Diseño de un circuito multiplexor de señal con Modbus

Miguel González Jiménez

Diseño de Sistemas Electrónicos

Nombre Tutor/a de TF
Carlos Gonzalo Moreno Soriano

Profesor/a responsable de la asignatura
Germán Cobo Rodríguez

Fecha Entrega
junio de 2024

Universitat Oberta de Catalunya

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Diseño de un circuito multiplexor de señal con Modbus*
Nombre del autor:	*Miguel González Jiménez*
Nombre del consultor/a:	*Carlos Gonzalo Moreno Soriano*
Nombre del PRA:	*Germán Cobo Rodríguez*
Fecha de entrega (mm/aaaa):	*06/2024*
Titulación o programa:	Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación
Área del Trabajo Final:	*Diseño de Sistemas Electrónicos*
Idioma del trabajo:	*Castellano*
Palabras clave	*Microcontrolador, Modbus, PCBA*

Resumen del Trabajo

En la fase de fabricación de un producto electrónico se utilizan medios para asegurar la calidad que van más allá de la comprobación del correcto montaje de los componentes en la PCB; existen diferentes tipos de control en el proceso de fabricación de una PCBA, como pueden ser, inspecciones del espesor de la pasta de soldadura (SPI), inspecciones de montaje de los componentes (AOI), test analógicos del circuito (ICT), o test funcionales del circuito (FCT) y es para estos dos últimos para los que se propone la solución de este proyecto.

Este proyecto se centrará en el desarrollo y montaje de un circuito para multiplexar señales procedentes de los puntos de test de un DUT (Device Under Test), lo que permitirá conducir estas señales a instrumentos de medida tales como, multímetros o cargas electrónicas programables. De forma análoga puede utilizarse como demultiplexor para llevar alimentaciones de baja corriente desde una fuente de alimentación o generador de funciones, hasta diferentes partes en un DUT.

La necesidad de este tipo de circuitos multiplexores viene dada por la robustez que ofrecen a diferencia de la utilización de otros sistemas donde interviene una instalación con cableado. La posibilidad de distribuir las señales desde el propio conector del circuito, así como su mantenibilidad, son la clave de la proliferación de este tipo de soluciones. También es importante resaltar que facilitan el "*Build to Print",* uno de los conceptos más importantes en la fabricación de máquinas de test que persigue que dos equipos iguales presenten las mínimas diferencias entre ellos.

Abstract

During the manufacturing phase of an electronic product, quality assurance methods are employed that go beyond simply verifying the correct assembly of components on the PCB. There are different types of control in the manufacturing process of a PCBA, such as solder paste thickness inspections (SPI), component assembly inspections (AOI), analog circuit tests (ICT), or functional circuit tests (FCT), and it is for these last two that the solution of this project is proposed.

This project will focus on the development and assembly of a circuit to multiplex signals from the test points of a DUT (Device Under Test), which will allow these signals to be routed to measuring instruments such as multimeters or programmable electronic loads. Similarly, it can be used as a demultiplexer to carry low current power supplies from a power supply or function generator to different parts of a DUT.

The necessity for these types of multiplexer circuits comes from their superior robustness compared to other systems that require cable installations. The ability to distribute signals from the circuit's own connector, as well as its maintainability, are key to the proliferation of such solutions. It is also important to highlight that they facilitate "Build to Print," one of the most important concepts in the manufacture of test machines, which aims to ensure that two identical pieces of equipment present the minimal differences between them.

Índice

1. Introducción ... 1
1.1. Contexto y justificación del Trabajo ... 1
1.2. Objetivos del Trabajo ... 3
1.3. Objetivos específicos ... 4
1.4. Enfoque y método seguido ... 4
1.5. Planificación del Trabajo ... 5
1.6. Breve sumario de productos obtenidos ... 8
1.7. Incidencias y riesgos ... 8
1.8. Breve descripción de los otros capítulos de la memoria ... 9
2. Materiales y métodos ... 10
2.1. Elección de la lista de materiales (BOM) ... 10
2.1.1. Microcontrolador ... 10
2.1.2. Relés de señal ... 11
2.1.3. Conector del receptor ... 11
2.1.4. Conector para instrumentación ... 13
2.1.5. Conectores de alimentación y comunicación RS485 ... 14
2.1.6. Circuitos integrados ... 14
2.1.7. Tabla de comprobación de requisitos ... 15
2.2. Diseño del esquemático en Altium Designer ... 16
2.2.1. Librería de componentes ... 16
2.2.2. Diseño del esquemático ... 16
2.3. Cálculos ... 20
2.3.1. Corriente de los LEDs ... 20
2.3.2. Integrado ULN2003 ... 21
2.4. Diseño del PCB en Altium Designer ... 22
2.4.1. Características del rutado de las pistas ... 23
2.4.2. Diseño 3D del PCB ... 24
2.4.3. Generación de los documentos para la fabricación ... 24
2.5. Aprovisionamiento de materiales ... 25
3. Diseño y desarrollo del software ... 26
3.1. Arquitectura del software ... 26
3.1.1. Capa de aplicación ... 27
3.1.2. Capa HAL ... 31
3.2. Implementación Modbus ... 32
3.2.1. Estado *Idle*. ... 32
3.2.2. Estado *Receiving.* ... 32
3.2.3. Estado *Handle.* ... 33
3.2.4. Estado *Transmitting.* ... 37
4. Resultados ... 38
4.1. Montaje del prototipo ... 38
4.2. Verificación del prototipo ... 39
5. Conclusiones y trabajos futuros ... 45
5.1. Cumplimiento de los objetivos iniciales ... 45
5.2. Líneas de trabajo futuras ... 45
6. Glosario ... 47
7. Bibliografía ... 51

Lista de figuras

Figura 1: Diferentes soluciones de test del fabricante 6TL [4] 1
Figura 2: Diagrama de bloques propuesto para plataforma Offline 2
Figura 3: Diagrama de bloques del hardware del circuito multiplexor 3
Figura 4: Planificación de tareas 6
Figura 5: Diagrama de *Gantt* 7
Figura 6: Página principal STM32G431RB [13] 10
Figura 7: Compromiso de longevidad de 10 años ST 11
Figura 8: Tarjeta "Double Pole Matrix" de 6TL con conector VPC [15] 12
Figura 9: Tarjeta NI "PXI-2527" [16] 12
Figura 10: Conectores DIN 41612 Harting [17] 12
Figura 11: Comparativa Harting DIN 41612 tipo F con 32 pines 13
Figura 12: Extracto *datasheet* Cliff FCR7359 [18] 13
Figura 13: Página del producto Omnimate SL 3.50 [19] 14
Figura 14: Tabla de comprobación de componentes 15
Figura 15: Manufacturer Part Search de Altium Designer 16
Figura 16: Configuración de Nets Altium Designer 17
Figura 17: Hoja de esquema Main.SchDoc 17
Figura 18: Hoja de esquema Relays.SchDoc 18
Figura 19: Hoja de esquema extracto de UM1817 STMicroelectronics [23] 18
Figura 20: Hoja de esquema Comm_periph.SchDoc 19
Figura 21: Inserción de directiva "Generic No ERC" 19
Figura 22: Reporte de *compilación* satisfactoria 19
Figura 23: Extracto *datasheet* LITE-ON C150EKT [24] 20
Figura 24: Extracto *datasheet* relé IM06PGR [25] 21
Figura 25: Extracto *datasheet* ULN2003A [26] 21
Figura 26: Posicionado preliminar de los componentes 22
Figura 27: Rutado del circuito 23
Figura 28: Diseño 3D del PCBA 24
Figura 29: BOM Generada por Altium Designer 25
Figura 30: Proyecto generado desde STM32CubeMX 26
Figura 31: Arquitectura del software 27
Figura 32: Estructura de la capa de aplicación 27
Figura 33: Estructura del proyecto en Keil uVision 28
Figura 34: Contenido del archivo relays.c 29
Figura 35: Extracto del archivo registers.c 30
Figura 36: Extracto del archivo hal.c 31
Figura 37: Máquina de estados diseñada para la implementación Modbus 32
Figura 38: Función check_end_of_rx 33
Figura 39: Función handle_message 33
Figura 40: Función handle_read_holding_registers 34
Figura 41: Función handle_write_single_register 35
Figura 42: Función handle_write_multiple_registers 36
Figura 43: Función send_error_response 37
Figura 44: Función transmitting_handle 37
Figura 45: PCB y componentes 38
Figura 46: PCBA 38
Figura 47: IDE keil uVision en modo depuración 39

Figura 48: Activación de todos los relés 39
Figura 49: Comprobación de continuidad 40
Figura 50: Comunicación Modbus 40
Figura 51: Modbus, activación relés 5, 6 y 10 41
Figura 52: Modbus, Write Single Register reset 41
Figura 53: Modbus, Write Multiple Registers 42
Figura 54: Modbus, escritura registro *state* 42
Figura 55: Modbus, intento de función ilegal 43
Figura 56: Modbus, intento de lectura de 4 registros desde el registro 0 43
Figura 57: Modbus, intento de lectura del registro 4 43
Figura 58: Modbus, 10 minutos de conexión 44

1. Introducción

1.1. Contexto y justificación del Trabajo

La calidad en los productos electrónicos, ya sean de consumo o industriales, están sometidos a un alto nivel de exigencia por parte de los usuarios finales. En la fase de diseño de un producto electrónico esto se debe tener muy presente, así mismo, la fase de industrialización en este tipo de productos debe seguir unos procesos de control completos, capaces de detectar desviaciones desde las primeras fases de montaje, hasta el momento del embalaje del producto final.

En la fase de montaje de una PCBA con tecnología SMT, existen varios controles de inspección en línea que persiguen la detección temprana de desviaciones, tales como variaciones en el espesor de la pasta de soldadura o correcto posicionado de los componentes, sin embargo, esto por sí sólo no garantiza el funcionamiento del circuito, será necesaria una comprobación funcional para asegurar esta parte.

Existen diferentes tipos de máquinas de test para la comprobación de circuitos a nivel funcional, estas se agrupan principalmente en tres grupos:

- Pupitres individuales. Son conocidos vulgarmente como “camas de agujas” o “camas de pinchos”, estas máquinas están enfocadas a testear productos sencillos cuyos requerimientos de test no precisan de instrumentos de medida de gran tamaño o valor.
- Plataformas de test Offline. Este tipo de máquinas se utilizan cuando los pupitres individuales no son lo suficientemente completos para la comprobación de un determinado circuito, o bien para agrupar varios productos sobre una misma plataforma; esto se consigue realizando un *fixture* de conexión para cada uno de los modelos de circuito, lo que permite reutilizar instrumentos de medida de alto valor económico.
- Plataformas de test Online. Este tipo de máquinas se integran en la línea de montaje SMT y solo se utilizan en series de producción muy grandes debido a su elevado coste.

Diferentes empresas del sector de la industrialización electrónica, como 6TL [1], IPTE [2] o EIIT [3] tienen un completo porfolio de máquinas de test. A continuación, se muestra una imagen del catálogo de 6TL, que muestra diferentes tipos de máquinas de cada una de las agrupaciones que se han mencionado anteriormente.

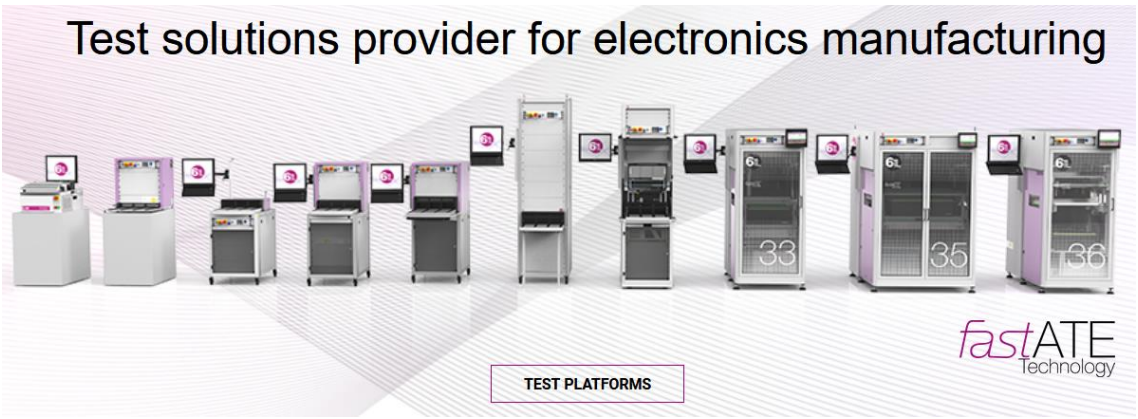


Figura 1: Diferentes soluciones de test del fabricante 6TL [4]

El objetivo de este proyecto consiste en la creación de uno de los elementos más importantes que incorporaría una plataforma de test *Offline*, como son las tarjetas de multiplexación. Estas tarjetas conectan los puntos de test de un DUT (Device Under Test), con los instrumentos de medida de forma muy sencilla, evitando un excesivo cableado. En la parte inferior, se muestra el diagrama de bloques de una plataforma de test con el fin de dar visibilidad a la utilidad de este tipo de tarjetas.

