentes bloques. Entre ellos se diferencian las bobinas, cuyo tipo de dato que albergan en sus direcciones de memoria es boolean, al cual el maestro tiene capacidad de acceso tanto a lectura como a escritura; otro bloque recibe el nombre

de registros de retención, que maneja datos de 16 bits a los cuales el maestro también dispone de acceso tanto a lectura como a escritura; uno de los bloques a los que el maestro solo tiene acceso de lectura es el bloque de las entradas discretas, donde los datos que se manejan son de tipo boolean; y por último, también con permiso de acceso sólo a lectura, están los registros de entrada, los cuales manejan datos de 16 bits [21]. Para completar la explicación de manera más visual, se muestran los bloques y sus características en la tabla 1:

TIPO DE OBJETO	ACCESO	TAMAÑO
Coil	Lectura / Escritura	1 bit
Discrete input	Lectura	1 bit
Input register	Lectura	16 bits
Holding registers	Lectura / Escritura	16 bits

Tabla 1: tabla de bloques de datos que se pueden tratar con Modbus.

El modo en el que indica el tipo de acceso que va a hacer a un bloque determinado lo hace a través de una serie de funciones que se especifican cada una de ellas con un código. En la tabla 2 se muestran algunas de las correspondencias entre las funciones más utilizadas por los maestros y los códigos con los que se identifican.

Código	Descripción
1	Leer Bobinas
2	Leer Entradas Discretas
4	Leer Registros de Entrada
5	Escribir a Bobina Individual
6	Escribir a Registro Individual
7	Leer Estado de Excepción (únicamente serial)
3	Leer Múltiples Registros
16	Escribir a Múltiples Registros

15	Escribir a Múltiples Bobinas
20	Leer Registro de Archivo
21	Escribir a Registro de Archivo
22	Escribir a Registro con Máscara
23	Leer/Escribir Múltiples Registros
24	Leer FIFO

Tabla 2: correspondencia entre identificador y funciones para comunicación Modbus.

Además, los esclavos también utilizan excepciones identificadas con códigos para indicar, en caso de haberlo, el tipo de error en el mensaje recibido. Dentro del estándar, hay 4 códigos de excepción que son los más utilizados. La correspondencia se muestra en la tabla 3.

Código de Excepción	Significado
01	El código de función recibido no está soportado. Para confirmar el código de función original, restar 0x80 del valor devuelto.
02	La solicitud intentó tener acceso a una dirección no válida. En el estándar, esto puede ocurrir únicamente si la dirección de inicio y el número solicitado de valores excede 2^{16}. Sin embargo, algunos dispositivos pueden limitar este espacio de dirección en su modelo de datos.
03	La solicitud tenía datos incorrectos. En algunos casos, esto significa que había una discrepancia de parámetros, por ejemplo entre el número de registros enviados y el campo "cantidad de bytes". Normalmente, el maestro solicitó más datos de lo que permite ya sea el esclavo o el protocolo. Por ejemplo, un maestro puede leer solamente 125 registros de detención a la vez y los dispositivos con recursos limitados pueden delimitar este valor a incluso menos registros.
04	Se ha producido un error irrecuperable al intentar procesar la solicitud. Este es un código de excepción general que indica que la solicitud era válida, pero el esclavo no podía ejecutarla.

Tabla 3: correspondencia entre identificador y excepciones en comunicación Modbus.

CAPÍTULO 4

TEST Y VERIFICACIÓN

4.1. Simulador de Red Eléctrica España

REE, como ya se ha explicado, funciona a modo de maestro, por lo que el sistema necesario debe funcionar como tal. En este caso, el software ha sido proporcionado por la empresa, al igual que el empleado para el caso de los inversores con comportamiento de esclavo que comentaremos en el siguiente apartado.

Modbus Poll es un simulador de especial ayuda para desarrolladores de dispositivos esclavo, como es nuestro caso, para testear el correcto funcionamiento del protocolo Modbus en ellos [22]. El servicio que ofrece al usuario es el de poder realizar tanto tareas de lectura como de escritura en diferentes esclavos Modbus de manera simultánea. Para ello permite abrir varias ventanas, cada una de las cuales ha de ser configurada de acuerdo al ID del esclavo Modbus, el código de la función que se desea que el maestro realice sobre el esclavo, la dirección de memoria, el tamaño y la tasa de sondeo. Además, cada ventana tiene una estructura de memoria con el tamaño especificado de direcciones dentro de las cuales se pueden cambiar los registros mediante un doble clic en la celda correspondiente. En nuestro caso solo precisamos de un esclavo Modbus, por lo que con abrir una ventana es suficiente. En la figura 13 se muestra el ajuste que correspondería a realizar el ejercicio de lectura de registros de entrada de 10 (*quantity*) direcciones de memoria empezando en la dirección 0 (*address*), con una frecuencia de una solicitud por segundo (*Scan Rate*).

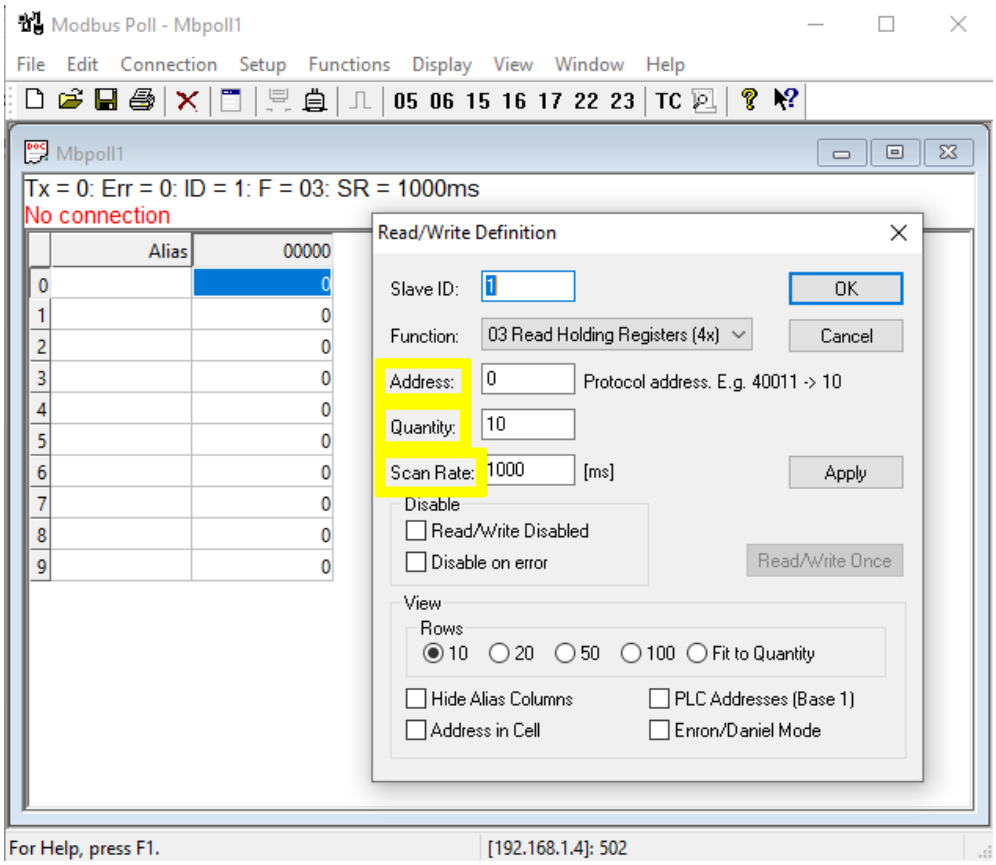


Figura 13: ventana de ajuste del Master en Modbus Poll.

De todas las funciones Modbus que hemos visto que existen, en este software concretamente pueden usarse las siguientes:

- 01: leer el estado de las salidas digitales
- 02: leer el estado de las entradas digitales
- 03: Leer registros de retención
- 04: Leer registros de entrada
- 05: Forzar salida digital simple
- 06: Forzar registro de retención simple
- 15: Forzar múltiples salidas digitales
- 16: Forzar múltiples registros de retención
- 17: Informar ID de esclavo
- 22: registro de escritura de máscara
- 23: registros de lectura / escritura

De las cuales para nuestro sistema únicamente precisamos de la utilización de la 06 - Escritura de registro simple (recordemos que el dato que queremos escribir es un entero positivo no superior a 50000).

Tras definir los datos de la ventana correspondiente al esclavo, es momento de realizar el ajuste de conexión. En esta ventana tiene lugar la indicación del protocolo de comunicación (en nuestro caso Modbus TCP/IP), de la dirección IP del dispositivo esclavo, del puerto en el que escucha, y del ajuste del tiempo de timeout de conexión y de respuesta, principalmente. En la figura 14 se muestra la ventana en la que se ajustan los parámetros anteriores:

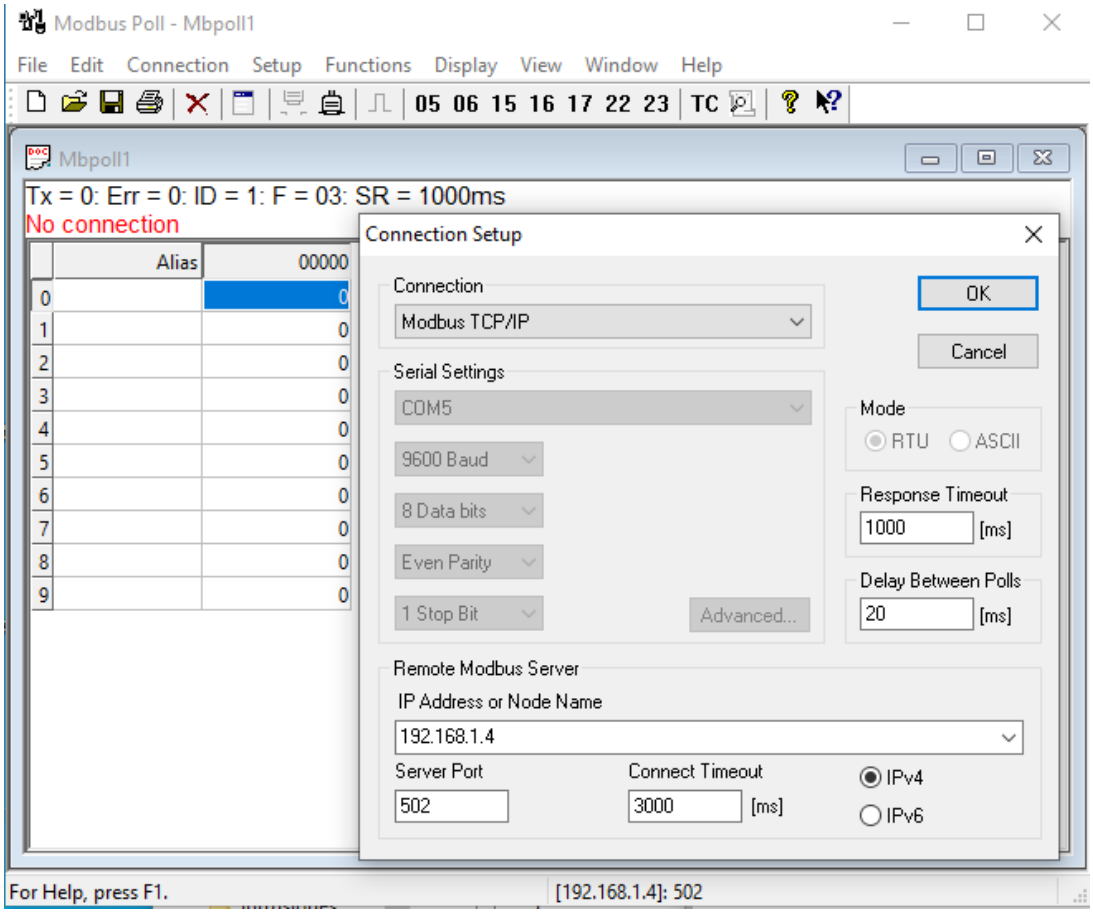


Figura 14: ventana de ajuste de la conexión.

Una vez se acepta la configuración pulsando el botón de OK, el programa comienza a intentar conectarse al dispositivo esclavo para el que se ha realizado el ajuste. En caso de haber errores, la variable *Err* de la parte superior de la ventana va incrementándose y se muestran en la línea de estado.

4.2. Simulador de inversores

Es el software con el que se han simulado los inversores. En el paquete de descarga vienen ambos programas, tanto el de naturaleza de servidor y el de cliente, pero en nuestro caso solo se ha considerado el uso del servidor. Es un software muy sencillo que permite al usuario utilizarlo desde el primer momento, mostrando una apariencia nada más abrir la aplicación de lo más intuitiva.