Como se puede apreciar en el diagrama, existe la zona de *Mass Interconexion* donde el ITA (Interface Test Adapter) se conecta con el *Receiver* (Receptor de conexión) y es en este último, donde la plataforma, mediante conectores destinados a tal fin, permite la conexión de fixtures de los diferentes productos.

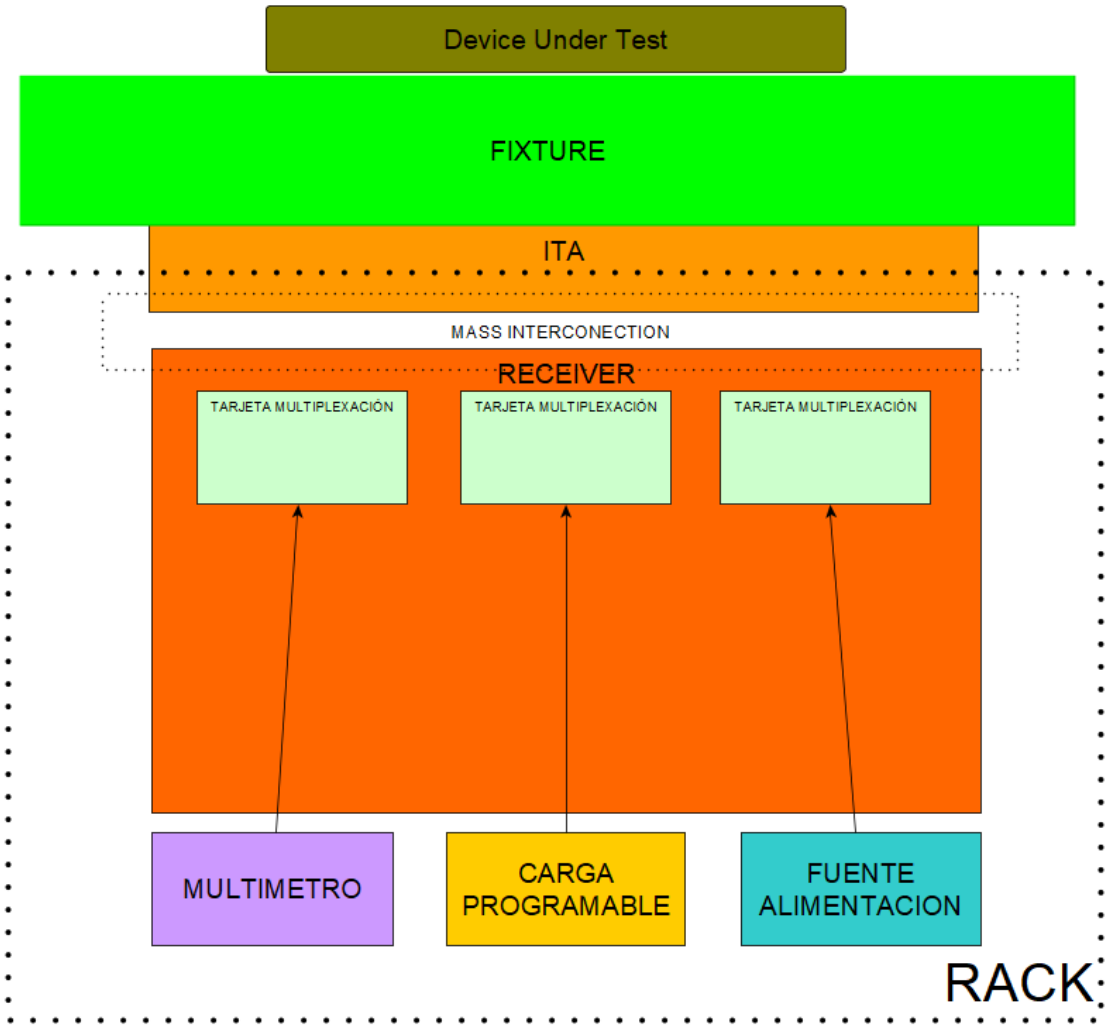


Figura 2: Diagrama de bloques propuesto para plataforma Offline

Las tarjetas de multiplexación representadas en el diagrama anterior dispondrán, por un extremo, del conector que será fijado en el *Receiver* y que servirá en última instancia como interfaz con el DUT y por el otro extremo, el circuito dispondrá de los conectores para la conexión en la plataforma, tales como los de alimentación, comunicación RS-485 y bornes para conexión con el instrumento de medida.

En la siguiente figura se muestra el diagrama de bloques del circuito multiplexor.

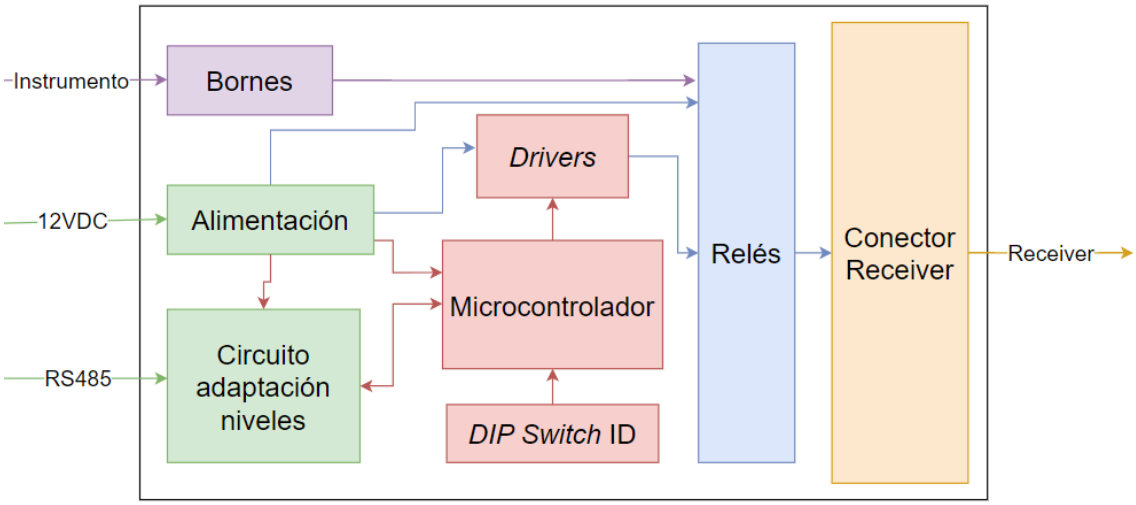


Figura 3: Diagrama de bloques del hardware del circuito multiplexor

A continuación, se hace un breve resumen de los bloques que componen el diagrama del hardware:

- Bornes: Consiste en un conector hembra tipo banana para la conexión de los instrumentos de medida.
- Alimentación. El circuito recibirá alimentación de 12 V procedentes de la máquina de test y mediante un regulador lineal distribuirá los 3,3 V necesarios para la alimentación de los circuitos integrados del circuito.
- Circuito adaptación niveles. Este consiste en la adaptación de los niveles diferenciales para el bus RS485 en niveles TTL, necesarios para su utilización desde el puerto de la UART del Microcontrolador.
- Microcontrolador. Será el encargado de gestionar la lógica de la aplicación, activando o desactivando los relés según el estado de los registros Modbus.
- Drivers. Son los encargados de la alimentación de los relés, así como de la protección de las señales de retorno en la desactivación.
- DIP Switch ID. Consiste en un interruptor tipo DIP para la configuración del número de identificador del esclavo en la red modbus.
- Relés. Serán los que conduzcan las señales desde el DUT hasta el instrumento de medida conectado en los bornes.
- Conector Receiver. Este conector será el que reciba las múltiples conexiones procedentes de un DUT; se puede considerar parte del *Receiver*.

1.2. Objetivos del Trabajo

Los principales objetivos de este trabajo final de grado son:

- Aprender a planificar las etapas que forman parte de un proyecto desde la concepción hasta los resultados.
- Relacionar los conceptos teóricos adquiridos en el aula con aplicaciones prácticas.
- Aprender a documentar los trabajos prácticos de manera precisa y rigurosa.
- Resolver las faltas de conocimiento mediante la búsqueda de información en diversos formatos.
- Comprender textos científicos y *datasheets* escritos en inglés.

1.3. Objetivos específicos

Las principales temáticas que abarcan las competencias de este trabajo final de grado son:

- Diseño placas de circuito impreso.
- Diseño de PCB con Altium Designer [5].
- Montaje de prototipos con tecnología SMD y THT.
- Microcontroladores ST.
- Diseño de software abstrayendo la aplicación del hardware.
- Depurar y compilar *firmware* con IDE Keil uVision [6].
- Comunicaciones industriales Modbus RTU.

1.4. Enfoque y método seguido

<u>Elección de lista de materiales BOM (Bill Of Materials).</u>
Esta tarea consiste en buscar los componentes necesarios para el circuito, tomando los siguientes requisitos indispensables:

- Disponibilidad de stocks en distribuidores online como RS [7], Farnell [8] o Mouser [9].
- Existencia de componentes alternativos de distinto fabricante y con el mismo *footprint*, a excepción de microcontrolador.
- Suministro garantizado o EOL (End Of Life), de mínimo 10 años.
- Cumplimiento de la directiva ROHS.

<u>Diseño del esquemático en Altium Designer.</u>
En esta tarea se realiza el diseño del esquema del circuito electrónico con la herramienta profesional Altium Designer. Se buscan las librerías de todos los componentes, o se crean en caso necesario, para que incluyan tanto su símbolo, *footprint* y representación 3D.

<u>Creación de proyecto *firmware* con drivers de bajo nivel.</u>
Consiste en la creación del proyecto base de *firmware* con los controladores de bajo nivel. Más concretamente se trata de utilizar la herramienta STM32CubeMX [10] que facilita el fabricante ST, donde se configuran todos los periféricos del microcontrolador y se genera un proyecto de *firmware* cuya aplicación está por desarrollar.

<u>Diseño del PCB en Altium Designer.</u>
Esta tarea tendrá como objetivo la realización del PCB resultante del diseño del esquemático realizado. Esta fase concluirá con la generación de los *Gerber* y ficheros necesarios para el montaje.

<u>Aprovisionamiento de PCB y BOM.</u>
Una vez finalizado el diseño del PCB se dispone de la información necesaria para aprovisionar los materiales. En esta etapa, por un lado, se realizará el pedido del PCB en algún fabricante económico online y, por otro lado, se realizará la compra de la lista de componentes con las cantidades necesarias para montar un prototipo.

<u>Diseño del programa de control.</u>
En esta fase se realizará el software contenido en el microcontrolador que se encargará del cambio de estado de los relés mediante la manipulación del valor de registros.

<u>Implementación de Modbus.</u>

Una vez realizado el software de control que permite el cambio de estado de los relés mediante registros, el software se completa implementando el protocolo esclavo Modbus. Su implementación se realizará ad-hoc a las necesidades del proyecto.

Montaje y puesta en marcha del prototipo.
Esta tarea consiste en el montaje manual del prototipo, comenzando por los componentes SMD para finalizar por los componentes THT.

Verificación del prototipo.
Llegados a este punto, el prototipo debe ser 100% funcional; se comprobará con ayuda de algún software para PC que implemente un Modbus maestro, como puede ser Modbus Poll [11].

Rediseño de PCB.
Una vez realizada la verificación, en caso necesario, se realizaría un rediseño del PCB, consistente en corregir los problemas que se hayan encontrado.

Adaptación del prototipo al nuevo diseño.
En este caso se trata de retrabajar el prototipo para adaptarlo al rediseño realizado tras la corrección de posibles fallos, incluyendo además las modificaciones de *firmware* necesarias.

Verificación final del prototipo.
Este es el punto final del desarrollo, donde se comprueba el correcto funcionamiento del circuito rediseñado.

1.5. Planificación del Trabajo

En el punto anterior se detallaban las diferentes tareas en las que se divide el trabajo y que son las que se planifican a continuación en el diagrama de *Gantt*.

Los hitos definidos corresponden con las diferentes entregas de las PEC, siendo el primer hito la entrega de la **PEC2**, con el diseño del PCB finalizado y la creación de un proyecto *firmware* base. Por su parte, el siguiente hito correspondiente con la entrega de la **PEC3,** consiste en la finalización del diseño del prototipo, su fabricación y verificación. Por último, la entrega de la **PEC4** incluirá toda la documentación finalizada y el prototipo con el diseño del hardware y *firmware*, modificado con las posibles correcciones que se hayan aplicado en caso necesario.

Como se puede observar en la planificación, se reserva un espacio amplio de tiempo entre la entrega de la PEC3 y PEC4 con la idea de absorber cualquier imprevisto que pueda surgir en el transcurso del desarrollo.

Nombre	Fecha de inicio	Fecha de fin ▲
Elección de lista de materiales (BOM)	19/3/24	24/3/24
Diseño del esquemático en Altium Designer	25/3/24	31/3/24
Creación de proyecto firmware	1/4/24	4/4/24
Diseño del PCB en Altium Designer	5/4/24	14/4/24
Entrega PEC2	15/4/24	15/4/24
Aprovisionamiento de PCB y BOM	15/4/24	29/4/24
Diseño del programa de control	15/4/24	29/4/24
Montaje y puesta en marcha del prototipo	30/4/24	7/5/24
Implementación de Modbus	30/4/24	15/5/24
Verificación del prototipo	16/5/24	19/5/24
Entrega PEC3	20/5/24	20/5/24
Rediseño de PCB	20/5/24	23/5/24
Adaptación del prototipo al nuevo diseño	24/5/24	28/5/24
Verificación final del prototipo	29/5/24	2/6/24
Cierre de documentación	3/6/24	23/6/24
Entrega PEC4	24/6/24	24/6/24

Figura 4: Planificación de tareas

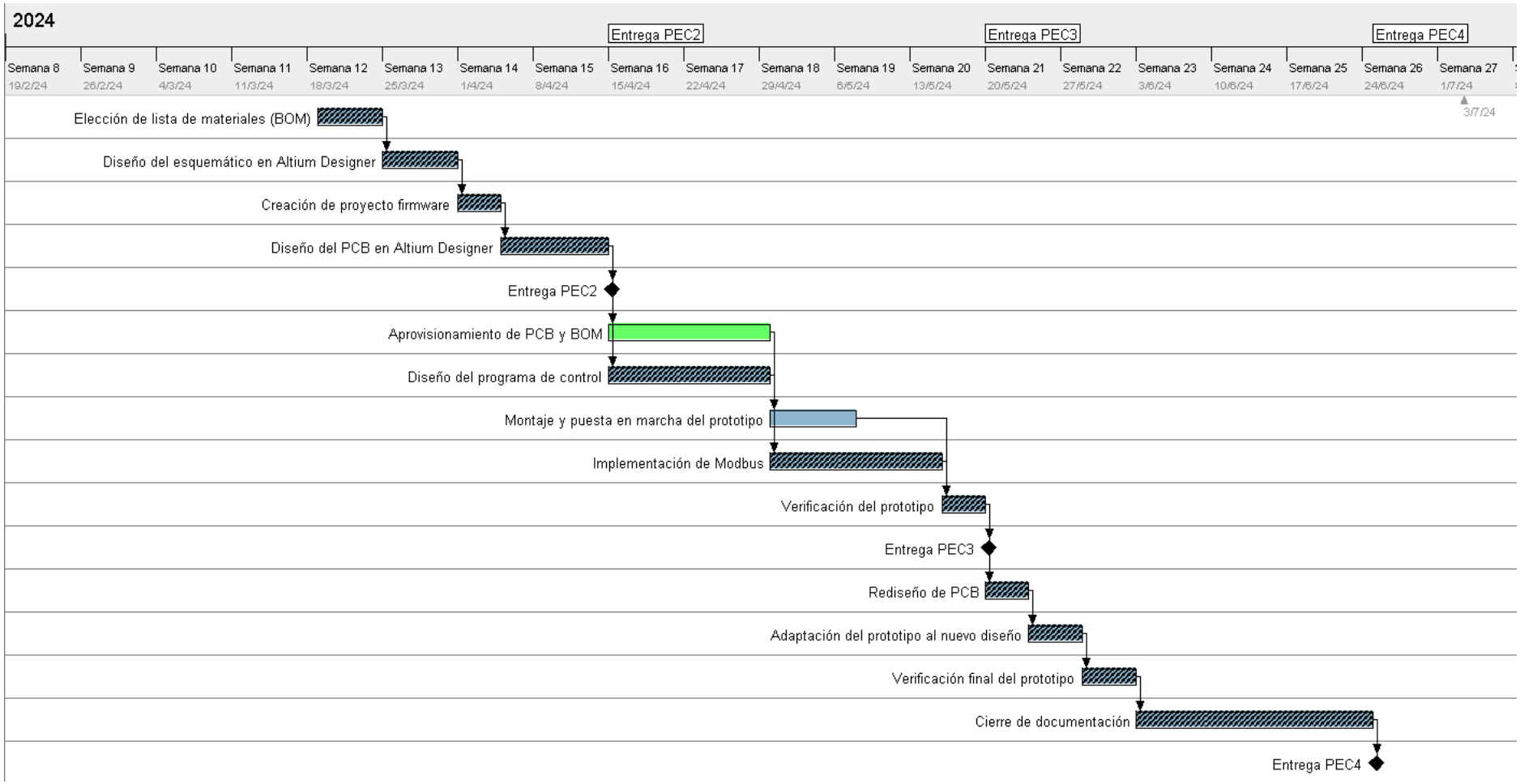


Figura 5: Diagrama de *Gantt*

1.6. Breve sumario de productos obtenidos

A la finalización del proyecto **se obtendrá un circuito totalmente funcional**, acompañado de la siguiente documentación:

- Proyecto PCB con esquemáticos en Altium Designer.
- Documentos de fabricación del PCB.
- Proyecto *firmware* en Keil uVision.
- Código objeto del microcontrolador en formato hexadecimal.
- Memoria del proyecto (este documento).

1.7. Incidencias y riesgos

A continuación, se detallan las posibles incidencias o riesgos que pueden tener lugar a lo largo del desarrollo del proyecto, así como cuál es su probabilidad, impacto y las acciones correctoras que se pueden llevar a cabo.

Error en la asignación de pines del microcontrolador. Este tipo de error es muy común debido a que los microcontroladores, a pesar de que ofrecen gran flexibilidad de configuración en sus pines, que pueden ser asignados como GPIO, UARTS, SPI, etc., también ofrecen limitaciones:

- Probabilidad: ALTA
- Impacto: ALTO
- Acciones correctoras: Se añade una tarea antes del diseño del PCB que consiste en configurar mediante el software STM32CubeMX los periféricos del micro según se han asignado en el esquemático, de esta forma, se pueden detectar errores tempranos y reasignar pines para la corrección de estos errores.

Retrasos en la entrega de materiales. Se pueden dar retrasos en la entrega de materiales de la BOM o en la fabricación del PCB.

- Probabilidad: MEDIA
- Impacto: ALTO
- Acciones correctoras: Los componentes se irán comprando sin esperar a que el PCB esté diseñado. En caso de retraso en el PCB se contemplan dos medidas, 1ª dejar margen de tiempo suficiente, incorporando tareas de firmware antes del hito correspondiente y 2ª en caso necesario, realizar otro pedido en un segundo fabricante y con entrega más rápida.

Errores de diseño en el esquemático o en el PCB. Este error es sin duda el más probable; normalmente, antes de congelar un diseño electrónico se realizan varias series de prototipos, en sectores como automoción a estas series se le conocen por las siglas A, B, o C, las cuales representan el tipo de prototipo, siendo C prototipos candidatos a la producción.

- Probabilidad: ALTA
- Impacto: ALTO
- Acciones correctoras: Se contempla la realización de un rediseño de esquemático y PCB en la planificación, así como una adaptación del prototipo mediante añadidos si fuera necesario, con el fin de validar el nuevo diseño sin ejecutar la compra de nuevo material.

Optimismo en la planificación. Una planificación optimista puede llegar a causar retrasos en las entregas.

- Probabilidad: BAJA
- Impacto: ALTO
- Acciones correctoras: Se analizan las fases que pueden causar mayor incertidumbre por las faltas de conocimiento. En este análisis se detecta dificultad en la implementación del protocolo modbus por la falta de experiencia y se decide hacer una implementación simple, adaptada únicamente a las necesidades del proyecto.

Incapacidad temporal. Este trabajo no puede ser delegado, por tanto, se debe tener en cuenta esta posibilidad.

- Probabilidad: BAJA
- Impacto: MEDIO
- Acciones correctoras: Se irán avanzando las tareas para dejar margen de tiempo suficiente con las entregas, tanto parciales como final, con el fin de minimizar el impacto en caso de que se produjera esta circunstancia.

1.8. Breve descripción de los otros capítulos de la memoria

La memoria del proyecto se divide en 7 capítulos que se describen brevemente a continuación:

1. Introducción. Contiene información sobre el contexto y objetivos del proyecto, el enfoque y método seguidos, la planificación y los resultados esperados.
2. Materiales y métodos. Se detallan las decisiones tomadas en la realización del circuito, así como los materiales a utilizar y cálculos.
3. Diseño y desarrollo del software. En este capítulo se describe la arquitectura del software y se detalla su implementación.
4. Resultados. Contiene detalles sobre el resultado obtenido, montaje y verificación del prototipo.
5. Conclusiones y trabajos futuros. Reflexión sobre los resultados obtenidos y análisis de las mejoras que pueden aplicarse al producto en futuras revisiones.
6. Glosario. Listado de términos y acrónimos.
7. Bibliografía. Listado de referencias de texto o gráficas.

2. Materiales y métodos

2.1. Elección de la lista de materiales (BOM)

En los siguientes puntos se detallan las decisiones tomadas para cada uno de los componentes más relevantes.

2.1.1. Microcontrolador

El microcontrolador del circuito se encargará de todo el procesado de la lógica de control y comunicaciones. La elección de este se ha basado principalmente en la utilización de una arquitectura presente en la actualidad como es ARM, -un dispositivo con amplios periféricos y ampliamente utilizado-, por otra parte, se valora un fabricante que ponga a disposición de los desarrolladores herramientas y software para facilitar tanto la curva de aprendizaje, como el desarrollo del *firmware*. Con todo lo anterior, el dispositivo elegido, descartando otras opciones como NXP o Microchip que cumplen de igual forma los requisitos anteriormente descritos, es STMicroelectronics [12], por ser uno de los objetivos específicos que fueron fijados en la propuesta de este trabajo.

En cuanto a la gama del microcontrolador se ha optado por un Cortex-M4; este dispositivo está sobredimensionado para la aplicación de este proyecto, en la que sería más que suficiente con un Cortex-M0, sin embargo, está prevista su utilización en más dispositivos futuros, por lo que conviene que disponga de periféricos que podrían ser utilizados como USB y FDCAN. Por otro lado, este diseño no es sensible a precio ya que no será destinado al gran público, sino que se destinará a pequeños proyectos de industrialización.

La serie elegida es la STM32G que es más actual y de similares características que la serie STM32F. Como requisito indispensable se fijó que el dispositivo tenía que garantizar una disponibilidad en el mercado de al menos 10 años y que como se puede observar en las siguientes figuras, se cumple tal requisito.