En la figura 15 puede verse el aspecto de la pantalla principal, en donde arriba a la derecha se abre una ventana de ajustes haciendo clic en la pestaña de *Setup* donde configurar el puerto en el que el esclavo está escuchando (por defecto viene dado el puerto 502).

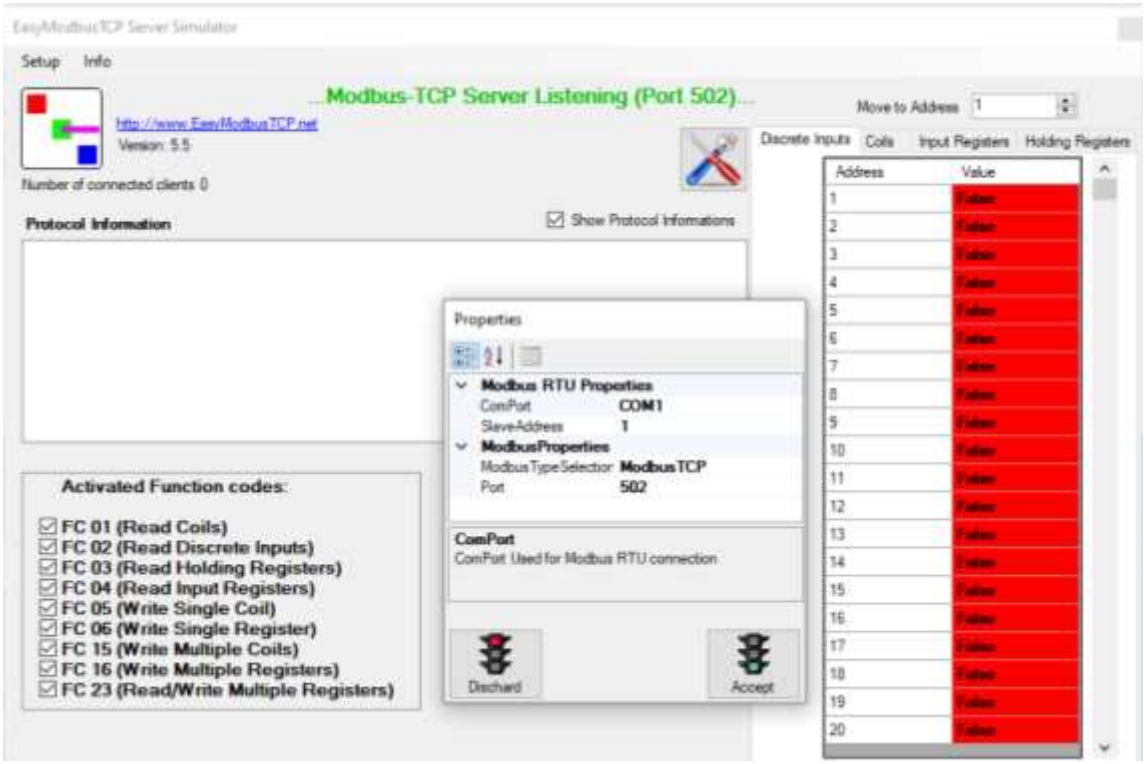


Figura 15: ventana de propiedades del Slave en EasyModbusTCP Server.

Abajo a la izquierda se muestran las funciones que son soportadas por la aplicación dentro de todas las comentadas anteriormente, donde se puede seleccionar cuales de ellas va a utilizarse en el ejercicio. Por otro lado, arriba a la izquierda indica el número de usuarios maestro están conectados en cada momento. A la derecha se muestra de manera muy ilustrativa el abanico de bloques de datos cuyas direcciones de memoria se quiere monitorizar, pudiendo elegir la primera dirección de la lista en el campo *Move to Address* que se encuentra inmediatamente en la parte superior. Por último, en la parte central se encuentra la ventana que se abre al clicar sobre *Setup*, pudiendo ajustar los parámetros correspondientes en función del tipo de comunicación Modbus que se

pretende emplear, soportando tanto Modbus RTU como Modbus TCP. En nuestro caso, la utilizada es la última de las dos anteriores.

CAPÍTULO 5

MODO DE FUNCIONAMIENTO

En este capítulo se explica de manera conceptual el funcionamiento del código implementado para el sistema, apoyando el texto con una figura de tipo diagrama de bloques para que sea más ilustrativa la comprensión del flujo de datos y de las operaciones realizadas en cada momento.

Como hemos explicado en capítulos anteriores, el sketch se divide en dos partes. Al ejecutar el código y poner en marcha el sistema, tiene lugar el proceso de *setup*, como en cualquier dispositivo. En él tiene lugar, primeramente, la inicialización del puerto Serial, de la tarjeta SD, de la extracción de la memoria EEPROM las direcciones de red del sistema predeterminadas, la conexión del sistema a la red, la inicialización del maestro esclavo Modbus TCP y la espera a recibir un dato de limitación de potencia para que la planta pueda comenzar a producir. Una vez recibido el dato, se procede a la carga de toda la información correspondiente a los inversores registrados y guardados desde la sesión anterior y el cálculo y escritura del factor en los inversores, el cual les indica el rendimiento respecto de su potencia nominal al cual debe funcionar cada uno para producir el máximo permitido por el dato de REE. En el diagrama de flujo de la figura 16 se grafica toda esta secuencia, que ocurre siguiendo el orden en que se ha redactado:

Figura 16: diagrama de flujo para la parte de código de Setup del sistema.

Una vez tiene lugar el proceso de *setup*, se procede a la ejecución del código encerrado en el bloque estructural llamado *loop,* que como se ha comentado, se ejecuta de manera repetida. Lo primero que hace el sistema en cada ciclo del código es actualizar la variable que recoge el dato que escribe REE en la dirección de memoria indicada, que en nuestro caso, es la dirección 0. Inmediatamente seguido de esto, lo que hace es escuchar si hay solicitudes del usuario web a la espera de ser atendidas. En el supuesto de que así sea, se pasa la solicitud HTTP a una función para que esta devuelva solamente la parte de la solicitud que realmente contiene información sobre el interés del usuario. El fragmento de la solicitud devuelto por la función anterior se pasa como argumento a otra, que se encarga realmente de averiguar de cual de entre todas las posibles opciones que está el sistema capacitado para servir se trata la presente, y realizar las acciones pertinentes o gestionar de una manera determinada los elementos que se precise. Esta parte se explica con más detenimiento teniendo en cuenta las posibles solicitudes que el sistema puede recibir con las correspondientes acciones que lleva a cabo. Una vez hecho lo anterior, es momento de volcar el fichero que contiene el código en HTML al usuario para que se muestre servida la acción solicitada en el diseño propio de la página. Una vez servido al usuario web, o suponiendo el caso en el que no lo hubiese habido, el sistema comprueba cuánto tiempo ha pasado desde la última vez que comprobó tanto el funcionamiento de los inversores como si se había recibido una nueva limitación de potencia. De haber pasado un tiempo determinado (en nuestro caso,

5 segundos) una función se encarga de solicitar el dato de potencia activa a cada uno de los inversores disponibles. Para cada uno de los inversores, comprueba que el dato de potencia activa actual se corresponda con lo esperado de acuerdo a su potencia nominal y al factor de rendimiento que se le escribió con anterioridad. En el caso de que para alguno de los inversores no se corresponda, será un indicador de que el inversor tiene un comportamiento anómalo y que tiene que ser sometido a una inspección técnica para arreglar los desperfectos. La medida que el sistema toma es la de dejar de contar con él para ejercicios de producción posteriores y escribirle un 0 en la dirección de memoria en la que se le indica el factor de rendimiento al que tiene que funcionar (lo que corresponde a pedirle que produzca 0 KW de potencia activa). La manera en la que el dispositivo volverá a funcionar será configurándolo nuevamente desde la página web como si se tratase de la primera vez que se añade a la red. Posterior a esta parte de detección de anomalías, tiene lugar la comprobación de si el dato de limitación de potencia actual difiere del anterior, y actualizarlo. En este momento, calcula el factor que se tiene que escribir en los inversores teniendo solo en cuenta aquellos para los que se haya comprobado que están disponibles y funcionan de manera correcta, se escribe el factor a través de la llamada a una función y se vuelve al comienzo del ciclo. Los inversores pueden funcionar incluso a rendimientos mayores del 100%, pero hay que conocer el límite para el cual el aumento deja de ser lineal y puede dañarse. Este límite es el 125% de su rendimiento, de manera que el factor que se le escriba no será en ninguno de los casos superior a 1250.

Este proceso se puede seguir mejor con la ayuda de la figura 17.

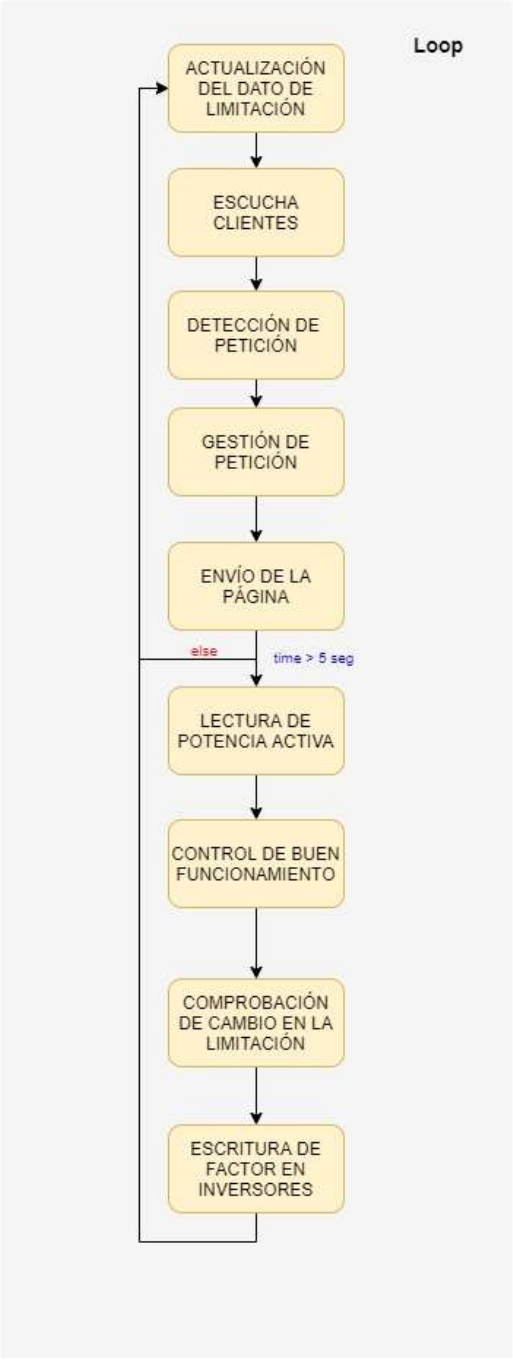


Figura 17: diagrama de flujo para la parte de código repetitiva del sistema.

Para entender mejor cómo funciona la gestión de la solicitud, se ha elaborado este apartado en el que se explica más detalladamente lo que ocurre en la función que se encarga de realizar la tarea en cuestión.

La solicitud que contiene la información que la función necesita, como hemos comentado, se le pasa por parámetro. Esta solicitud puede contener la palabra "username", lo que indicaría que se está recibiendo una solicitud de inicio de sesión. De ser así, la solicitud lleva el nombre de usuario y la contraseña en unas posiciones que el

sistema conoce, por lo que extrae ambos datos y comprueba si coinciden con los que guarda como válidos. De ser así, la respuesta que recibiría sería la página de inicio, en la que puede elegir acceder a diferentes secciones. De no coincidir, la página que recibirá será la misma de inicio de sesión pero con la indicación de que los datos introducidos son incorrectos.

En caso de contener la palabra "system", el sistema interpreta que se está solicitando acceso a la página en donde se gestionan las direcciones de red propias del sistema, por lo que volcará el fichero correspondiente. De contener las palabras "dnssys", "ipsys", "masksys" o "gatewaysys", se estará solicitando que se tomen como direcciones de la sesión actual, por lo que tras someterse a prueba y en el caso de que sean direcciones válidas, así se hará. Si no pasan la prueba, se mostrará la misma página pero con el mensaje de error correspondiente a la dirección que lo contiene. Si las palabras contenidas son "reset" o "reconnect", se estará solicitando que se tomen como direcciones del sistema las que venían por defecto al inicio del mismo, o que se reestablezca la conexión con las direcciones que en ese momento el sistema considere como propias, respectivamente. En el caso de contener "EEPROM", lo solicitado será que las direcciones que en ese momento considera el sistema como propias sean guardadas en memoria EEPROM para que sean las que se tomen por defecto al inicio de sesiones futuras.

Si la palabra que contiene es "manage", se entiende que el acceso solicitado es a la página en la que se gestionan los inversores. Si es "dir", se estará solicitando añadir un nuevo inversor, de manera que si todavía no se ha alcanzado el número máximo de inversores que soporta el sistema (25, limitado por el espacio disponible para ello en memoria dinámica), se comprueba la validez de todos los parámetros que han de ser introducidos por el usuario para poder añadir un inversor y de ser correctos, se añade a la lista y se muestra la página con la tabla actualizada. De ser erróneo el dato introducido en alguno de los campos, la página se muestra con el mensaje de error especificando la información que ha de ir en cada uno de ellos. Si por el contrario se encuentra "retry", el usuario pretenderá reintentar la conexión con todos los inversores que aparezcan en la tabla, y de encontrarse "rem", se estará solicitando eliminar de la lista alguno de los inversores. Para cada una de las opciones, se vuelca el fichero con la página de gestión de inversores con las particularidades que precise la solicitud. Otra opción puede ser que aparezca "add", y el significado será que se está solicitando la edición de los ficheros contenidos en la memoria SD que almacenan la información de los nombres, las direcciones IP, los puertos y las potencias nominales de los inversores, donde pasarán a escribirse nuevamente con los datos que en ese momento se muestren en la tabla de la página en cuestión.

Puede contenerse la palabra "view", lo cual querrá decir que el usuario intenta acceder desde la página de inicio a la página en la que se visualiza la potencia de los inversores, total del parque y la limitación en tiempo real. Puede darse el caso en que la palabra encontrada sea "refresh" o "aplicar", las cuales significarán que se desea que la página se refresque cada 5 segundos con la información actualizada (predeterminado) o que se

pretende realizar cambios con respecto al refresco de la página, respectivamente. En cada caso, se escribirá el fichero de la página de visualización de los inversores en tiempo real con las especificaciones de la solicitud.

Además, en todas las páginas se muestra en la parte superior izquierda un botón en el que pone “BACK TO HOMEPAGE”, el cual devuelve al usuario a la página de inicio.

En la figura 18 puede entenderse de manera conceptual el procedimiento que utiliza el sistema.

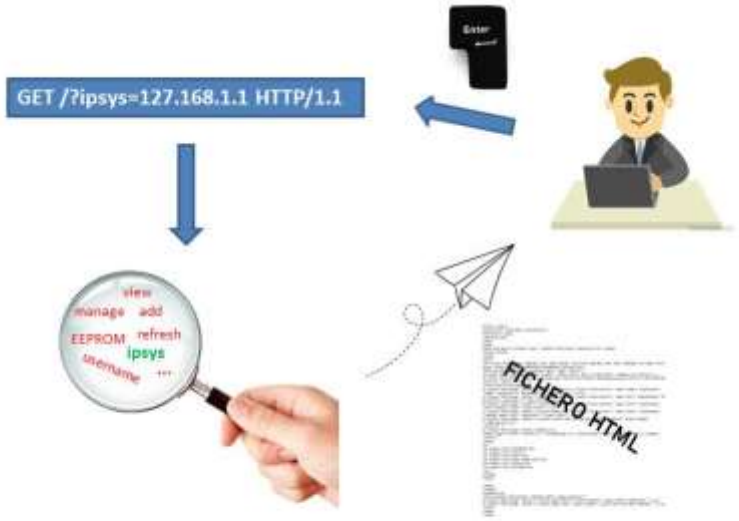


Figura 18: esquema del servicio de peticiones web.

CAPÍTULO 6

DISEÑO WEB

El sistema consta de un total de 6 páginas web, diseñadas todas ellas utilizando el mismo formato. A continuación, se muestra una imagen de cada una de ellas comentando los aspectos más importantes.

6.1. Inicio de sesión

En la página de inicio de sesión, aparece en la parte superior a modo de título información del parque fotovoltaico al que se accede al introducir los datos que lo habilitan. En este caso, como el parque que desde un principio se habló que sería el de testeo del sistema era el que está situado en Alfaro, se ha dado el nombre correspondiente al sistema que estaría en este centro. En el diseño de las celdas en donde se introducen los datos de inicio de sesión se ha considerado utilizar el formato de texto para el nombre de usuario y el de puntos propio de los campos de contraseña. Al pulsar el botón de "Login" se accede a la página principal del sistema. En la figura 19 se muestra la página de inicio de sesión.

Figura 19: página de inicio de sesión.

6.2. Página principal

Esta página tiene un diseño sencillo. Nuevamente, a modo de título se indica que el usuario está en la página principal. En ella, se indica en un texto que el usuario puede elegir una de las 3 opciones para dirigirse al departamento que desee. Las 3 opciones ofrecidas se muestran a modo de botón, en cuyo interior aparecen los nombres de cada una de las secciones. La página de inicio se muestra en la figura 20.

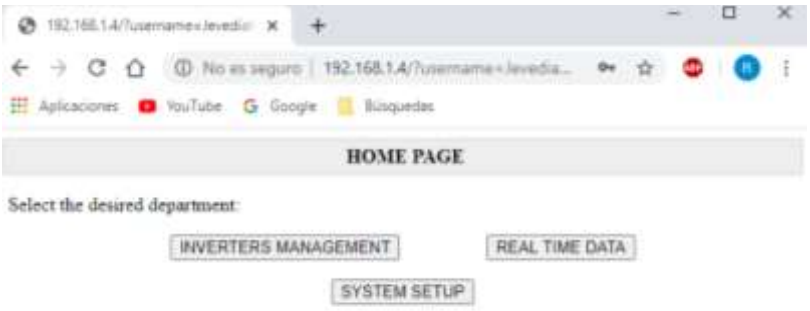


Figura 20: página de inicio.

6.3. Ajuste de direcciones del sistema

En caso de seleccionar la opción de “SYSTEM SETUP”, el usuario aparece en una página en la que encuentra los datos contenidos en una tabla. El total es de 4 filas y cada una de ellas contiene la información sobre el tipo de dirección de red, un cuadro de texto para introducir la dirección que se desee en el cual se muestra la actual mientras no se haga clic sobre el campo en cuestión, y un botón que es el que tras presionarse enviará la información contenida en la celda al sistema para que se encargue de gestionarla. Además, en la parte inferior de la página se encuentran 3 botones, en cuyos interiores pone “RESET”, “RECONNECT” Y “SAVE TO THE NEXT SESSION”. El primero vuelve a cargar como direcciones del sistema las que están contenidas en memoria EEPROM, que es con las que se ha establecido la conexión al iniciarse el sistema. El segundo de ellos es el encargado de que el sistema reestablezca la conexión con las direcciones que en ese momento tenga como propias. El último botón es el encargado de que las direcciones que en ese momento se muestren en la tabla sean guardadas en memoria EEPROM para que a partir de la siguiente sesión sean las predeterminadas, con las que se establezca la conexión al iniciarse el sistema. En la figura 21 aparece la página de ajuste de direcciones del sistema.

Figura 21: página de direcciones red del sistema.

6.4. Gestión de inversores utilizados

Otra de las opciones que ofrece la página principal es la de acceder a la de gestión de inversores. En esta página, aparecen campos para rellenar con la información del inversor que se pretende añadir. En primer lugar, una celda en la que se escribirá el nombre de inversor, o la información con la que se pretende identificar. Este campo admite cadenas de caracteres hasta un máximo de 30. Seguido está el campo en el que el usuario debe introducir la dirección IP del inversor. Los últimos dos campos son el del puerto en el que el inversor escucha, el cual si no se especifica se asocia por defecto el 502, y el campo en el que se introduce la potencia nominal del inversor en KW. Debajo de esta sucesión de campos, aparece una tabla en la que se muestra toda la información de los inversores registrados incluyendo el estado de conexión de cada uno de ellos, indicando en verde los que están conectados y en rojo los que no. Esta conexión se puede comprobar todas las veces que se desee pulsando el botón en cuyo interior pone “RETRY CONNECTION”. Debajo de éste botón se encuentra otro que ofrece la posibilidad de salvar la lista en memoria SD para que en las próximas sesiones estén ya ajustados desde el inicio. Esta página se muestra en la figura 22.

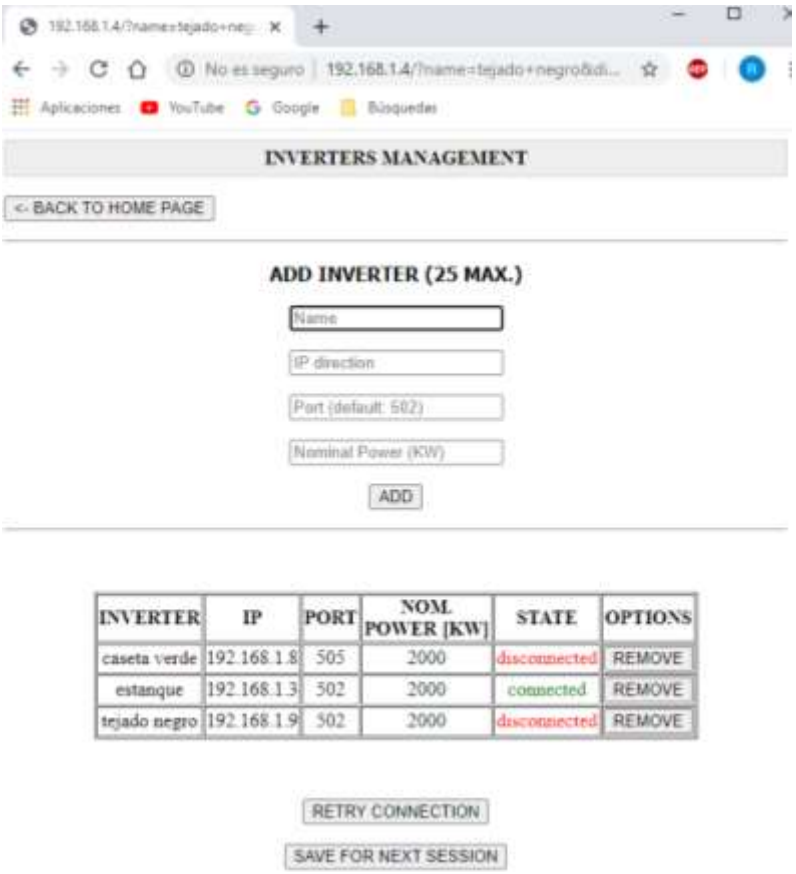
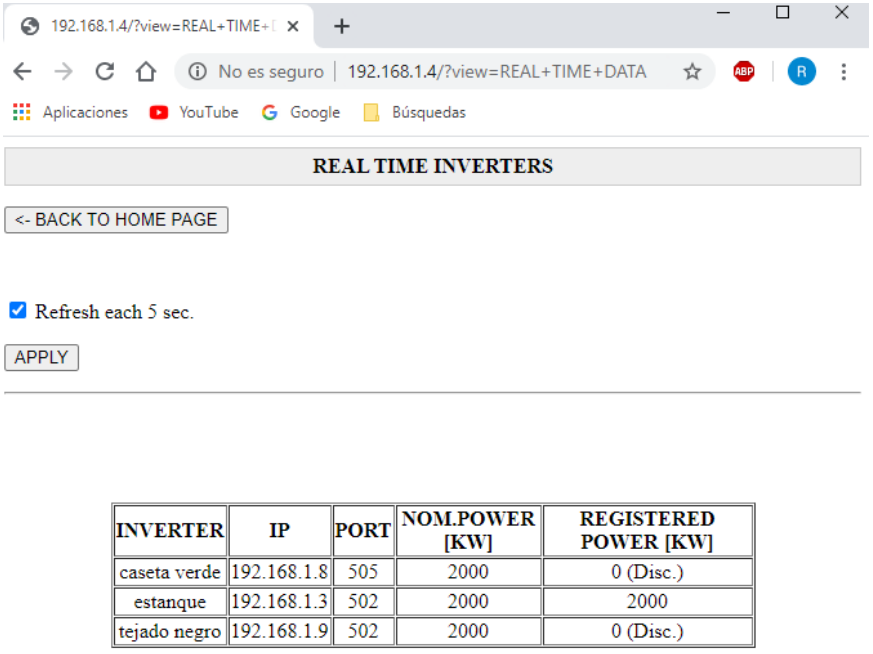


Figura 22: página de gestión de inversores.