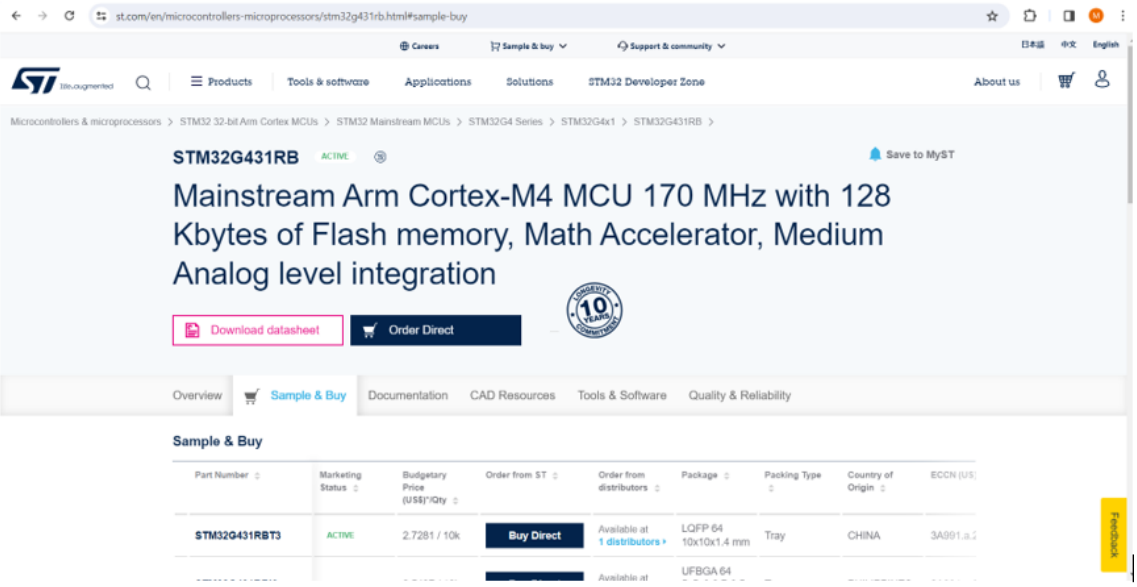


Figura 6: Página principal STM32G431RB [13]

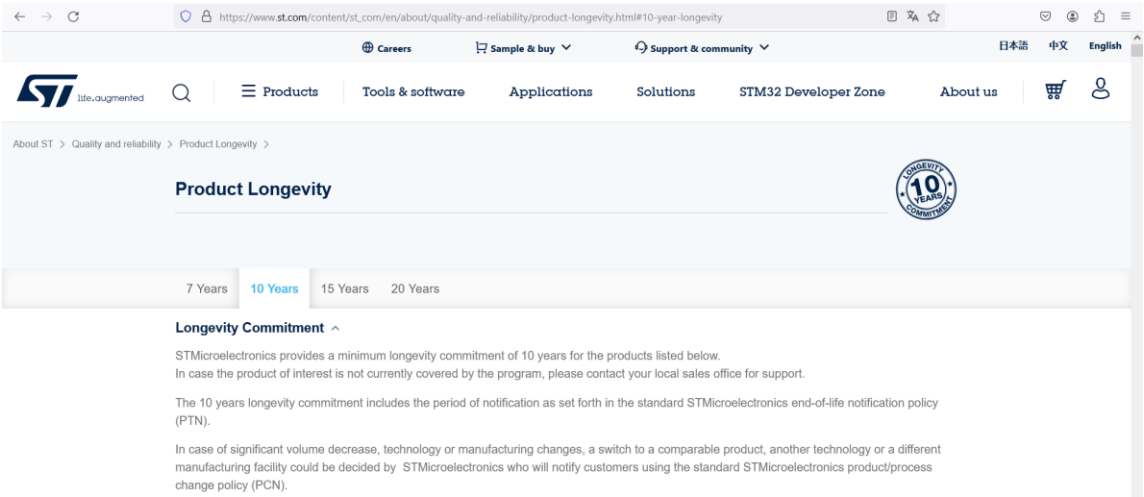


Figura 7: Compromiso de longevidad de 10 años ST

2.1.2. Relés de señal

La elección de los relés del circuito ha estado condicionada por su aplicación; al tratarse de relés para la medición de pequeñas señales, se necesita una referencia recomendada para este uso.

El fabricante TE connectivity dispone de Axicom [14], una reconocida marca de relés de baja señal para aplicaciones en el campo de las telecomunicaciones. A su vez, se ha optado por una referencia cuya característica principal sea la alta estabilidad del contacto, que en este caso está realizado con Paladio, Rutenio y un recubrimiento de oro.

La durabilidad del contacto se encuentra entre 1 x 10^5 operaciones para aplicaciones con carga resistiva de 2 A a 30 V y 2,5 x 10^6 operaciones para cargas de hasta 10 mA y 30 mV. Será importante tener este dato en cuenta para programar un reemplazo de la tarjeta una vez superada su vida útil.

Durante la exploración se valora también la utilización de relés del fabricante Omron, sin embargo, a pesar de ser de similares prestaciones presentaban un mayor coste, es por esto por lo que se deja la referencia G6K-2F-Y como componente alternativo en la BOM.

2.1.3. Conector del receptor

En cuanto al nombre “conector de receptor” se hace referencia al conector que estará presente en el receptor del equipo de test.

Tras una exploración del mercado se observa que fabricantes como 6TL utilizan conectores con terminales mecanizados del fabricante VPC que están destinados a equipos de test y garantizan más de 50.000 operaciones de conexión, sin embargo, su elevado coste, que supera los 200€ por unidad, la dificultad para adquirirlo, junto a la falta de alternativos hace que se descarte esta opción.

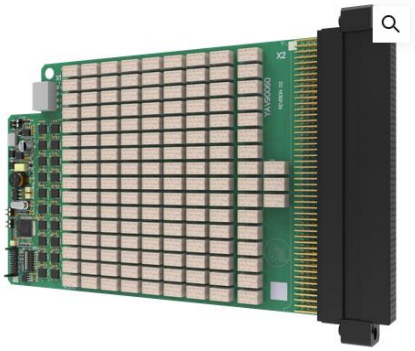


Figura 8: Tarjeta "Double Pole Matrix" de 6TL con conector VPC [15]

Por otro lado, el fabricante NI utiliza diferentes tipos de conector dependiendo de la gama de producto, siendo conectores de fabricación propia los que se utilizan en productos similares a la aplicación de este proyecto.

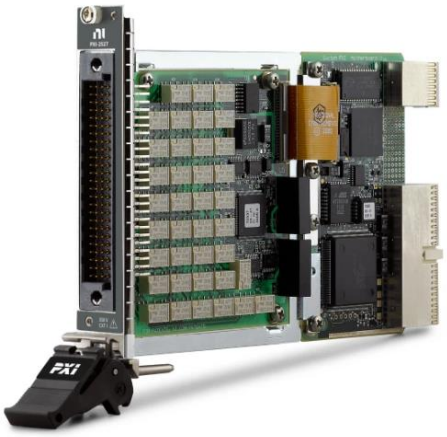
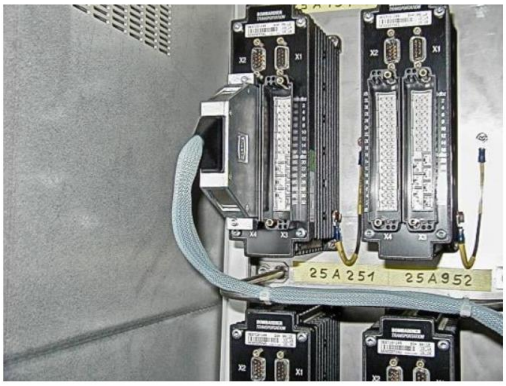


Figura 9: Tarjeta NI "PXI-2527" [16]

Tras la exploración, se decide utilizar conectores DIN 41612 que, a la vez de estar pensados para dispositivos en rack, también son utilizados para el diseño de equipos de testeo. La principal ventaja de esta elección es que hay diferentes fabricantes alternativos, entre los que se encuentran Amphenol, ERNI o Harting; finalmente se elige al fabricante Harting por tener un catálogo actualizado y stocks en diferentes distribuidores.



Conectores DIN

El clásico robusto: conectores DIN

Fiables, robustos y probados un millón de veces: nuestros conectores de contactos DIN y sus diversas variantes son objeto de mejoras y avances constantes destinados a cumplir los requisitos de nuestros clientes. Las tres familias principales de productos de la serie DIN 41612 son los conectores de potencia, los conectores de señales y los productos correspondientes en la serie har-bus®64.

Conectores DIN 41612 en la tienda electrónica (eShop) HARTING

Figura 10: Conectores DIN 41612 Harting [17]

La referencia elegida garantiza más de 400 operaciones de conexión, a la vez que supera los 2A en sus contactos, es por ello por lo que se elige la referencia 09 06 132 6901 de la serie F del estándar 41612.

Su comparación de productos

Imagen				
Descripción	CON. MACHO 32 ACOD. B-Z TIPO F 09 06 132 6901	MACHO ACOD. 32 CTOS B-Z TIPO F 09 06 132 2901	CONECTOR MACHO ACODADO 32 CTOS TIPO F 09 06 132 7901	CON. MACHO 32 ACOD. D-Z TIPO F 09 06 132 6931
Productos combinables	Alternativas Combinables Herramientas	Alternativas Combinables Herramientas	Alternativas Combinables Herramientas	Alternativas Combinables Herramientas
Descargas				
Performance level	2 acc. to IEC 60603-2	1 acc. to IEC 60603-2	3 acc. to IEC 60603-2	2 acc. to IEC 60603-2
Mating cycles	≥ 400	≥ 500	≥ 50	≥ 400
Contact configuration	Rows z and b, positions 2, 4, ... , 30, 32	Rows z and b, positions 2, 4, ... , 30, 32	Rows z and b, positions 2, 4, ... , 30, 32	Rows z and d, positions 2, 4, ... , 30, 32

4 products on compare list

Reset all | Hide all

Figura 11: Comparativa Harting DIN 41612 tipo F con 32 pines

2.1.4. Conector para instrumentación

Para la conexión con el instrumento de medida del equipo de test se busca un conector que facilite esta unión y se encuentra la opción de utilizar un *socket* para conector del tipo banana, también presente en el propio instrumento de medida. Con esta solución se simplifica la conexión del instrumento con la tarjeta mediante cables destinados a este uso.
La referencia seleccionada es la FCR7350 del fabricante Cliff, con colores negro y rojo para distinción de la polaridad.

Figura 12: Extracto *datasheet* Cliff FCR7359 [18]

2.1.5. Conectores de alimentación y comunicación RS485

Para el conexionado de la alimentación eléctrica del circuito, así como las comunicaciones RS485, se opta por un tipo de conector que permita la conexión y desconexión de forma sencilla mediante conectores aéreos, lo cual facilita la mantenibilidad del equipo de testeo. De las múltiples opciones disponibles en el mercado, se valoran los fabricantes Phoenix Contact y Weidmüller, finalmente se elige el fabricante Weidmüller por disponer de un amplio catálogo para este tipo de conectores de circuito impreso, además de presentar una mayor disponibilidad de stocks entre los distribuidores de ámbito europeo.

La referencia elegida es de la serie Omnimate, con un paso entre contactos de 3,5 milímetros.

Figura 13: Página del producto Omnimate SL 3.50 [19]

2.1.6. Circuitos integrados

En los siguientes párrafos se justifica la elección de los principales circuitos integrados que son utilizados en el diseño.

<u>Regulador de voltaje de 3,3 V</u>. Para generar la línea de 3,3 V necesaria para la alimentación del microcontrolador y resto de circuitos integrados, se opta por un regulador lineal de baja caída LDO.

Como ventajas, presenta la sencillez de utilización frente a las fuentes de alimentación, ya que solo es necesario añadir al integrado unos condensadores de desacoplo según las recomendaciones del fabricante. También hay que destacar que este tipo de reguladores de voltaje no presentan ruido añadido por conmutación a diferencia de las fuentes de alimentación; aunque no todo son ventajas, los convertidores *Buck,* que serían la alternativa más adecuada, son más eficientes y no necesitan disipar tanto calor.

Para este diseño, puesto que la tensión de entrada es de 12 V y la regulación a 3,3 V, se necesitará disipar gran cantidad de calor y por ello se toman dos decisiones: la primera es utilizar relés a 12 V que pueden ser alimentados desde la red de entrada, la

segunda decisión es utilizar un encapsulado con buena capacidad de disipación del calor como lo es SOT-223.

La referencia elegida es LD1117S33CTR del fabricante STMicroelectronics [20], aunque existen gran cantidad de componentes alternativos con el mismo encapsulado de diferentes fabricantes, como la referencia LM1117IMP-3.3/NOPB del fabricante Texas Instrument.

Array de transistores Darlington. Para la activación de los relés se utilizará un *array* de transistores Darlington, muy utilizados para este tipo de aplicaciones. La referencia elegida es ULN2003ADR del fabricante Texas Instrument [21]. Las ventajas que nos ofrece este circuito integrado son:

- **Aislamiento entre entrada y salida**. Necesita muy poca corriente para el control a pesar de una corriente significativa en la carga.
- **Simplicidad de diseño**. Ocupa muy poco espacio en comparación a la utilización de transistores discretos.
- **Protección integrada**. Incorpora un diodo de protección también conocido como diodo de *feedback*, necesario para proteger de las sobretensiones a la desconexión de una carga inductiva.
- **Alta disponibilidad**. La utilización de este circuito integrado está muy extendida y existen numerosas alternativas en el mercado.

Transceptor RS485. Para la comunicación por RS485 necesitaremos un transceptor que será el encargado transformar señales TTL procedentes de la UART del microcontrolador, en las señales diferenciales propias del estándar y viceversa.

En este caso, se ha elegido la referencia ST3485EBDR del fabricante STMicroelectronics [22], aunque hay numerosas alternativas con la misma distribución de pines, como la referencia MAX485ESA+T del fabricante Maxim, siguiendo ambos la distribución del estándar 75176. La referencia elegida incorpora además protección ESD y tiene una capacidad de transmisión de hasta 12Mbps, muy superior a la tasa necesaria en esta aplicación.

2.1.7. Tabla de comprobación de requisitos

A continuación, se realiza una tabla para facilitar la comprobación de los requisitos marcados en la elección de los componentes y que son los siguientes:

- Disponibilidad de stocks en distribuidores online, como RS, Farnell o Mouser.
- Existencia de componentes alternativos de distinto fabricante y con el mismo *footprint*, a excepción de microcontrolador.
- Suministro garantizado o EOL (End Of Life), de mínimo 10 años.
- Cumplimiento de la directiva ROHS.

Componente	Refencia principal			Referencia alternativa			EOL>10 años	ROHS
Tipo	Fabricante	Referencia	Disponible	Fabricante	Referencia	Disponible		
Conector	Harting	9061326901	Si	Amphenol	5159009326394101LF	Si	Si	Si
Conector	Weidmuller	SL3.50/03/180	Si	Anytek	EL03120A000AG	Si	Si	Si
Conector	Weidmuller	SL3.50/04/180	Si	Anytek	EL04120A000AG	Si	Si	Si
Conector	Cliff	FCR7350R	Si	Staubli	23.3200-21	Si	Si	Si
Conector	Cliff	FCR7350B	Si	Staubli	23.3200-22	Si	Si	Si
Microcontrolador	STMicroelectronic	STM32G431RBT6	Si	STMicroelectronic	STM32F…	*	Si	Si
Regulador LDO	STMicroelectronic	LD1117S33CTR	Si	Texas Instrument	LM1117IMP-3.3/NOPB	Si	Si	Si
Array Darlington	Texas Instrument	ULN2003ADR	Si	STMicroelectronic	ULN2003A	Si	Si	Si
Transceptor RS485	STMicroelectronic	ST3485EBDR	Si	Maxim Integrated	MAX485ESA+T	Si	Si	Si
Relé	TE Connectivity	IM06PGR	Si	Omron	G6K-2F-Y	Si	Si	Si

Figura 14: Tabla de comprobación de componentes

*El microcontrolador dispone de numerosas alternativas de pin a pin compatibles de las series STM32G y STM32F, sin embargo, todas ellas del mismo fabricante.

2.2. Diseño del esquemático en Altium Designer

El software elegido para la creación del esquemático y el PCB es Altium Designer Professional con licencia de evaluación para estudiantes.

2.2.1. Librería de componentes

La versión de Altium Designer Professional ofrece la posibilidad de ir creando una librería con componentes que se encuentran en stock a tiempo real en los principales distribuidores; al buscar un componente, además de comprobar si su estado es recomendado para nuevos diseños, se puede descargar el símbolo para el esquema, su huella para el PCB, representación 3D y *datasheet*.

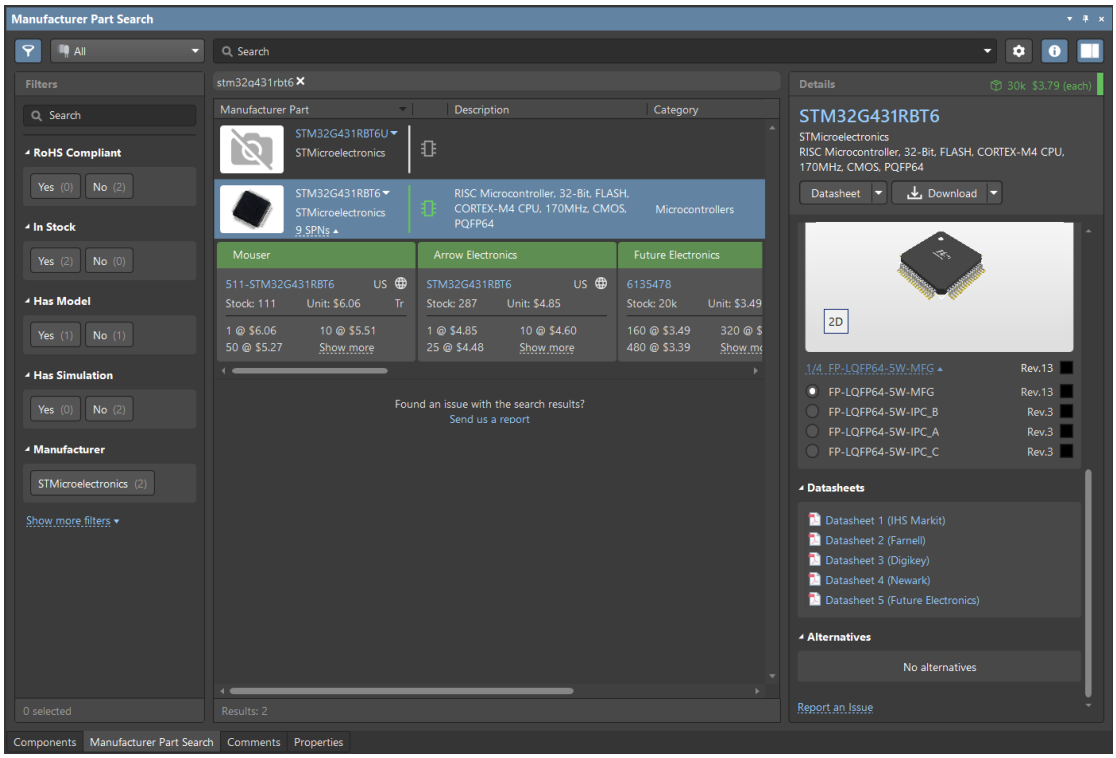


Figura 15: Manufacturer Part Search de Altium Designer

2.2.2. Diseño del esquemático

El diseño de los esquemas se realiza utilizando referencias globales, es decir, las etiquetas que se utilizan para unir puntos entre componentes se podrán utilizar en cualquier hoja de esquemático, no sólo en un único esquema, para ello modificamos la configuración en Altium Designer.

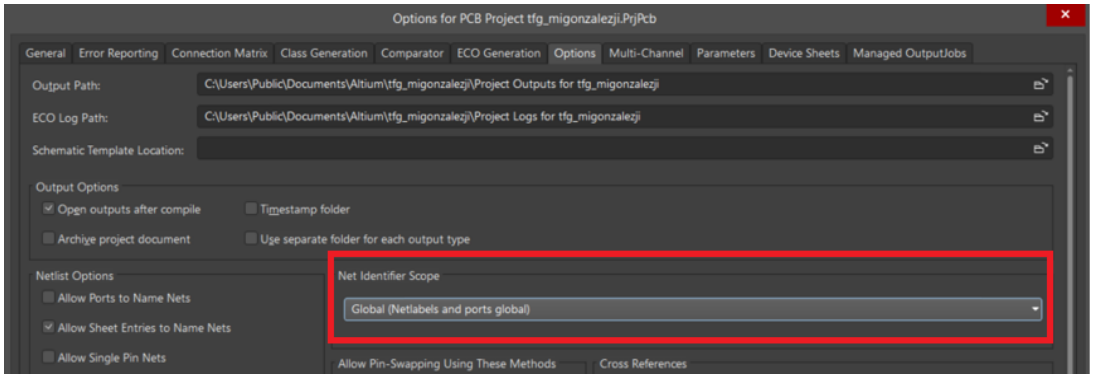


Figura 16: Configuración de Nets Altium Designer

Los esquemáticos se distribuyen en tres hojas de esquema que se detallan en los siguientes párrafos.

<u>Main.SchDoc.</u> En esta hoja de esquema se representan los siguientes elementos:

- **Conectores.** Tales como entrada de alimentación, comunicación RS485 o entrada de los bornes de conexión con el instrumento.
- **Microcontrolador.** Este se divide en dos partes, una de las cuales representa los pines de alimentación y en la otra el resto de los pines.
- **Regulador de voltaje LDO.** Se representa el regulador con su entrada a 12 V y salida a 3,3 V.
- **LED de vida.** Se incluye un LED que indicará que el circuito está en funcionamiento.
- **Conector para grabación y depuración SWD.**
- ***Dip Switch*.** Para la identificación del número de esclavo de modbus.

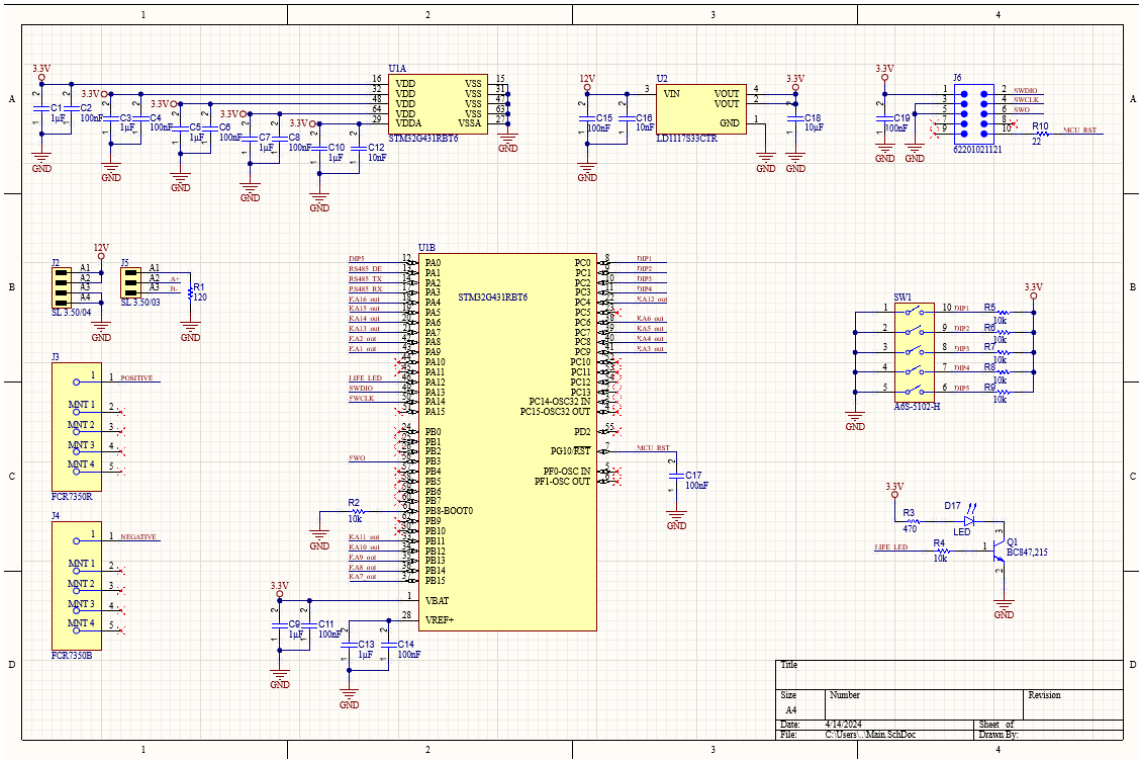


Figura 17: Hoja de esquema Main.SchDoc

Relays.SchDoc. En esta hoja se representan los 16 relés utilizados en el diseño, cada uno de ellos junto a un LED que indicará el estado. También se representan los *arrays* de Darlington y el conector de salida.

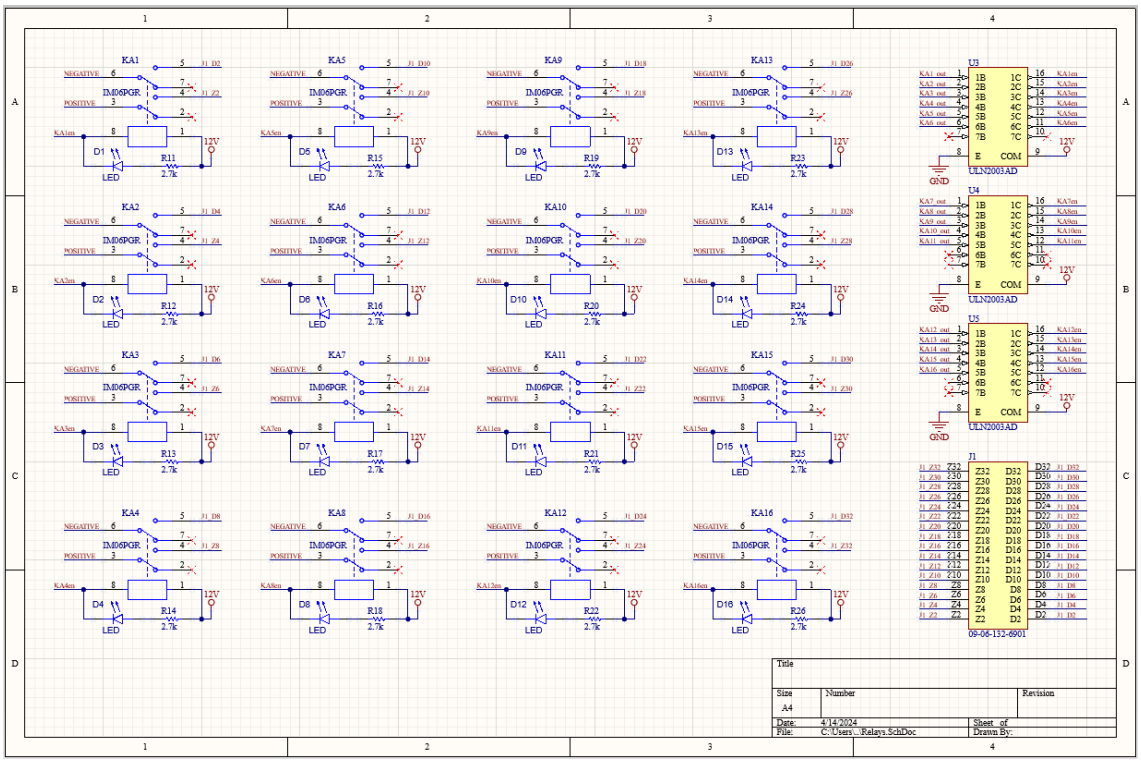


Figura 18: Hoja de esquema Relays.SchDoc

Comm_periph.SchDoc. En esta hoja de esquema se representa el hardware utilizado para la comunicación RS485. El diseño sigue el esquema recomendado por el fabricante STMicroelectronic en su manual de usuario UM1817 [23], en el que añade una red de polarización que puede ser activada mediante jumpers, así como un jumper adicional para la conexión de la resistencia de fin de línea.