6.5. Visualización de datos en tiempo real

Por último, está la página en la que se monitoriza la actividad de los inversores, visualizando en una tabla los valores de potencia activa que está produciendo cada uno de manera individual. Además, en la tabla también aparece si alguno está desconectado o ha sido capado por comportamiento anómalo. Debajo de la tabla, se observan los valores en tiempo real de limitación de potencia y de potencia total que está siendo producida por el parque. En la parte superior izquierda, mediante un botón de tipo "check", el usuario puede elegir si desea que la página se refresque cada 5 segundos de manera automática con los datos actualizados o si prefiere que no. En la figura 23 aparece la página en la que se visualizan los datos del parque en tiempo real.



INVERTER	IP	PORT	NOM.POWER [KW]	REGISTERED POWER [KW]
caseta verde	192.168.1.8	505	2000	0 (Disc.)
estanque	192.168.1.3	502	2000	2000
tejado negro	192.168.1.9	502	2000	0 (Disc.)

CURRENT POWER LIMITATION: 2000 KW

CURRENT POWER GENERATED: 2000 KW

Figura 23: página de visualización de datos entiempo real.

CAPÍTULO 7

PRUEBAS DE ROBUSTEZ DEL SISTEMA

Conseguir el correcto funcionamiento del sistema siempre es el primer gran paso, pero a menudo no es suficiente solo con eso. El servidor web debe estar en funcionamiento en todo momento y una manera de que falle podría ser introducir datos en un formato que él no espera recibir en un campo concreto. Así pues, se han tenido en cuenta una serie de casos en los que el sistema podría dejar de funcionar para prepararlo con el fin de asegurar una continuidad y una gestión correcta de cara al usuario.

7.1. Contenido erróneo en campos

El usuario web se encuentra con campos de texto en los cuales debe introducir datos, pero el formato de ellos no es siempre el mismo. En unos campos debe introducir palabras, en otros, direcciones de red, y también hay campos en los que debe introducir números enteros, pero incluso en este último caso hay restricciones. Por ejemplo, cada una de las 4 cifras que forman una dirección IP no pueden ser números negativos ni superiores a 255. Otra de las limitaciones está en el puerto que se asigna en los inversores, el cual en ningún caso será inferior a 0 o superior a 65535. El formato en el que deben rellenarse cada uno de los campos se especifica en el mensaje de error que aparece en color rojo en las figuras 24 y 25, correspondientes a las páginas de gestión de inversores y de ajuste de direcciones de red del sistema, respectivamente.

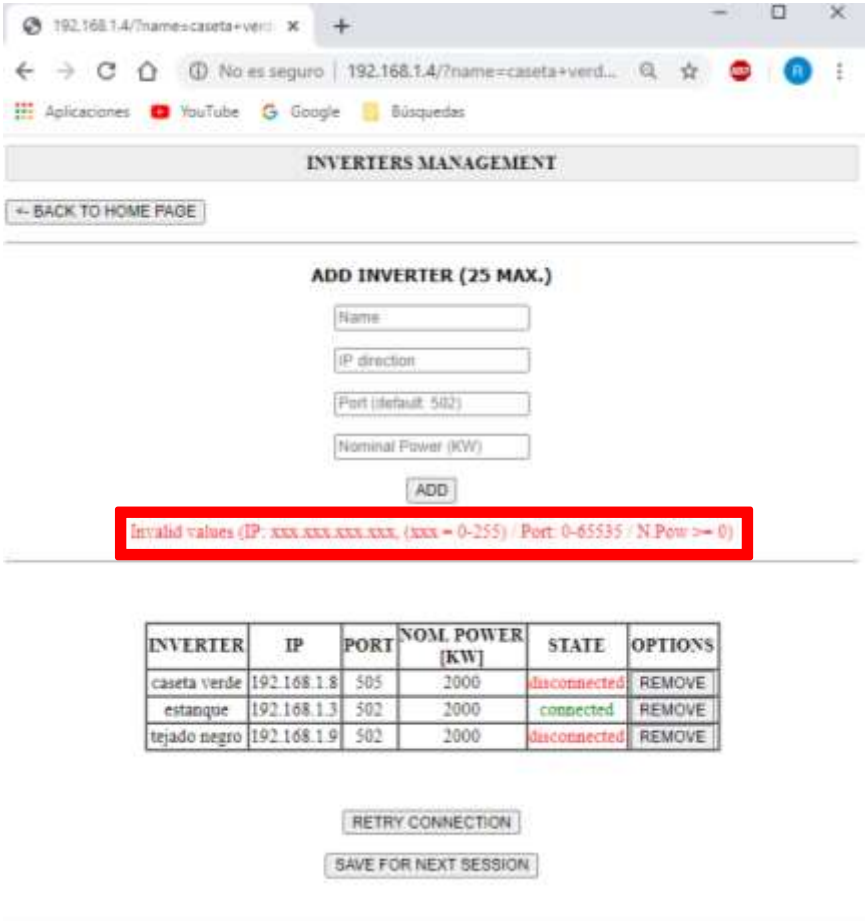


Figura 24: mensaje de error al intentar registrar un inversor.

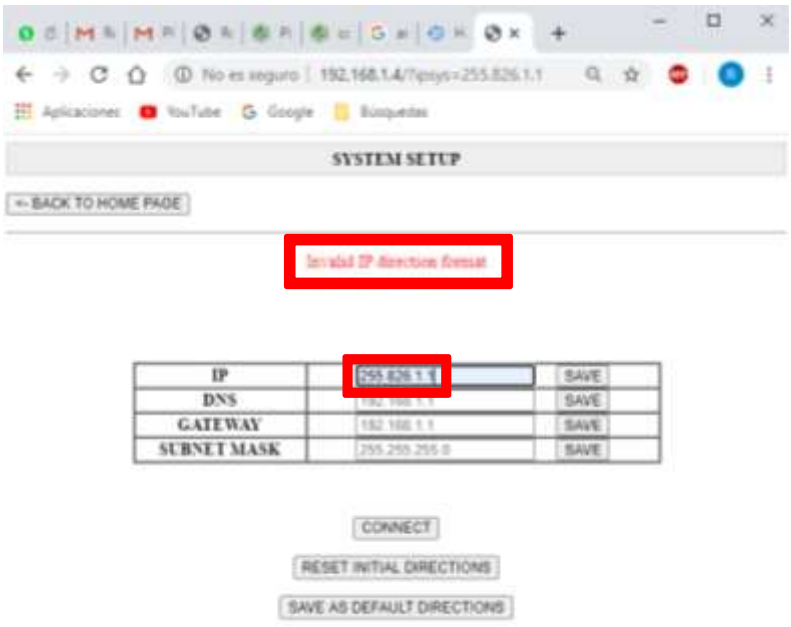


Figura 25: mensaje de error al introducir una dirección IP errónea.

7.2. Caída de inversores

Otro de los casos en los que el sistema puede colapsarse por encontrarse frente a una situación que no espera sería la caída de un inversor que lo supone en correcto funcionamiento. La manera de operar correctamente frente a este supuesto sería repartir, con un incremento de porcentaje de rendimiento, la producción de potencia activa que el inversor caído deja de aportar. Para ello, el sistema debe realizar un seguimiento continuado de la conexión de cada uno de los inversores, y de esta manera se producirá en todo momento el máximo de potencia para el que está capacitado con el número de recursos que tiene a su disposición en planta. Además, el usuario recibe información del estado de los inversores en todo momento tanto en la página de gestión de inversores (figura 26) como en la de visualización de datos en tiempo real (figura 27).

INVERTER	IP	PORT	NOM. POWER [KW]	STATE	OPTIONS
caseta verde	192.168.1.8	505	2000	disconnected	REMOVE
estanque	192.168.1.3	502	2000	connected	REMOVE
tejado negro	192.168.1.9	502	2000	disconnected	REMOVE

Figura 26: estado de conexión de los inversores en la página de gestión.

INVERTER	IP	PORT	NOM.POWER [KW]	REGISTERED POWER [KW]
caseta verde	192.168.1.8	505	2000	0 (Disc.)
estanque	192.168.1.3	502	2000	2000
tejado negro	192.168.1.9	502	2000	0 (Disc.)

Figura 27: estado de conexión de los inversores en la página de tiempo real.

7.3. Comportamiento anómalo de inversores

Como se ha comentado en alguno de los apartados anteriores, podría darse el caso de que un inversor inyecte más o menos potencia en la red de la que se le indica en el dato de rendimiento que se le haya escrito en memoria. Si se da el primer caso, la empresa podría ser multada, y de darse el segundo, la planta no estaría trabajando de manera eficiente. Cualquiera de las dos consecuencias son negativas para la empresa, por lo que el sistema que busca quiere que contemple ambos casos y esté previsto de soluciones si alguno de los dos se da. La manera en la que actúa el sistema es escribiéndole al inversor en cuestión un 0 en la dirección de memoria en la que se le comunica al rendimiento que debe funcionar, sacándolo de la lista en la que tiene todos los inversores disponibles conectados y mostrando al usuario web un mensaje de fallo en la columna de la producción de potencia individual de la fila correspondiente a ese inversor, como se puede ver en la figura 28. Como ya se ha comentado con anterioridad, el modo de actuar sería supervisar y corregir el fallo y volver a registrar el inversor en el sistema a través de la página de gestión de inversores.

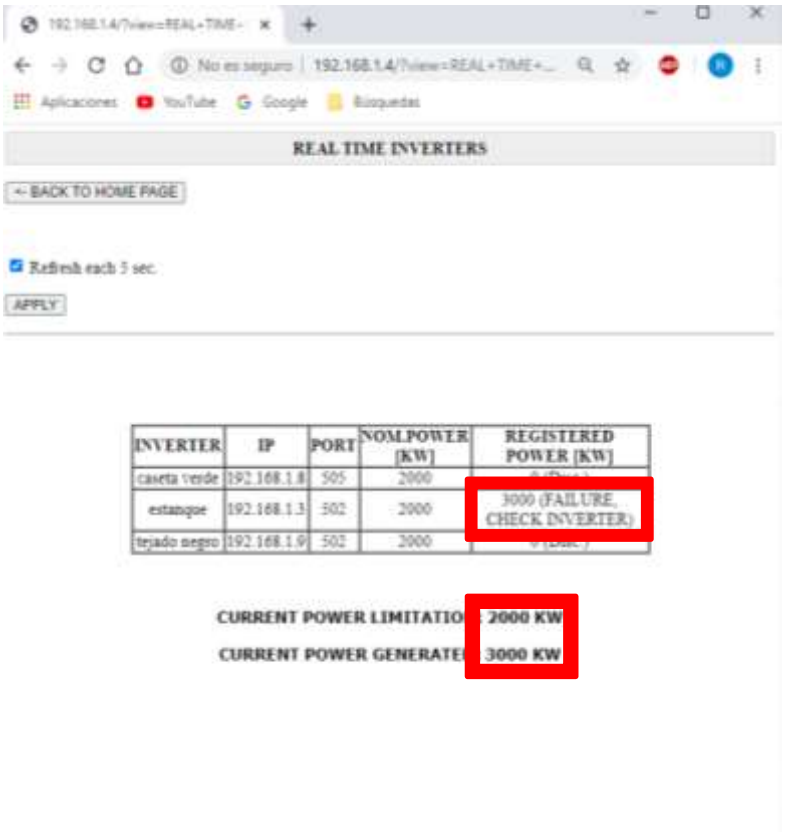


Figura 28: mensaje de comportamiento anómalo del inversor.

CAPÍTULO 8

CONCLUSIONES Y TRABAJO FUTURO

8.1. Conclusiones

Tras comprender el total de factores que había que tener en cuenta en cuestión de acatar restricciones legales de limitación de potencia, gestionar todos los parámetros de los inversores para hacer un buen uso de ellos y obtener el rendimiento que se espera, y servir al usuario a modo de servidor web para que de la manera más sencilla posible y sin requerir conocimientos de programación pueda realizar el ajuste todos los elementos que intervienen en el ejercicio del control de la producción de potencia, además de monitorizarla y conocer en todo momento el estado de la planta, se puede terminar el trabajo concluyendo que se ha realizado desde cero un sistema de control de potencia de plantas fotovoltaicas que cubre todos esos requerimientos.

Por un lado, comprueba continuamente el dato que marca REE y modifica el rendimiento de los inversores en caso de que sea necesario.

Yendo más allá de los objetivos principales, se ha conseguido detectar y corregir comportamientos anómalos de inversores tras contemplar que estamos operando con máquinas y no sería prudente esperar un comportamiento exento de fallos por su parte.

El único servicio que no se ha podido implementar ha sido el de notificar al centro de control a través del correo electrónico, puesto que ha supuesto más problemas de lo que se supuso en un principio al ser necesaria la creación de un correo corporativo, entre otros aspectos, y se ha planteado como posibles mejoras futuras para el sistema.