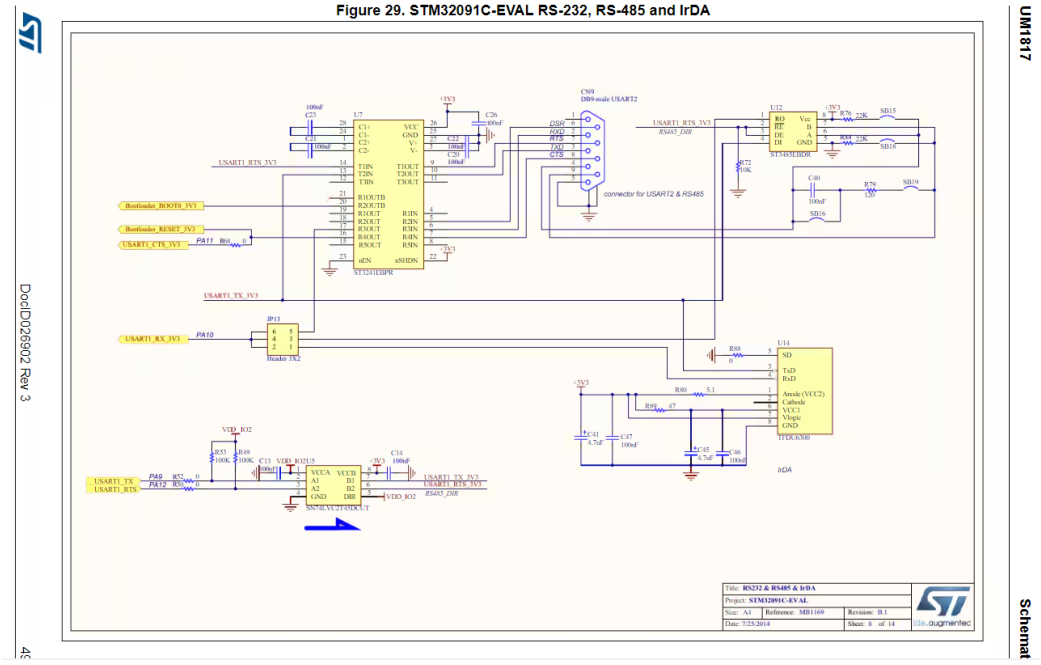


Figura 19: Hoja de esquema extracto de UM1817 STMicroelectronics [23]

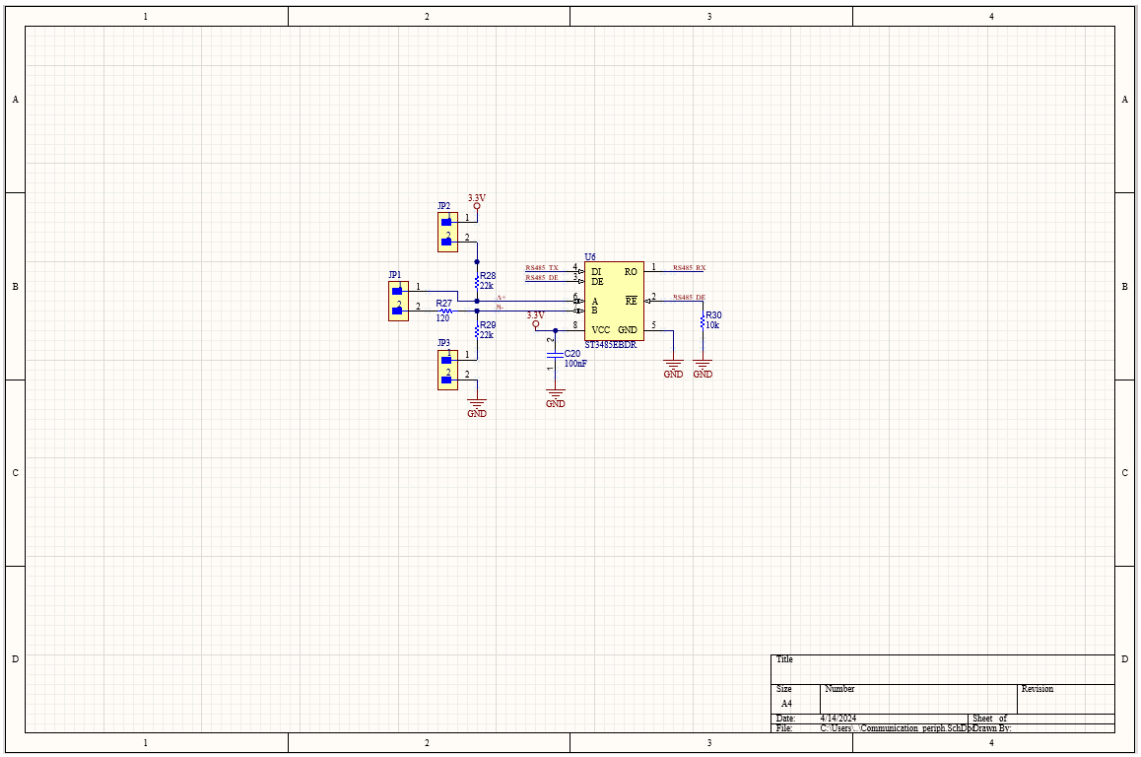


Figura 20: Hoja de esquema Comm_periph.SchDoc

Una vez realizados los esquemáticos, una buena práctica es compilar para detectar si hubiera errores tales como los siguientes:

- Errores de conexión.
- Conflictos en los nets.
- Componentes fuera de *grid*.
- Pines no conectados.

Por tanto, antes de compilar es necesario que todos los pines que no serán conectados se marquen con la directiva “Generic No ERC”.

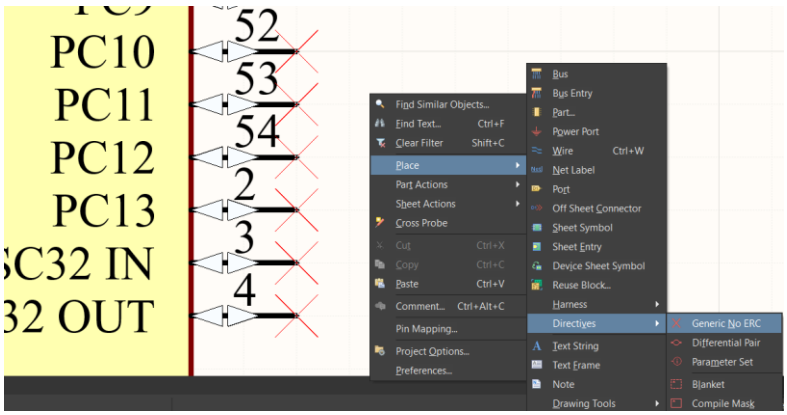


Figura 21: Inserción de directiva "Generic No ERC"

Una vez todas las conexiones están realizadas y resueltos los errores, se obtiene un reporte satisfactorio.

Figura 22: Reporte de *compilación* satisfactoria

2.3. Cálculos

2.3.1. Corriente de los LEDs

Para el cálculo de la resistencia que limitará la corriente en los LEDs, en primer lugar, se decide que estos estarán limitados en luminosidad, debido a que en esta aplicación se utilizarán como indicadores de funcionamiento.

Como se puede observar en la siguiente figura, la tensión de caída en el diodo LED será de un valor típico de 2 V para una corriente de 20 mA.

LITEON LITE-ON TECHNOLOGY CORPORATION

Property of Lite-On Only

Electrical Optical Characteristics At Ta=25℃

Parameter	Symbol	Part No. LTST-	Min.	Typ.	Max.	Unit	Test Condition
Luminous Intensity	IV	C150EKT	1.12	3.00		mcd	IF = 10mA Note 1
Viewing Angle	2θ1/2	C150EKT		130		deg	Note 2 (Fig.6)
Peak Emission Wavelength	λP	C150EKT		630		nm	Measurement @Peak (Fig.1)
Dominant Wavelength	λd	C150EKT		621		nm	Note 3
Spectral Line Half-Width	Δλ	C150EKT		24		nm	
Forward Voltage	VF	C150EKT		2.0	2.6	V	IF = 20mA
Reverse Current	IR	C150EKT			10	μA	VR = 5V
Capacitance	C	C150EKT		20		PF	VF = 0 f = 1MHZ

Figura 23: Extracto *datasheet* LITE-ON C150EKT [24]

Para el LED de vida, que va conectado a la línea de 3,3 V, se calcula la resistencia para una corriente de 3 mA:

$$R = \frac{V - V_{LED}}{I} = \frac{3{,}3 - 2}{0{,}003} = 433\ \Omega \tag{1}$$

Se elige una resistencia comercial con valor 470 Ω.

Para los LEDs de los relés que se alimentarán a 12 V se decide que emitan algo más que el LED de Vida; se realizan los cálculos para 4 mA:

$$R = \frac{V - V_{LED}}{I} = \frac{12 - 2}{0{,}004} = 2500\ \Omega \tag{2}$$

Se eligen resistencias comerciales con valor 2k7 Ω.

2.3.2. Integrado ULN2003

En la siguiente figura se muestra la potencia necesaria para activar los relés. que en este caso es de 140 mW.

IM06PGR

Signal Relays, 220 VDC Contact Voltage, 250 VAC Contact Voltage, 140mW Coil Power Rating DC, Printed Circuit Board, Axicom IM

Contact Voltage Rating	250 VAC
Coil Power Rating DC	140 mW
Coil Voltage Rating	12 VDC
Contact Switching Voltage (Max)	250 VAC
Coil Magnetic System	Monostable, DC, Polarized

Figura 24: Extracto *datasheet* relé IM06PGR [25]

La corriente que tendrá que pasar por el relé es de:

$$I = \frac{P}{V} = \frac{0{,}14}{12} = 11{,}6\ mA \tag{3}$$

La corriente que pasará por el transistor será de 11,6 mA más los 4 mA del LED, lo que da una corriente total de 15,6 mA, muy por debajo de su capacidad que se sitúa en 500 mA.

Por otro lado, el ULN2003 tendrá que soportar la disipación térmica. Se calcula la potencia máxima para el máximo 6 salidas que se utilizan, activas a la vez:

$$P = 0{,}14 * 6 = 840\ mW \tag{4}$$

Con el *datasheet* del ULM2003 se observa, por un lado, que la T_J máxima es de 125 ºC, por otro lado, que para el encapsulado elegido (SOIC), el caso más desfavorable es la resistencia térmica $R_{\vartheta JA}$.

5.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

		MIN	MAX	UNIT
V_{CC}	Collector-emitter voltage (non-V devices)	0	50	V
T_J	Junction temperature	–40	125	°C

5.4 Thermal Information

THERMAL METRIC[1]		ULx200x				UNIT
		D (SOIC)	N (PDIP)	NS (SO)	PW (TSSOP)	
		16 PINS	16 PINS	16 PINS	16 PINS	
$R_{\theta JA}$	Junction-to-ambient thermal resistance	88.6	66.7	95	114.1	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	50.1	54.2	53.3	50.3	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	49.8	46.7	57.2	59.3	°C/W
ψ_{JT}	Junction-to-top characterization parameter	12.4	33.7	19.6	9.7	°C/W
ψ_{JB}	Junction-to-board characterization parameter	49.3	46.4	56.8	58.9	°C/W

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report, SPRA953.

Figura 25: Extracto *datasheet* ULN2003A [26]

Realizando una regla de tres se calcula la resistencia térmica $R_{\vartheta JA}$:

$$R_{\theta JA} = 88{,}6 * \frac{840}{1000} = 74{,}42 ºC \tag{5}$$

Para este caso más desfavorable, al sumar la temperatura ambiente máxima de 40ºC de esta aplicación, se obtiene:

$$\Delta T = 40 + 74{,}42 = 114{,}42\ ºC \tag{6}$$

Hay que destacar que a pesar de estar muy cerca del máximo de 125 ºC, la aplicación para la que se diseña este circuito no contempla una activación continua de los relés, sino que se destina a la conexión puntual para la toma de medidas. Por tanto, se considera aceptable el resultado del cálculo térmico.

2.4. Diseño del PCB en Altium Designer

Al crear un nuevo archivo de PCB en Altium y transferir el esquemático, se consigue disponer de todos los componentes para ubicar en el PCB; como se puede apreciar en la siguiente figura, las líneas entre nodos indican la conexión pendiente de realizar y gracias a estas líneas se puede encontrar una mejor disposición de los componentes para su posterior rutado.

En la siguiente figura se muestra la disposición preliminar propuesta para comenzar el pistado.

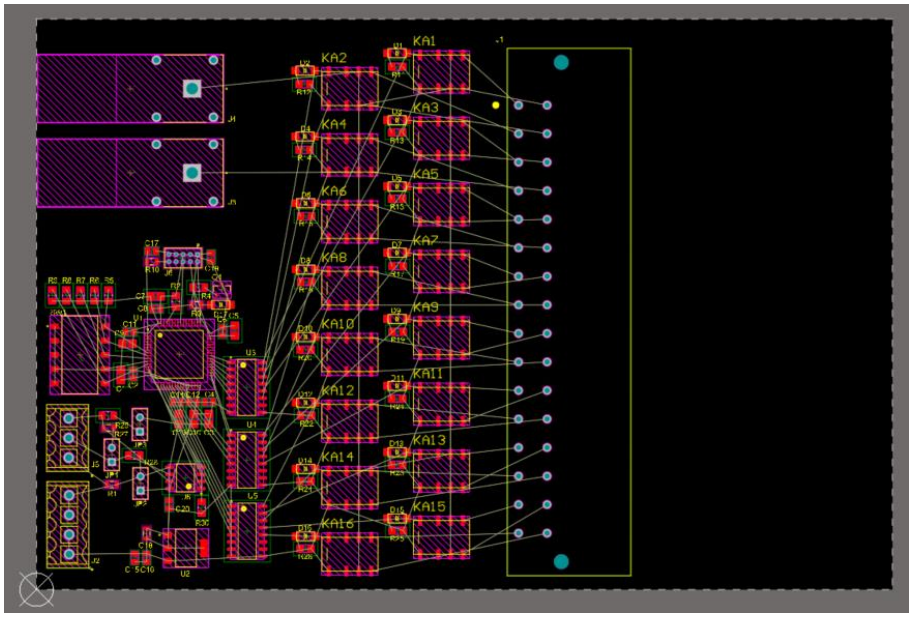


Figura 26: Posicionado preliminar de los componentes

2.4.1. Características del rutado de las pistas

El rutado de las pistas se distribuirá entra las capas *Top* y *Bottom* ya que se decide utilizar un PCB de dos capas. Para ello, diferenciaremos dos partes del circuito:

Rutado del circuito de control. Para el rutado de la zona con los componentes pertenecientes al circuito de control, se dispondrá de un plano de masa conectado con el nodo GND y que abarcará el máximo área de la capa *Bottom*. El objetivo es "cortar" el mínimo posible el plano de masa, para conseguir un retorno adecuado para todas las señales.

Por la cara *Top*, se realiza el rutado de todas las pistas; es importante una adecuada disposición de los componentes con objeto de no cruzar las mismas. Para facilitar el proceso de rutado, en primer lugar, se realizan todas las conexiones de nodos, para posteriormente conectar los nodos de alimentación. Se realiza un pequeño plano conectado a 3,3 V por la cara *Top*, principalmente por debajo de la ubicación del microcontrolador, lo que facilita el conexionado de los numerosos puntos que del mismo que requieren alimentación.

Rutado de los relés. Por su parte, el rutado de las conexiones de los relés también se distribuye entre las caras *Top* y *Bottom* como se muestra a continuación en el rutado completo del circuito.

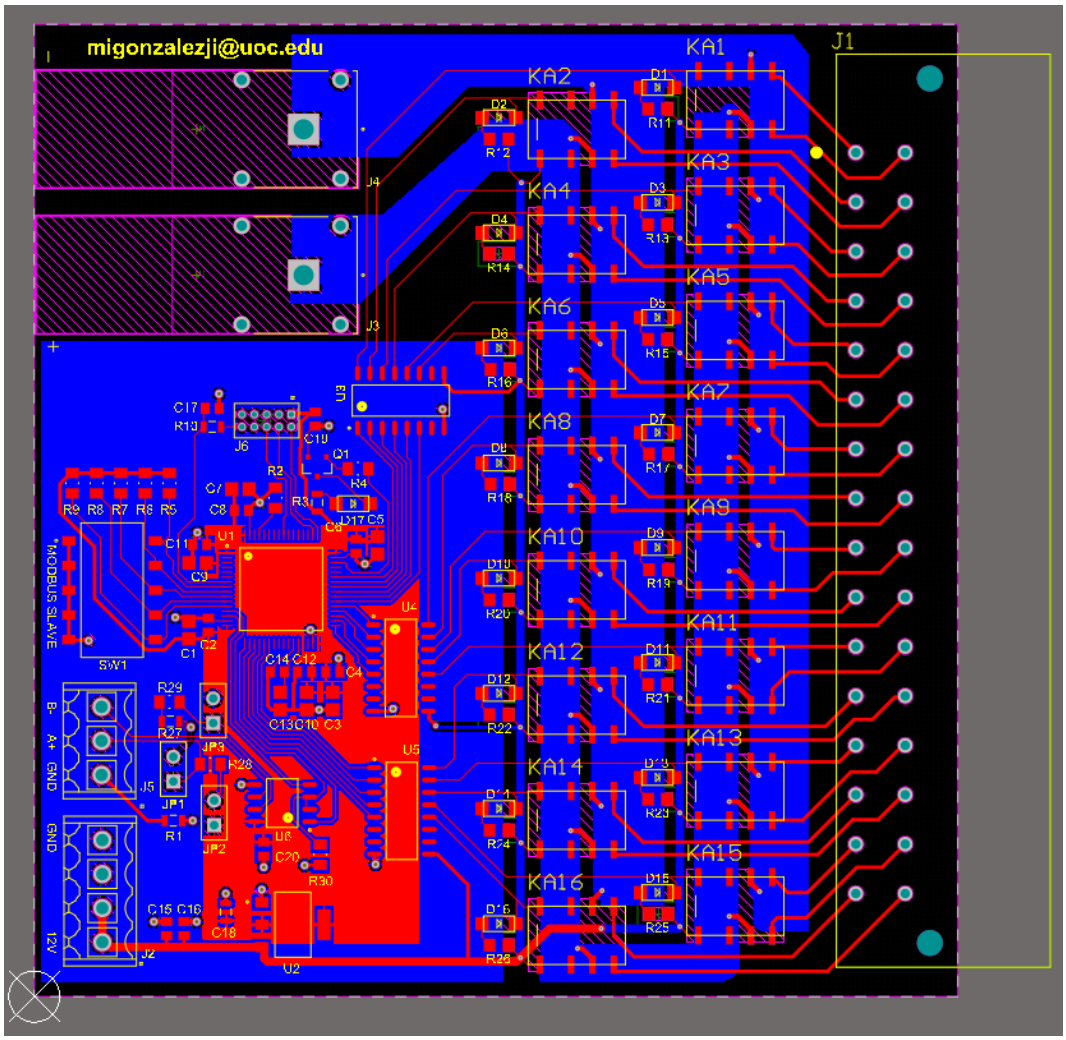


Figura 27: Rutado del circuito

2.4.2. Diseño 3D del PCB

Desde Altium Designer se permite *renderizar* el PCBA y observar cómo quedará aproximadamente una vez fabricado.

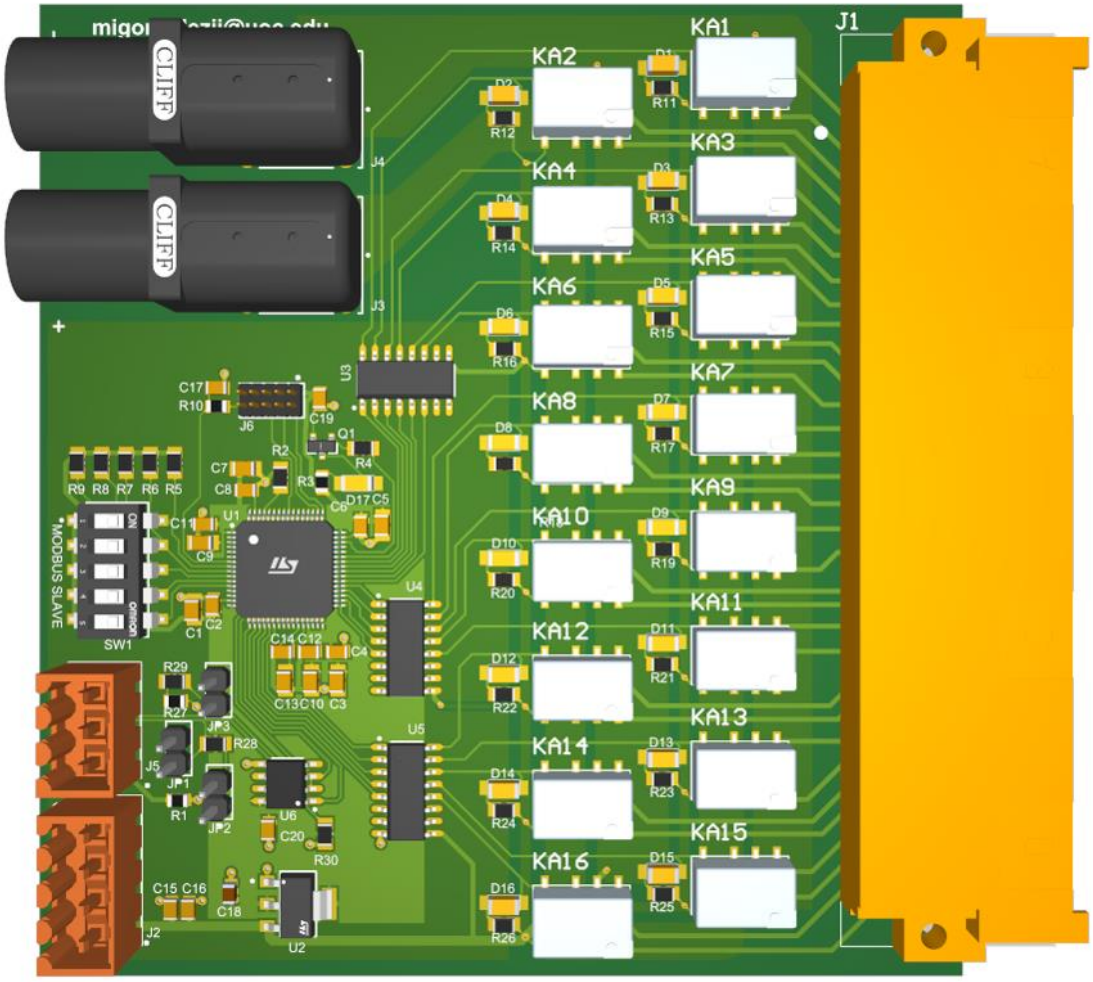


Figura 28: Diseño 3D del PCBA

2.4.3. Generación de los documentos para la fabricación

Una vez se finaliza el diseño del PCB se está en disposición de generar los archivos necesarios para la fabricación. Para este caso, el montaje del PCBA se realiza manualmente por lo que se siguen los siguientes pasos:

- **Pedido del PCB**. Es necesaria la generación de los ficheros con la información de las capas del PCB (*Gerber*), así como el archivo con los taladros del circuito.
- **Pedido del material**. Es necesario realizar el pedido de todos los componentes que aparecen en la BOM con las cantidades necesarias.
- **Montaje manual del PCBA**. Es necesario un *pdf* con la posición de los componentes, sin embargo, por estar todos ellos identificados sobre el PCB, este paso no será imprescindible.

2.5. Aprovisionamiento de materiales

Con la finalización del diseño del PCB, se generan los archivos necesarios para llevar a cabo la fabricación de un PCBA completo y se realizan los pedidos del PCB y resto de materiales.

Para la fabricación del PCB se elige el fabricante JLCPCB [27], la decisión se basa en el precio económico que ofrece al disponer de una promoción para nuevos usuarios.