Por último, igual de importante que se consiguiese un control riguroso de la potencia activa del parque era que el usuario tuviese un acceso al parque a través de la web, lo cual se ha logrado diseñando un servidor web que se encargase del manejo de las peticiones del usuario y le permitiese monitorizar los datos de interés que se han explicado en secciones anteriores.

8.2. Trabajo futuro

Si cierto es que se ha implementado un sistema que cubra los requerimientos básicos iniciales, también lo es que podría trabajarse todavía más para conseguir un servicio más completo.

Los inversores trabajan con muchos más parámetros distribuidos en memoria de la misma manera que lo hacen los de potencia activa o factor de rendimiento, y podría resultar interesante gestionarlos desde el mismo autómata y acceder a ellos desde la misma web.

Además, y como se ha comentado en la sección anterior, otro de los aspectos en los que se podría trabajar sería en la comunicación de información que se considere de importancia a través de correo electrónico o SMS, para que pueda asegurarse que llegue a la persona que tenga que llegar sin necesidad de que se encuentre en ese momento visualizando los datos desde la página web.

8.3. Valoración personal

Desde el primer momento en el que se me ofreció la oportunidad de trabajar con la empresa en este proyecto sentí una motivación que me ha acompañado a lo largo de todo el trabajo.

El tema entorno al que se ha hecho este trabajo me llama la atención por la creciente importancia que está ganando día a día, el hardware y software me resultaban muy interesantes y el hecho de conocer el lenguaje de programación me ha hecho sentirme muy cómodo.

Cabe destacar sobre todo la atención por parte de la empresa Ríos Renovables en todo momento frente a cualquier tipo de duda que me podía surgir, en concreto Juan Calvo, a cargo de quien he estado en este proyecto.

El único aspecto negativo que me he encontrado en la elaboración del proyecto es que las medidas tomadas a nivel nacional en consecuencia de la crisis del COVID-19 han hecho todavía más difícil el desarrollo del trabajo puesto que no he podido asistir ni un solo día al lugar donde está situada la empresa y desde donde se pensaba llevar a cabo, y personalmente me han privado de poder tener mi primer contacto con el mundo laboral que tanta ilusión me hacía.

BIBLIOGRAFÍA

1. La energía y el cambio climático [Internet]. European Environment Agency. 2020 [citado el 26 de mayo de 2020]. Disponible en: https://www.eea.europa.eu/es/senales/senales-2017-configuracion-del-futuro/articulos/la-energia-y-el-cambio-climatico
2. ¿Hasta dónde llega la Luz Solar? | Ciencia de la NASA [Internet]. Ciencia.nasa.gov. 2002 [citado el 27 de mayo de 2020]. Disponible en: https://ciencia.nasa.gov/science-at-nasa/2002/08jan_sunshine
3. Einstein y la energía solar fotovoltaica - EADIC - Cursos y Master para Ingenieros y Arquitectos [Internet]. 2017 [citado el 28 de mayo de 2020]. Disponible en: https://www.eadic.com/einstein-y-la-energia-solar-fotovoltaica/
4. Almacenamiento energético | Red Eléctrica de España [Internet]. Ree.es. [citado el 30 de mayo de 2020]. Disponible en: https://www.ree.es/es/red21/almacenamiento-energetico
5. Operación del sistema eléctrico | Red Eléctrica de España [Internet]. Ree.es. [citado el 1 de junio de 2020]. Disponible en: https://www.ree.es/es/actividades/operacion-del-sistema-electrico
6. PLUS M, PLUS M. M-DUINO PLC Arduino Ethernet 21 I/Os Analog/Digital PLUS [Internet]. Boot & Work Corp. S.L. [citado el 2 de junio de 2020]. Disponible en: https://www.industrialshields.com/es_ES/shop/product/is-mduino-21-m-duino-plc-arduino-ethernet-21-i-os-analog-digital-plus-3
7. Arduino - Software [Internet]. Arduino.cc. 2019 [citado el 2 de junio de 2020]. Disponible en: https://www.arduino.cc/en/main/software
8. Foundation P. Processing.org [Internet]. [citado el 2 de junio de 2020]. Disponible en: https://processing.org/
9. Modbus tools [Internet]. Modbus Slave Simulator. 2020 [citado el 3 de junio de 2020]. Disponible en: https://www.modbustools.com/modbus_slave.html
10. EasymodbusTCP Modbus Library for .NET/Java and Python – Communication library and professional tools for industrial communication [Internet]. Easymodbustcp.net. [citado el 3 de junio de 2020]. Disponible en: http://easymodbustcp.net/en/
11. loop – Aprendiendo Arduino [Internet]. Aprendiendo Arduino. 2017 [citado el 5 de junio de 2020]. Disponible en: https://aprendiendoarduino.wordpress.com/tag/loop/
12. Arduino - EEPROM [Internet]. Arduino.cc. 2019 [citado el 6 de junio de 2020]. Disponible en: https://www.arduino.cc/en/Reference/EEPROM
13. Arduino - SD [Internet]. Arduino.cc. 2019 [citado el 6 de junio de 2020]. Disponible en: https://www.arduino.cc/en/Reference/SD

14. Arduino - Ethernet [Internet]. Arduino.cc. 2019 [citado el 6 de junio de 2020]. Disponible en: https://www.arduino.cc/en/Reference/Ethernet
15. Modbus TCP Master con PLCs basados en Arduino industrial [Internet]. Boot & Work Corp. S.L. [citado el 7 de junio de 2020]. Disponible en: https://www.industrialshields.com/es_ES/blog/nuestro-blog-1/post/modbus-tcp-master-con-plcs-basados-en-arduino-industrial-103
16. Anciano, J. T. (2010). Manual de Introducción al lenguaje HTML. Formación para el Empleo. EDITORIAL CEP. 2010. [citado el 9 de junio de 2020]
17. HTML C, CSS C, Hostings C, HTML C, Mario R, ANTELO H et al. Estructura HTML básica de una página web (HTML básico) [Internet]. Haz una web. 2017 [citado el 9 de junio de 2020]. Disponible en: https://www.hazunaweb.com/curso-de-html/estructura-basica-una-pagina-web/
18. Swales, A. Open modbus/tcp specification. Schneider Electric, 29. 1999. [citado el 9 de junio de 2020]
19. Programming Arduino on Industrial Environments: Chapter #10 [Internet]. Boot & Work Corp. S.L. [citado el 10 de junio de 2020]. Disponible en: https://www.industrialshields.com/es_ES/programming-arduino-on-industrial-environments-course-10-chapters-10-159753?utm_source=Sign+in+up+to+the+course+-+%5BIS.AC002.GC%5D+and+go+to+chapter+%231&utm_medium=Email
20. Olaya, A. F. R., López, A. B., & Moreno, F. G. G. Implementación de una Red MODBUS/TCP. Ingeniería y Competitividad. 2004; 6(2), 35-44. [citado el 10 de junio de 2020]
21. Información Detallada sobre el Protocolo Modbus [Internet]. 2019 [citado el 10 de junio de 2020]. Disponible en: https://www.ni.com/es-es/innovations/white-papers/14/the-modbus-protocol-in-depth.html
22. Modbus tools [Internet]. Modbus Master Simulator. 2020 [citado el 10 de junio de 2020]. Disponible en: https://www.modbustools.com/modbus_poll.html

ANEXOS

1. Control de comunicaciones con los inversores

1.1. Registros de entrada

En la siguiente tabla aparecen los diferentes parámetros del inversor a los que se puede acceder en modo escritura desde un dispositivo maestro.

Parameter	Register address	Name/Description	Type	Transmitted value		Scaling		Unit
				Minimum	Maximum	Multiplier	Offset	
Data in 1	42240	Heartbeat[1]	uint16	0	30000	1	0	-
Data in 2	42241	Control word	bf16	0x0000	0xFFFF	1	0	-
Data in 3	42242	Active power limit	uint16	0	2000	0.1	0	$\%P_{nom}$
Data in 4	42243	Q-ref operation mode <2> VAR mode (kVAr) <3> VAR mode (%) <5> PF mode <6> VAC mode (V) <8> Q(x) regulation curve	uint16	0	8	-	-	-
Data in 5	42244	Reactive power reference (kVAr)	int16	-4000	4000	1	0	kVAr
Data in 6	42245	Reactive power reference (%)	int16	-2000	2000	0.1	0	$\%Q_{nom}$
Data in 7	42246	PF reference	int16	-1000	1000	0.001	0	-
Data in 8	42247	Vac reference[2,3]	uint16	5470	6690	0.1	0	V
Data in 9	42248	Active power limit ramp up rate	uint16	0	60000	1	0	ms / $\%P_{nom}$
Data in 10	42249	Active power limit ramp down rate	uint16	0	60000	1	0	ms / $\%P_{nom}$
Data in 11	42250	Reactive power/current ramp up rate	uint16	0	60000	1	0	ms / $\%Q_{nom}$
Data in 12	42251	Reactive power/current ramp down rate	uint16	0	60000	1	0	ms / $\%Q_{nom}$

1.2. Registros de salida

En la siguiente tabla aparecen los diferentes parámetros del inversor a los que se puede acceder en modo lectura desde un dispositivo maestro.

Parameter	Register address	Name/Description	Type	Transmitted value		Scaling		Unit
				Minimum	Maximum	Multiplier	Offset	
Data out 1	42496	Heartbeat	uint16	0	30000	1	0	-
Data out 2	42497	Inverter main status word	bf16	0x0000	0xFFFF	-	0	-
Data out 3	42498	Active power	int16	-32768	32767	1	0	kW
Data out 4	42499	Reactive power	int16	-32768	32767	1	0	kVAR
Data out 5	42500	Grid voltage (1st positive sequence RMS voltage.)	uint16	0	20000	0.1	0	V
Data out 6	42501	Grid frequency	uint16	0	10000	0.01	0	Hz
Data out 7	42502	Power factor (- inductive, + capacitive)	int16	-1000	1000	0.001	0	-

2. Código fuente

```
#include <SPI.h>
#include <EEPROM.h>
#include <SD.h>
#include <Ethernet.h>
#include <ModbusTCPMaster.h>
#include <ModbusTCPSlave.h>

#define len 16  // longitud del array que alberga direcciones IP
#define checkPeriod 5000  // periodo para checkeo de inversores
#define nmax 25 // numero máximo de inversores soportable
#define lenmax 30 // longitud máxima del nombre
#define factormax 1125  // factor máximo para escribir en inversor


// Estructura de la EEPROM
//  SISTEMA[0:21]
//    RESET/STARTING SESSION
//      MAC[0:5]
//      IP[6:9]
//      MASK[10:13]
//      GATEWAY[14:17]
//      DNS[18:21]


// (puerto 80 por defecto para HTTP):
EthernetServer server(80);
EthernetClient slave;

// Variables para manejo de ficheros
File fileR;
File fileW;

// Variables de operación
uint16_t slavePort = 502;
uint16_t masterPort = 502;
uint8_t n = 0;
uint8_t ** ipdirs;
uint16_t * portdirs;
uint32_t * npowers;
char ** names;
uint8_t * nameslen;
boolean * state;
boolean * fail;
uint32_t * inputReg;
uint32_t totalpg = 0;
uint32_t totalNP = 0;
uint16_t limit[1];
uint16_t last_limit = 0;
uint16_t factor;
uint32_t lastCheckTime = 0UL;

// Datos del systema (VALORES POR DEFECTO GUARDADOS EN EEPROM)
uint8_t sysMAC [6];
uint8_t sysIP [4];
uint8_t sysMASK [4];
uint8_t sysGATEWAY [4];
uint8_t sysDNS [4];

ModbusTCPMaster master;
ModbusTCPSlave REE(masterPort);
```

```
void setup() {

  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port
only
  }
  Serial.println("Ethernet WebServer Example");

  // inicio de la SD
  if (SD.begin()) {
    Serial.println("APERTURA DE SD CORRECTA");
  }else{
    Serial.println("ERROR EN LA APERTURA DE LA SD");
    while(1);
  }

  // Obtención de direcciones MAC, IP, MASK y GATEWAY de la EEPROM
  resetSystem();

  // iniciamos conexión Ethernet:
  reconnectSystem();

  ipdirs = (uint8_t **)malloc(nmax*sizeof(uint8_t *));
  for (int i=0; i<nmax; i++) {
    ipdirs[i] = (uint8_t *)malloc(4*sizeof(uint8_t));
  }
  names = (char **)malloc(nmax*sizeof(char *));
  for (int i=0; i<nmax; i++) {
    names[i] = (char *)malloc(lenmax*sizeof(char));
  }
  portdirs = (uint16_t *)malloc(nmax*sizeof(uint16_t));
  npowers = (uint32_t *)malloc(nmax*sizeof(uint32_t));
  nameslen = (uint8_t *)malloc(nmax*sizeof(uint8_t));
  state = (boolean *)malloc(nmax*sizeof(boolean));
  fail = (boolean *)malloc(nmax*sizeof(boolean));
  inputReg = (uint32_t *)malloc(nmax*sizeof(uint32_t));


  REE.begin();
  REE.setHoldingRegisters(limit, 1);
  while (last_limit == 0) {
    REE.update();
    last_limit = limit[0];
  }

  // leemos ficheros con los datos de los inversores
  readInvFile();
  getInputRegister();

  for (int i=0; i<n; i++) {
    if (state[i]) {
      totalNP += npowers[i];
    }
  }
  if (totalNP > 0) {
    float ffactor = floor(1000*(float)last_limit/(float)totalNP);
    factor = (uint16_t)ffactor;
    Serial.print("Limitación: ");
    Serial.println(last_limit);
```

```
    Serial.print("NP: ");
    Serial.println(totalNP);
    Serial.print("Factor: ");
    Serial.println(factor);
    writeFactor(true, factor, 0);
  }
}

void loop() {

  REE.update();

  // escucha a clientes
  EthernetClient client = server.available();

  if (client) {
    String request = getRequest(client);
    uint8_t reqType = manageRequest(request);

    switch (reqType) {
      case 0:           // login page first time (login.txt)
        fileR = SD.open("login.txt");
        break;

      case 1:           // failed loginning, retry (loginf.txt)
        fileR = SD.open("loginf.txt");
        break;

      case 2:           // homepage (home.txt)
        fileR = SD.open("home.txt");
        break;

      case 3:           // managing page (manage.txt)
        fileR = SD.open("manage.txt");
        dumpFile(client, fileR, 1784);
        managingPage(client);
        break;

      case 4:           // managing page error (managef)
        fileR = SD.open("managef.txt");
        dumpFile(client, fileR, 1945);
        managingPage(client);
        break;

      case 5:
        fileR = SD.open("managem.txt");
        dumpFile(client, fileR, 1901);
        managingPage(client);
        break;

      case 6:           // real time with page refreshing (realr.txt)
        fileR = SD.open("realr.txt");
        dumpFile(client, fileR, 1088);
        realTimePage(client);
        dumpFile(client, fileR, 152);
        client.print(limit[0]);
        client.print(" KW");
        dumpFile(client, fileR, 138);
        client.print(totalpg);
        client.print(" KW");
        break;
```

```
      case 7:           // real time without page refreshing
(real.txt)
        fileR = SD.open("real.txt");
        dumpFile(client, fileR, 1059);
        realTimePage(client);
        dumpFile(client, fileR, 152);
        client.print(last_limit);
        dumpFile(client, fileR, 138);
        client.print(totalpg);
        break;

      case 8:           // system setup page (system.txt)
        fileR = SD.open("system.txt");
        dumpFile(client, fileR, 652);
        systemSetup(client);
        break;

      case 9:           // system setup page (invalid format of IP)
(systemfi.txt)
        fileR = SD.open("systemfi.txt");
        dumpFile(client, fileR, 760);
        systemSetup(client);
        break;

      case 10:           // system setup page (invalid format of DNS)
(systemfd.txt)
        fileR = SD.open("systemfd.txt");
        dumpFile(client, fileR, 761);
        systemSetup(client);
        break;

      case 11:          // system setup page (invalid format of
GATEWAY) (systemfg.txt)
        fileR = SD.open("systemfg.txt");
        dumpFile(client, fileR, 765);
        systemSetup(client);
        break;

      case 12:          // system setup page (invalid format of MASK)
(systemfs.txt)
        fileR = SD.open("systemfs.txt");
        dumpFile(client, fileR, 762);
        systemSetup(client);
        break;

      case 13:          // system setup page (invalid format of EMAIL)
(systemfe.txt)
        fileR = SD.open("systemfe.txt");
        dumpFile(client, fileR, 763);
        systemSetup(client);
        break;

      default:          // login page by default (login.txt)
        fileR = SD.open("login.txt");
        break;
    }
    dumpFile(client, fileR, 0);
    delay(1);
    client.stop();
  }else if (millis() - lastCheckTime > checkPeriod) {
```

```
    lastCheckTime = millis();
    getInputRegister();
    totalNP = 0;
    totalpg = 0;
    for (int i=0; i<n; i++) {
      if (state[i]) {
        if (!fail[i]) {
          uint32_t teoreth = (uint32_t)(npowers[i]*factor)/1000;
          if ((inputReg[i] == teoreth)){
            fail[i] = false;
            totalNP += npowers[i];
          }else{
            fail[i] = true;
            writeFactor(false, 0u, i);
            Serial.print("DETECTADO COMPORTAMIENTO ANÓMALO - ");
            Serial.println(i);
          }
        }
        totalpg =+ inputReg[i];
      }
    }
    if (limit[0] != last_limit) { // cambio en la limitación de
potencia
      last_limit = limit[0];
      Serial.print("Nueva limitación: ");
      Serial.println(last_limit);
      // generar email
    }
    if (totalNP > 0) {
      Serial.print("Potencia nominal disponible: ");
      Serial.println(totalNP);
      float ffactor = floor(1000*(float)limit[0]/(float)totalNP);
      factor = (uint16_t)ffactor;
      Serial.print("Factor: ");
      Serial.println(factor);
      writeFactor(true, factor, 0);
    }
  }
}

void updateSystem (void) {
  // Actualización de direcciones de red en EEPROM

  // Actualización de IP para inicio predeterminado
  for (int i=6; i<6+sizeof(sysIP)/sizeof(sysIP[0]); i++) {
    EEPROM.update(i, sysIP[i-6]);
  }

  // Actualización de SUBNET MASK para inicio predeterminado
  for (int i=10; i<10+sizeof(sysMASK)/sizeof(sysMASK[0]); i++) {
    EEPROM.update(i, sysMASK[i-10]);
  }

  // Actualización de GATEWAY para inicio predeterminado
  for (int i=14; i<14+sizeof(sysGATEWAY)/sizeof(sysGATEWAY[0]); i++) {
    EEPROM.update(i, sysGATEWAY[i-14]);
  }

  // Actualizadión de DNS para inicio predeterminado
  for (int i=18; i<18+sizeof(sysDNS)/sizeof(sysDNS[0]); i++) {
    EEPROM.update(i, sysDNS[i-18]);
```

```
  }
  Serial.println("Actualizado");
}

void resetSystem (void) {
  // Recarga de las direcciones de red al sistema

  // Obtención de MAC de la EEPROM
  for (int i=0; i<sizeof(sysMAC)/sizeof(sysMAC[0]); i++) {
    EEPROM.get(i, sysMAC[i]);
  }

  // Obtención de IP de la EEPROM
  for (int i=6; i<6+sizeof(sysIP)/sizeof(sysIP[0]); i++) {
    EEPROM.get(i, sysIP[i-6]);
  }

  // Obtención de MASK de la EEPROM
  for (int i=10; i<10+sizeof(sysMASK)/sizeof(sysMASK[0]); i++) {
    EEPROM.get(i, sysMASK[i-10]);
  }

  // Obtención de GATEWWAY de la EEPROM
  for (int i=14; i<14+sizeof(sysGATEWAY)/sizeof(sysGATEWAY[0]); i++) {
    EEPROM.get(i, sysGATEWAY[i-14]);
  }

  // Obtención de DNS de la EEPROM
  for (int i=18; i<18+sizeof(sysDNS)/sizeof(sysDNS[0]); i++) {
    EEPROM.get(i, sysDNS[i-18]);
  }
}

void reconnectSystem (void) {
  // Reconexión Ethernet del sistema

  Ethernet.begin(sysMAC, sysIP, sysDNS, sysGATEWAY, sysMASK);

  // Comprobación de hardware Ethernet
  if (Ethernet.hardwareStatus() == EthernetNoHardware) {
    Serial.println("Ethernet shield was not found.  Sorry, can't run
without hardware. :(");
    while (true) {
      delay(1); // medios no conectados
    }
  }
  if (Ethernet.linkStatus() == LinkOFF) {
    Serial.println("Ethernet cable is not connected.");
  }

  // iniciamos servidor:
  server.begin();

  Serial.print("MAC address: ");
  byte macBuffer[6];
  Ethernet.MACAddress(macBuffer);
  for (byte octet = 0; octet < 6; octet++) {
    Serial.print(macBuffer[octet], HEX);
    if (octet < 5) {
      Serial.print(':');
    }
```

```
  }
  Serial.println();
  Serial.print("IP address: ");
  Serial.println(Ethernet.localIP());
  Serial.print("MASK address: ");
  Serial.println(Ethernet.subnetMask());
  Serial.print("GATEWAY address: ");
  Serial.println(Ethernet.gatewayIP());
  Serial.print("DNS address: ");
  Serial.println(Ethernet.dnsServerIP());
}

void readInvFile(void) {
  // Lectura de datos de inversores de los ficheros de uSD

  // carga las direcciones IP de los inversores
  String cadena = "";
  File fr = SD.open("invips.txt");
  while (fr.available()) {
    char c = fr.read();
    if (c!=13 && c!=10) {
      cadena.concat(c);
    }else if (c == 10) {
      cadena.concat('-');
      n += 1;
    }
  }
  fr.close();
  int i = 0;
  int guion_ant = 0;
  int guion;
  while (i<n) {
    guion = cadena.indexOf('-', guion_ant);
    String subc = cadena.substring(guion_ant, guion);
    char cc [subc.length()+1];
    subc.toCharArray(cc, subc.length()+1);
    int leidos = sscanf(cc, "%u.%u.%u.%u", &ipdirs[i][0],
&ipdirs[i][1], &ipdirs[i][2], &ipdirs[i][3]);
    if (leidos != 4) {
      Serial.print("Error guardando ip ");
      Serial.println(i);
    }
    guion_ant = guion+1;
    i += 1;
  }

  // carga los puertos de los inversores
  cadena = "";
  fr = SD.open("invports.txt");
  while (fr.available()) {
    char c = fr.read();
    if (c!=13 && c!=10) {
      cadena.concat(c);
    }else if (c == 10) {
      cadena.concat('-');
    }
  }
  fr.close();
  i = 0;
  guion_ant = 0;
  while (i<n) {
```

```
  guion = cadena.indexOf('-', guion_ant);
  String subc = cadena.substring(guion_ant, guion);
  char cc [subc.length()+1];
  subc.toCharArray(cc, subc.length()+1);
  int leidos = sscanf(cc, "%u", &portdirs[i]);
  if (leidos != 1) {
    Serial.println("Error guardando puertos.");
  }
  guion_ant = guion+1;
  i += 1;
}

// carga los valores de potencia nominal de los inversores
cadena = "";
fr = SD.open("npowers.txt");
while (fr.available()) {
  char c = fr.read();
  if (c!=13 && c!=10) {
    cadena.concat(c);
  }else if (c == 10) {
    cadena.concat('-');
  }
}
fr.close();
i = 0;
guion_ant = 0;
while (i<n) {
  guion = cadena.indexOf('-', guion_ant);
  String subc = cadena.substring(guion_ant, guion);
  char cc [subc.length()+1];
  subc.toCharArray(cc, subc.length()+1);
  int leidos = sscanf(cc, "%lu", &npowers[i]);
  if (leidos != 1) {
    Serial.println("Error guardando potencia nominal.");
  }
  guion_ant = guion+1;
  i += 1;
}

// carga los nombres de los inversores
cadena = "";
fr = SD.open("names.txt");
while (fr.available()) {
  char c = fr.read();
  if (c!=13 && c!=10) {
    cadena.concat(c);
  }else if (c == 10) {
    cadena.concat('-');
  }
}
fr.close();
i = 0;
guion_ant = 0;
while (i<n) {
  guion = cadena.indexOf('-', guion_ant);
  String subc = cadena.substring(guion_ant, guion);
  nameslen[i] = (uint8_t)subc.length();
  char cc [subc.length()+1];
  subc.toCharArray(cc, subc.length()+1);
  for (int j=0; j<subc.length(); j++) {
    names[i][j] = cc[j];
```

```
    }
    guion_ant = guion+1;
    i += 1;
  }

  for (int i=0; i<n; i++) {
    fail[i] = false;
  }

  Serial.print("Número de inversores inicial: ");
  Serial.println(n);
}

void writeInvFile(void) {
  // Actualiza el contenido de los ficheros de la uSD

  // actualización del fichero de las IPs en uSD
  File fw = SD.open("invips.txt", FILE_WRITE | O_TRUNC);
  for (int j=0; j<n; j++) {
    for (int i=0; i<3; i++) {
      fw.print(ipdirs[j][i]);
      fw.print(".");
    }
    fw.println(ipdirs[j][3]);
  }
  fw.close();

  // actualización del fichero de los puertos en SD
  fw = SD.open("invports.txt", FILE_WRITE | O_TRUNC);
  for (int j=0; j<n; j++) {
    fw.println(portdirs[j]);
  }
  fw.close();

  // actualización del fichero de las pot. nominales en SD
  fw = SD.open("npowers.txt", FILE_WRITE | O_TRUNC);
  for (int j=0; j<n; j++) {
    fw.println(npowers[j]);
  }
  fw.close();

  // actualización del fichero de los nombres en SD
  fw = SD.open("names.txt", FILE_WRITE | O_TRUNC);
  for (int j=0; j<n; j++) {
    fw.write(names[j], nameslen[j]);
    fw.println();
  }
  fw.close();
}

void dir (String IPdir, uint16_t port, String invname, uint32_t
invpow, unsigned type, uint8_t inv) {
  // Añade o elimina inversores

  if (type == 1){           // añadir inversor
    n += 1;
    char cc [IPdir.length()+1];
    IPdir.toCharArray(cc, IPdir.length()+1);
    int leidos = sscanf(cc, "%u.%u.%u.%u", &ipdirs[n-1][0], &ipdirs[n-
1][1], &ipdirs[n-1][2], &ipdirs[n-1][3]);
    if (leidos != 4) {
```

```
      Serial.println("Error añadiendo ip.");
    }
    portdirs[n-1] = port;
    nameslen[n-1] = (uint8_t)invname.length();
    char c [invname.length()+1];
    invname.toCharArray(c, invname.length()+1);
    for (int i=0; i<invname.length(); i++) {
      names[n-1][i] = c[i];
    }
    npowers[n-1] = invpow;
    state[n-1] = false;
    fail[n-1] = false;
    inputReg[n-1] = 0;

    Serial.print("Número actual de inversores: ");
    Serial.println(n);

  }else if (type == 0) {  // eliminar inversor
    n -= 1;
    for (int j=inv; j<n; j++) {
      portdirs[j] = portdirs[j+1];
      npowers[j] = npowers[j+1];
      nameslen[j] = nameslen[j+1];
      state[j] = state[j+1];
      fail[j] = fail[j+1];
      inputReg[j] = inputReg[j+1];
      for (int i=0; i<4; i++) {
        ipdirs[j][i] = ipdirs[j+1][i];
      }
      for (int i=0; i<nameslen[j]; i++) {
        names[j][i] = names[j+1][i];
      }
    }

    Serial.print("Número actual de inversores: ");
    Serial.println(n);
  }
}

void getInputRegister (void) {
  // Obtiene el valor de potencia activa de los inversores

  boolean done = false;
  uint8_t slaveIP[4];
  for (int i=0; i<n; i++) {
    for (int j=0; j<4; j++) {
      slaveIP[j] = ipdirs[i][j];
    }
    if (!slave.connected()) {
      slave.stop();
      slave.connect(slaveIP, portdirs[i]);
    }
    inputReg[i] = 0;
    if (slave.connected()) {
      Serial.println("CONECTADO");
      Serial.println(slaveIP[3]);
      if (!master.readInputRegisters(slave, 1, 2497, 1)) {
            // Failure treatment

            Serial.println("Error solicitando lectura.");
            Serial.print(slaveIP[0]);
```

```
            Serial.print(".");
            Serial.print(slaveIP[1]);
            Serial.print(".");
            Serial.print(slaveIP[2]);
            Serial.print(".");
            Serial.println(slaveIP[3]);
      }

      // Check available responses often
      if (master.isWaitingResponse()) {
        while (!done){
          ModbusResponse response = master.available();
          if (response) {
            if (response.hasError()) {
              Serial.println("Error en la respuesta obtenida.");
            } else {
              // Obtiene los datos de entradas analógicas de la
respuesta
              inputReg[i] = (uint32_t)response.getRegister(0);
              Serial.print("Valor leido: ");
              Serial.println(inputReg[i]);
            }
            done = true;
          }
        }
      }
      state[i] = true;
    }else{
      state[i] = false;
    }
    slave.stop();
  }
}

void writeFactor (boolean opperative, uint16_t factor, int inv) {
  // Escribe el factor de rendimiento en los inversores

  boolean done = false;
  uint8_t slaveIP[4];
  uint32_t inputReg = 0;

  if (factor > factormax) {
    factor = (uint16_t)factormax;
  }

  if (opperative) {
    Serial.println("Limitación de inversores disponibles.");
    for (int i=0; i<n; i++) {
      if (state[i] && !fail[i]) {
        for (int j=0; j<4; j++) {
          slaveIP[j] = ipdirs[i][j];
        }
        if (!slave.connected()) {
          slave.stop();
          slave.connect(slaveIP, portdirs[i]);
        }
        if (slave.connected()) {
          if (!master.writeSingleRegister(slave, 1, 2241, factor)) {
                // Fallo al intentar escribir el factor
                Serial.println("Error escribiendo limitación.");
                Serial.print(slaveIP[0]);
```

```
                Serial.print(".");
                Serial.print(slaveIP[1]);
                Serial.print(".");
                Serial.print(slaveIP[2]);
                Serial.print(".");
                Serial.println(slaveIP[3]);
            }

            // Check available responses often
            if (master.isWaitingResponse()) {
              while (!done){
                ModbusResponse response = master.available();
                if (response) {
                  if (response.hasError()) {
                    Serial.println("Error en la respuesta obtenida
limitando potencia.");
                  } else {
                    // Limitación escrita exitosamente
                    Serial.println("Limitación escrita exitosamente.");
                  }
                  done = true;
                }
              }
            }
          }else{
            // Serial.println("No es posible comunicarse con el
inversor.");
          }
          slave.stop();
        }
      }
    }else{
      Serial.println("Desactivación de inversor con comportamiento
anómalo.");
      for (int j=0; j<4; j++) {
        slaveIP[j] = ipdirs[inv][j];
      }
      if (!slave.connected()) {
        slave.stop();
        slave.connect(slaveIP, portdirs[inv]);
      }
      if (slave.connected()) {
        if (!master.writeSingleRegister(slave, 1, 2241, 0u)) {
              // Failure treatment
              Serial.println("Error escribiendo limitación.");
              Serial.print(slaveIP[0]);
              Serial.print(".");
              Serial.print(slaveIP[1]);
              Serial.print(".");
              Serial.print(slaveIP[2]);
              Serial.print(".");
              Serial.println(slaveIP[3]);
        }

        // Check available responses often
        if (master.isWaitingResponse()) {
          while (!done){
            ModbusResponse response = master.available();
            if (response) {
              if (response.hasError()) {
```

```
              Serial.println("Error en la respuesta obtenida limitando
potencia.");
            } else {
              // Limitación escrita exitosamente
              Serial.println("Limitación escrita exitosamente.");
            }
            done = true;
          }
        }
      }
    }else{
      //Serial.println("No es posible comunicarse con el inversor.");
    }
    slave.stop();
  }
}

String getRequest (EthernetClient client) {
  // Obtiene la parte de interés de la solicitud

  String req = "";
  boolean currentLineIsBlank = true;
  boolean first = true;

  while (client.connected()) {
    if (client.available()) {
      char c = client.read();
      if (first) {
        if (c != '\n') {
          req.concat(c);
        }else{
          first = false;
        }
      }
      // fin de la solicitud
      if (c == '\n' && currentLineIsBlank) {
        return req;
      }
      if (c == '\n') {
        // comienzo de nueva linea
        currentLineIsBlank = true;
      } else if (c != '\r') {
        // leido caracter
        currentLineIsBlank = false;
      }
    }
  }
}

uint8_t manageRequest (String req) {
  // Gestiona la petición encontrada en la solicitud
  static uint8_t lastType = 0;
  uint8_t type = 0;
  int pos_user = 0;
  int pos_pass = 0;
  char key = 'x';
  int pos_ini = 0;
  int pos_end = 0;
  String direct = "";
  char ipDir [len];
  uint16_t port;
```

```
  int connectionState = 0;
  int inversor = 0;

  Serial.println(req);

  if (req.indexOf("favicon.ico") == -1) {
    if (req.indexOf("username") != -1) {  // Inicio de sesión
      type = 1;
      pos_user = req.indexOf("Jevediah");
      pos_pass = req.lastIndexOf("keloke");
      key = req.charAt(39);
      if((pos_user != -1)&&(pos_pass != -1)){
        if (pos_user < pos_pass) {
          if((key == ' ')||(key == '&')){ // Datos correctos
            type = 2;
          }
        }
      }
    }else if (req.indexOf("system") != -1){
      type = 8;
    }else if ((req.indexOf("manage") != -1) || (req.indexOf("dir") !=
-1)) {  // Gestión de IPs
      type = 3;
      if(req.indexOf("dir") != -1) {
        if (n<nmax) {
          type = 4;
          char invip [len];
          uint8_t slaveIP[4];
          pos_ini = req.indexOf("dir");
          pos_end = req.indexOf("&", pos_ini);
          direct = req.substring(pos_ini+4, pos_end);
          boolean isValid = false;
          if (direct.length() <= 15) {
            int pos = 0;
            int points = 0;
            while(direct.indexOf(".", pos+1) != -1) {
              pos = direct.indexOf(".", pos+1);
              points += 1;
            }
            if (points == 3) {
              char ipc [len];
              direct.toCharArray(ipc, len);
              int test [4];
              if (sscanf(ipc, "%i.%i.%i.%i", &test[0], &test[1],
&test[2], &test[3]) == 4) {
                boolean fail = false;
                for (int i=0; i<4; i++) {
                  if (test[i]<0 || test[i]>255) {
                    fail = true;
                  }
                }
                if (!fail) {
                  uint8_t slaveIP[4];
                  if (sscanf(ipc, "%u.%u.%u.%u", &slaveIP[0],
&slaveIP[1], &slaveIP[2], &slaveIP[3]) == 4) {
                    isValid = true;
                  }
                }
              }
            }
          }
```

```
          if (isValid) {
            pos_ini = req.indexOf("port");
            pos_end = req.indexOf("&", pos_ini);
            if (pos_end-pos_ini > 5) {
              int puerto = (req.substring(pos_ini+5,
pos_end)).toInt();
              long int p = (long int)(req.substring(pos_ini+5,
pos_end)).toInt();
              if (p>=0 && p<=65535) {
                port = (uint16_t)(req.substring(pos_ini+5,
pos_end)).toInt();
                pos_ini = req.indexOf("name");
                pos_end = req.indexOf("&", pos_ini);
                String invname = req.substring(pos_ini+5, pos_end);
                if (invname.indexOf("+") != -1) {
                  invname.replace("+", " ");
                }
                pos_ini = req.indexOf("npow");
                pos_end = req.indexOf(" ", pos_ini);
                if ((req.substring(pos_ini+5, pos_end)).toInt()) {
                  long int testpow = (long
int)(req.substring(pos_ini+5, pos_end)).toInt();
                  if (testpow>=0) {
                    uint32_t invpow =
(uint32_t)(req.substring(pos_ini+5, pos_end)).toInt();
                    dir(direct, port, invname, invpow, 1u, 0);
                    type = 3;
                  }
                }
              }
            }else{
              port = slavePort;
              pos_ini = req.indexOf("name");
              pos_end = req.indexOf("&", pos_ini);
              String invname = req.substring(pos_ini+5, pos_end);
              invname.replace("+", " ");
              pos_ini = req.indexOf("npow");
              pos_end = req.indexOf(" ", pos_ini);
              if (req.substring(pos_ini+5, pos_end).toInt()) {
                long int testpow = (long int)req.substring(pos_ini+5,
pos_end).toInt();
                if (testpow>=0) {
                  uint32_t invpow = (uint32_t)req.substring(pos_ini+5,
pos_end).toInt();
                  dir(direct, port, invname, invpow, 1u, 0);
                  type = 3;
                }
              }
            }
          }
        }else{
          type = 5;
        }
      }
    }else if (req.indexOf("RETRY") != -1 || req.indexOf("REM") != -1)
{
      type = 3;
      if (req.indexOf("REM") != -1) {
        inversor = String(req[9]).toInt();
        dir("", 0, "", 0, 0u, inversor);
      }else{
```

```
        getInputRegister();
      }
    }else if(req.indexOf("backhome") != -1) {
      type = 2;
    }else if ((req.indexOf("view") != -1) || (req.indexOf("refresh")
!= -1)) {
      type = 6;
    }else if (req.indexOf("aplicar") != -1) {
      type = 7;
    }else if (req.indexOf("dnssys") != -1) {
      pos_ini = req.indexOf("dnssys");
      pos_end = req.lastIndexOf(" ");
      direct = req.substring(pos_ini+7, pos_end);
      boolean isValid = false;
      if (direct.length() <= 15) {
        int pos = 0;
        int points = 0;
        while(direct.indexOf(".", pos+1) != -1) {
          pos = direct.indexOf(".", pos+1);
          points += 1;
        }
        if (points == 3) {
          char ipc [len];
          direct.toCharArray(ipc, len);
          int test [4];
          if (sscanf(ipc, "%i.%i.%i.%i", &test[0], &test[1], &test[2],
&test[3]) == 4) {
            boolean fail = false;
            for (int i=0; i<4; i++) {
              if (test[i]<0 || test[i]>255) {
                fail = true;
              }
            }
            if (!fail) {
              uint8_t slaveIP[4];
              if (sscanf(ipc, "%u.%u.%u.%u", &slaveIP[0], &slaveIP[1],
&slaveIP[2], &slaveIP[3]) == 4) {
                isValid = true;
              }
            }
          }
        }
      }
      if (isValid){
        type = 8;
        uint16_t ip [4];
        direct.toCharArray(ipDir, len);
        sscanf(ipDir, "%u.%u.%u.%u", &ip[0], &ip[1], &ip[2], &ip[3]);
        for(int i=0; i<sizeof(sysDNS)/sizeof(sysDNS[0]); i++) {
          sysDNS[i] = ip[i];
        }
      }else{
        type = 10;
      }
    }else if (req.indexOf("ipsys") != -1) {
      pos_ini = req.indexOf("ipsys");
      pos_end = req.lastIndexOf(" ");
      direct = req.substring(pos_ini+6, pos_end);
      boolean isValid = false;
      if (direct.length() <= 15) {
        int pos = 0;
```

```
        int points = 0;
        while(direct.indexOf(".", pos+1) != -1) {
          pos = direct.indexOf(".", pos+1);
          points += 1;
        }
        if (points == 3) {
          char ipc [len];
          direct.toCharArray(ipc, len);
          int test [4];
          if (sscanf(ipc, "%i.%i.%i.%i", &test[0], &test[1], &test[2],
&test[3]) == 4) {
            boolean fail = false;
            for (int i=0; i<4; i++) {
              if (test[i]<0 || test[i]>255) {
                fail = true;
              }
            }
            if (!fail) {
              uint8_t slaveIP[4];
              if (sscanf(ipc, "%u.%u.%u.%u", &slaveIP[0], &slaveIP[1],
&slaveIP[2], &slaveIP[3]) == 4) {
                isValid = true;
              }
            }
          }
        }
      }
      if (isValid){
        Serial.println(direct);
        type = 8;
        uint16_t ip [4];
        direct.toCharArray(ipDir, len);
        sscanf(ipDir, "%u.%u.%u.%u", &ip[0], &ip[1], &ip[2], &ip[3]);
        for(int i=0; i<sizeof(sysIP)/sizeof(sysIP[0]); i++) {
          sysIP[i] = ip[i];
        }
      }else{
        type = 9;
      }
    }else if (req.indexOf("masksys") != -1) {
      pos_ini = req.indexOf("masksys");
      pos_end = req.lastIndexOf(" ");
      direct = req.substring(pos_ini+8, pos_end);
      boolean isValid = false;
      if (direct.length() <= 15) {
        int pos = 0;
        int points = 0;
        while(direct.indexOf(".", pos+1) != -1) {
          pos = direct.indexOf(".", pos+1);
          points += 1;
        }
        if (points == 3) {
          char ipc [len];
          direct.toCharArray(ipc, len);
          int test [4];
          if (sscanf(ipc, "%i.%i.%i.%i", &test[0], &test[1], &test[2],
&test[3]) == 4) {
            boolean fail = false;
            for (int i=0; i<4; i++) {
              if (test[i]<0 || test[i]>255) {
                fail = true;
```

```
            }
          }
          if (!fail) {
            uint8_t slaveIP[4];
            if (sscanf(ipc, "%u.%u.%u.%u", &slaveIP[0], &slaveIP[1],
&slaveIP[2], &slaveIP[3]) == 4) {
              isValid = true;
            }
          }
        }
      }
    }
    if (isValid){
      type = 8;
      uint16_t ip [4];
      direct.toCharArray(ipDir, len);
      sscanf(ipDir, "%u.%u.%u.%u", &ip[0], &ip[1], &ip[2], &ip[3]);
      for(int i=0; i<sizeof(sysMASK)/sizeof(sysMASK[0]); i++) {
        sysMASK[i] = ip[i];
      }
    }else{
      type = 12;
    }
  }else if (req.indexOf("gatewaysys") != -1) {
    pos_ini = req.indexOf("gatewaysys");
    pos_end = req.lastIndexOf(" ");
    direct = req.substring(pos_ini+11, pos_end);
    boolean isValid = false;
    if (direct.length() <= 15) {
      int pos = 0;
      int points = 0;
      while(direct.indexOf(".", pos+1) != -1) {
        pos = direct.indexOf(".", pos+1);
        points += 1;
      }
      if (points == 3) {
        char ipc [len];
        direct.toCharArray(ipc, len);
        int test [4];
        if (sscanf(ipc, "%i.%i.%i.%i", &test[0], &test[1], &test[2],
&test[3]) == 4) {
          boolean fail = false;
          for (int i=0; i<4; i++) {
            if (test[i]<0 || test[i]>255) {
              fail = true;
            }
          }
          if (!fail) {
            uint8_t slaveIP[4];
            if (sscanf(ipc, "%u.%u.%u.%u", &slaveIP[0], &slaveIP[1],
&slaveIP[2], &slaveIP[3]) == 4) {
              isValid = true;
            }
          }
        }
      }
    }
    if (isValid){
      type = 8;
      uint16_t ip [4];
      direct.toCharArray(ipDir, len);
```

```
        sscanf(ipDir, "%u.%u.%u.%u", &ip[0], &ip[1], &ip[2], &ip[3]);
        for(int i=0; i<sizeof(sysGATEWAY)/sizeof(sysGATEWAY[0]); i++)
{
          sysGATEWAY[i] = ip[i];
        }
      }else{
        type = 11;
      }
    }else if (req.indexOf("reset") != -1) {
      type = 8;
      resetSystem();
    }else if (req.indexOf("reconnect") != -1){
      type = 8;
      reconnectSystem();
    }else if (req.indexOf("ADD") != -1) {
      type = 3;
      writeInvFile();
    }else if (req.indexOf("EEPROM") != -1) {
      type = 8;
      updateSystem();
    }else{
      type = 0;
    }
  }else{
    type = lastType;
  }
  lastType = type;
  return type;
}

void managingPage (EthernetClient client) {
  // Envía el contenido dinámico de la página de gestión de inversores

  for (int j=0; j<n; j++) {
      client.println("<tr>");
      client.println("<form class=\"form-signin\">");
      client.print("<td style=\"text-align: center;\">");
      client.write(names[j], nameslen[j]);
      Serial.println(nameslen[j]);
      client.println("</td>");
      client.print("<td style=\"text-align: center;\">");
      client.print(ipdirs[j][0]);
      client.print(".");
      client.print(ipdirs[j][1]);
      client.print(".");
      client.print(ipdirs[j][2]);
      client.print(".");
      client.print(ipdirs[j][3]);
      client.println("</td>");
      client.print("<td style=\"text-align: center;\">");
      client.print(portdirs[j]);
      client.println("</td>");
      client.print("<td style=\"text-align: center;\">");
      client.print(npowers[j]);
      client.println("</td>");
      if (state[j]){
        client.println("<td style=\"text-align: center;\"><span
style=\"color:#008000;\">connected</span></td>");
      }else{
        client.println("<td style=\"text-align: center;\"><span
style=\"color:#FF0000;\">disconnected</span></td>");
```

```
      }
      client.print("<td style=\"text-align: center;\"><button
name=\"REM");
      client.print(j);
      client.println("\" class=\"btn btn-lg btn-primary btn-block\"
type=\"submit\">REMOVE</button></td>");
      client.println("</form>");
      client.println("</tr>");
  }
}

void realTimePage(EthernetClient client) {
  // Envía el contenido dinámico de la página de tiempo real

  for (int j=0; j<n; j++) {
    client.println("<tr>");
    client.print("<td style=\"text-align: center;\">");
    client.write(names[j], nameslen[j]);
    client.println("</td>");
    client.print("<td style=\"text-align: center;\">");
    client.print(ipdirs[j][0]);
    client.print(".");
    client.print(ipdirs[j][1]);
    client.print(".");
    client.print(ipdirs[j][2]);
    client.print(".");
    client.print(ipdirs[j][3]);
    client.println("</td>");
    client.print("<td style=\"text-align: center;\">");
    client.print(portdirs[j]);
    client.println("</td>");
    client.println("</td>");
    client.print("<td style=\"text-align: center;\">");
    client.print(npowers[j]);
    client.println("</td>");
    client.print("<td style=\"text-align: center;\">");
    client.print(inputReg[j]);
    client.print(" ");
    if (!state[j]) {
      client.print("(Disc.)");
    }else if (fail[j]){
      client.print("(FAILURE, CHECK INVERTER)");
    }
    client.println("</td>");
    client.println("</tr>");
  }
}

void systemSetup (EthernetClient client) {
  // Envía el contenido dinámico de la página de redes del sistema

  // IP
  client.println("<tr>");
  client.println("<form class=\"form-signin\">");
  client.println("<td style=\"text-align:
center;\"><strong>IP</strong></td>");
  client.print("<td style=\"text-align: center;\"><input
autofocus=\"\" class=\"form-control\" name=\"ipsys\" placeholder=\"");
  for (int i=0; i<sizeof(sysIP)/sizeof(sysIP[0]); i++) {
    client.print(sysIP[i]);
    if (i<sizeof(sysIP)/sizeof(sysIP[0])-1) {
```

```
      client.print(".");
    }
  }
  client.println("\" type=\"text\" />     <input
type=\"submit\" value=\"SAVE\" /></td>");
  client.println("</form>");
  client.println("</tr>");

  // DNS
  client.println("<tr>");
  client.println("<form class=\"form-signin\">");
  client.println("<td style=\"text-align:
center;\"><strong>DNS</strong></td>");
  client.print("<td style=\"text-align: center;\"><input
autofocus=\"\" class=\"form-control\" name=\"dnssys\"
placeholder=\"");
  for (int i=0; i<sizeof(sysDNS)/sizeof(sysDNS[0]); i++) {
    client.print(sysDNS[i]);
    if (i<sizeof(sysDNS)/sizeof(sysDNS[0])-1) {
      client.print(".");
    }
  }
  client.println("\" type=\"text\" />     <input
type=\"submit\" value=\"SAVE\" /></td>");
  client.println("</form>");
  client.println("</tr>");

  // GATEWAY
  client.println("<tr>");
  client.println("<form class=\"form-signin\">");
  client.println("<td style=\"text-align:
center;\"><strong>GATEWAY</strong></td>");
  client.print("<td style=\"text-align: center;\"><input
autofocus=\"\" class=\"form-control\" name=\"gatewaysys\"
placeholder=\"");
  for (int i=0; i<sizeof(sysGATEWAY)/sizeof(sysGATEWAY[0]); i++) {
    client.print(sysGATEWAY[i]);
    if (i<sizeof(sysGATEWAY)/sizeof(sysGATEWAY[0])-1) {
      client.print(".");
    }
  }
  client.println("\" type=\"text\" />     <input
type=\"submit\" value=\"SAVE\" /></td>");
  client.println("</form>");
  client.println("</tr>");

  // MASK
  client.println("<tr>");
  client.println("<form class=\"form-signin\">");
  client.println("<td style=\"text-align: center;\"><strong>SUBNET
MASK</strong></td>");
  client.print("<td style=\"text-align: center;\"><input
autofocus=\"\" class=\"form-control\" name=\"masksys\"
placeholder=\"");
  for (int i=0; i<sizeof(sysMASK)/sizeof(sysMASK[0]); i++) {
    client.print(sysMASK[i]);
    if (i<sizeof(sysMASK)/sizeof(sysMASK[0])-1) {
      client.print(".");
    }
  }
```

```
  client.println("\" type=\"text\" />     <input
type=\"submit\" value=\"SAVE\" /></td>");
  client.println("</form>");
  client.println("</tr>");
}

void dumpFile (EthernetClient client, File file, uint16_t siz) {
  // Vuelva el contenido de los ficheros de la uSD

  if (file) {
    if (siz) {
      char buf[siz];
      file.read(buf, siz);
      client.write(buf, siz);
    }else{
      while (file.available()) {
        client.write(file.read());
      }
      file.close();
    }
  }else{
    Serial.println("El fichero no está abierto.");
  }
}
```