Para la compra de los componentes se necesita la BOM generada; en la figura inferior se muestra el contenido de las columnas creadas por Altium Designer de forma automática.

Name	Description	Designator	Revision ID	Revision State	Revision Status	Quantity	Manufacturer 1	Manufacturer Part Number 1	Manufacturer Lifecycle 1	Supplier 1
09-06-132-6901	Backplane Connect	J1	CMP-002-00078-1	Draft	Up to date	1	HARTING Technology	09-06-132-6901	Volume Production	Rs
M20-9990246	Board Connector, 2	JP1, JP2, JP3	CMP-002-00080-2	Draft	Up to date	3	Harwin	M20-9990246	Volume Production	Mouser
Weimuller 3pin 3.5mm	Board Connector, 3	J5	CMP-002-00075-1	Draft	Up to date	1	Weidmuller	1604480000	Volume Production	Arrow Electron
Weidmuller 4 pin 3.5mm	Board Connector, 4	J2	CMP-002-00079-1	Draft	Up to date	1	Weidmuller	1604490000	Volume Production	Arrow Electron
62201021121	Board Connector, 1	J6	CMP-002-00071-1	Draft	Up to date	1	Wurth Electronics	62201021121	Volume Production	Newark
Capacitor 1 uF +/- 10% 25 V 0805	Chip Capacitor, 1 uF	C1, C3, C5, C7, C9,	CMP-001-00010-6	Draft	Up to date	7	Kyocera AVX	08053C105KAZ2A	Volume Production	Rs
Capacitor 10 nF +/-10% 50 V 080	Chip Capacitor, 10 n	C12, C16	CMP-001-00083-1	Draft	Up to date	2	Yageo	CC0805KRX7R9BB103	Volume Production	Digikey
Capacitor 10µF +/-20% 25V 0805	Chip Capacitor, 10µ	C18	CMP-001-00079-4	Draft	Up to date	1	TDK	C2012X5R1E106K125AB	Volume Production	Arrow Electron
Capacitor 100 nF +/-10% 50 V 08	Chip Capacitor, 100	C2, C4, C6, C8, C11	CMP-001-00018-6	Draft	Up to date	10	Yageo	CC0805KRX7R9BB104	Volume Production	Newark
Resistor 2k7 +/-1% 0805 125 mW	Chip Resistor, 2k7 K	R11, R12, R13, R14,	CMP-009-00182-2	Draft	Up to date	16	Vishay	CRCW08052K70FKEA	Volume Production	Mouser
Resistor 10k +/-1% 0805 125 mW	Chip Resistor, 10 KO	R2, R4, R5, R6, R7, R	CMP-009-00018-8	Draft	Up to date	8	Yageo	RC0805FR-0710KL	Volume Production	Rs
Resistor 22k +/-1% 0805 125 mW	Chip Resistor, 22 KO	R28, R29	CMP-009-00179-3	Draft	Up to date	2	Yageo	RC0805FR-0722KL	Volume Production	Mouser
Resistor 22R +/-1% 0805 125 mW	Chip Resistor, 22 OH	R10	CMP-009-00181-3	Draft	Up to date	1	Panasonic	ERJ-6ENF22R0V	Volume Production	Digikey
Resistor 120R +/-5% 0805 125 m	Chip Resistor, 120 O	R1, R27	CMP-009-00180-3	Draft	Up to date	2	Vishay Dale	CRCW0805120RJNEA	Volume Production	Newark
Resistor 470R +/-5% 0805 125 m	Chip Resistor, 470 O	R3	CMP-009-00122-6	Draft	Up to date	1	Vishay Dale	RCG0805470RJNEA	Volume Production	Arrow Electron
LD1117S33CTR	Fixed Positive LDO R	U2	CMP-019-00049-1	Draft	Up to date	1	STMicroelectronics	LD1117S33CTR	Volume Production	Digikey
ULN2003AD	High-Voltage, High-	U3, U4, U5	CMP-2000-04952-1	Released	Up to date	3	Texas Instruments	ULN2003AD	Volume Production	Arrow Electron
FCR7350R	Interconnection Dev	J3	CMP-002-00077-1	Draft	Up to date	1	Cliff	FCR7350R	Volume Production	Digikey
FCR7350B	Interconnection Dev	J4	CMP-002-00073-1	Draft	Up to date	1	Cliff	FCR7350B	Volume Production	Digikey
ST3485EBDR	Line Transceiver, 1 F	U6	CMP-00024-00007-1	Draft	Up to date	1	STMicroelectronics	ST3485EBDR	Volume Production	Mouser
IM06PGR	Power/Signal Relay	KA1, KA2, KA3, KA4,	CMP-008-00005-2	Draft	Up to date	16	TE Connectivity	IM06PGR	Volume Production	Arrow Electron
STM32G431RBT6	RISC Microcontrolle	U1	CMP-020-00023-1	Draft	Up to date	1	STMicroelectronics	STM32G431RBT6	Volume Production	Future Electron
LTST-C150EKT	Single Color LED, Re	D1, D2, D3, D4, D5,	CMP-007-00037-1	Draft	Up to date	17	Vishay Lite-On	LTST-C150EKT	Volume Production	Mouser
A6S-5102-H	Slide Dip Switch, 5 S	SW1	CMP-011-00015-1	Draft	Up to date	1	Omron	A6S-5102-H	Volume Production	Rs
BC847,215	Small Signal Bipolar	Q1	CMP-013-00034-1	Draft	Up to date	1	Nexperia	BC847,215	Volume Production	Arrow Electron

Figura 29: BOM Generada por Altium Designer

3. Diseño y desarrollo del software

Con la creación del proyecto software desde el asistente de STM32CubeMX, se obtiene un proyecto preparado para compilar sin errores. Sin embargo, la estructura de este está preparada para que el desarrollador incluya su código en espacios los cuales serán respetados por la herramienta en caso de volver a generar de nuevo el proyecto. A continuación, se muestra la estructura generada:

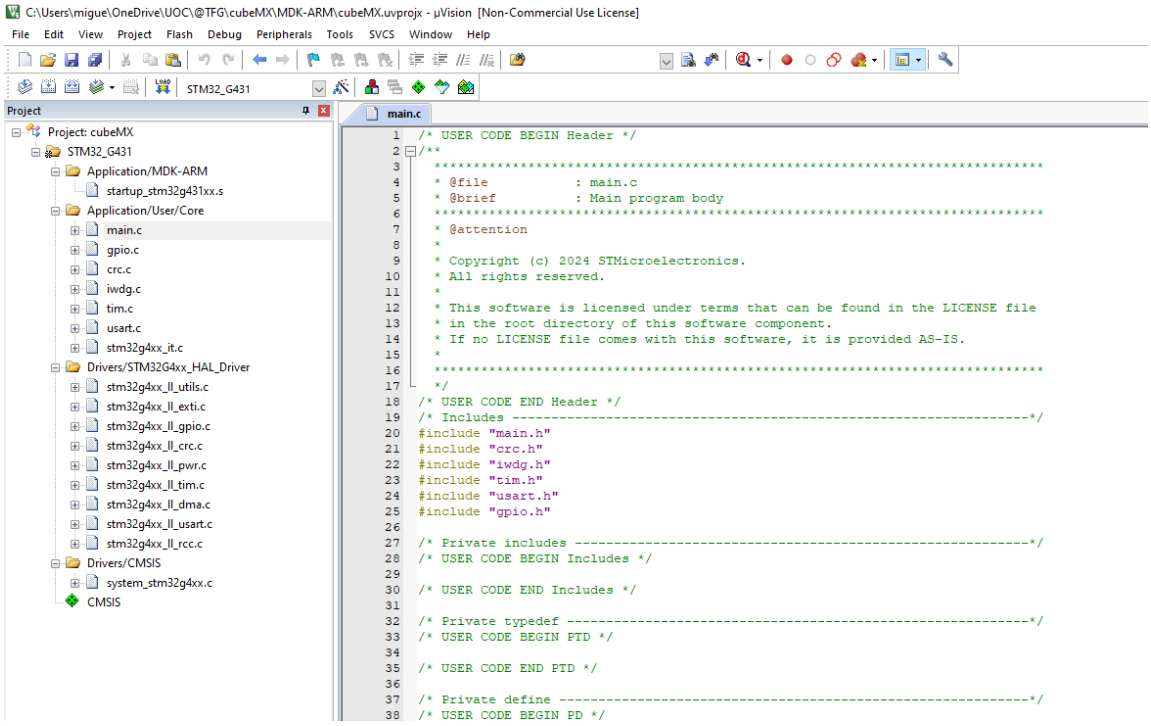


Figura 30: Proyecto generado desde STM32CubeMX

Como se puede observar, en el archivo abierto "*main.c*", se muestran las marcas donde se debe introducir código para que no sea eliminado en caso de nueva generación del proyecto.

El código generado por STM32CubeMX se utiliza como punto de partida y para el manejo de los periféricos, no obstante, se reestructuran las carpetas y se distribuyen por capas los diferentes archivos, con objeto de separar la parte de código de la aplicación, que debe ser agnóstica al tipo de hardware, de la parte de código que comunica con el hardware.

3.1. Arquitectura del software

A continuación, se muestra la estructura que sigue el software y que se detallará en los siguientes puntos de este apartado.

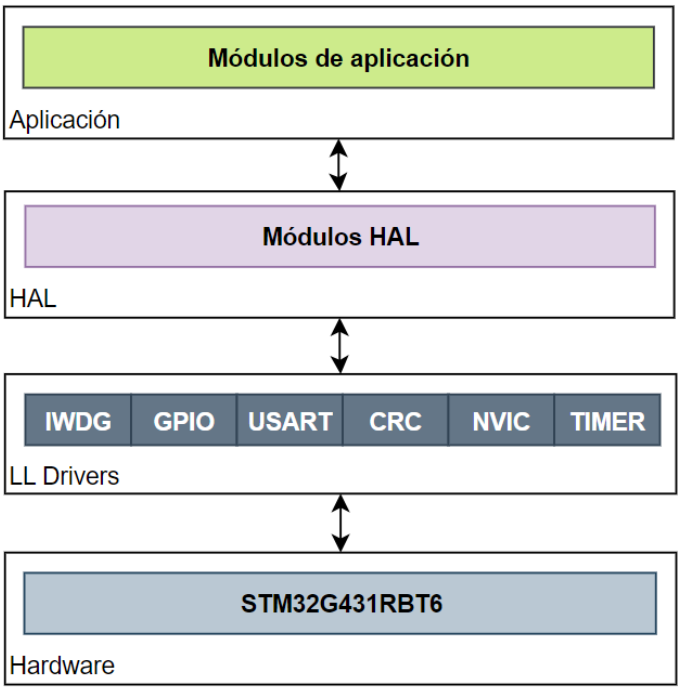


Figura 31: Arquitectura del software

3.1.1. Capa de aplicación

En la siguiente figura se muestra la estructura de bloques de la aplicación, así como su interacción con la capa inferior (HAL). Cada uno de los bloques representados está contenido entre dos archivos, uno con extensión ".c" que contine el código, y otro con extensión ".h" que contiene las cabeceras de funciones o macros que serán públicas para el resto de los módulos. El caso del bloque Main será diferente y no contendrá archivo ".h" ya que no publicará nada hacia el exterior.

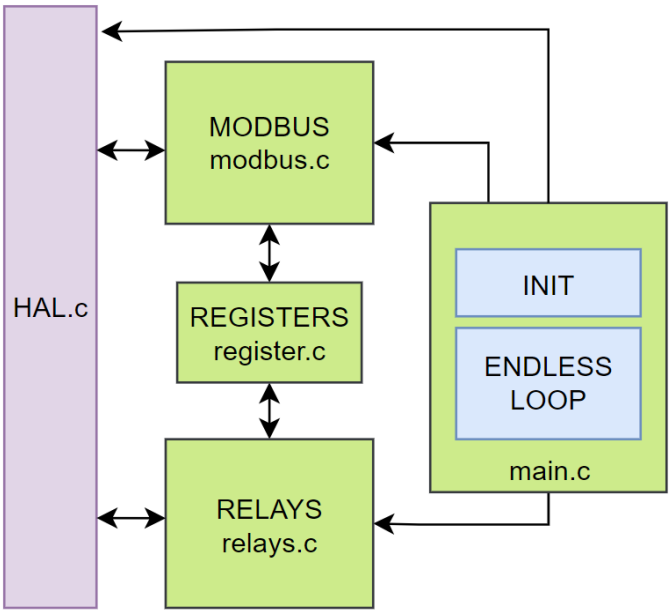


Figura 32: Estructura de la capa de aplicación

Una vez reestructurado el proyecto de *firmware* e implementado el código del bloque Main, presenta el siguiente aspecto.

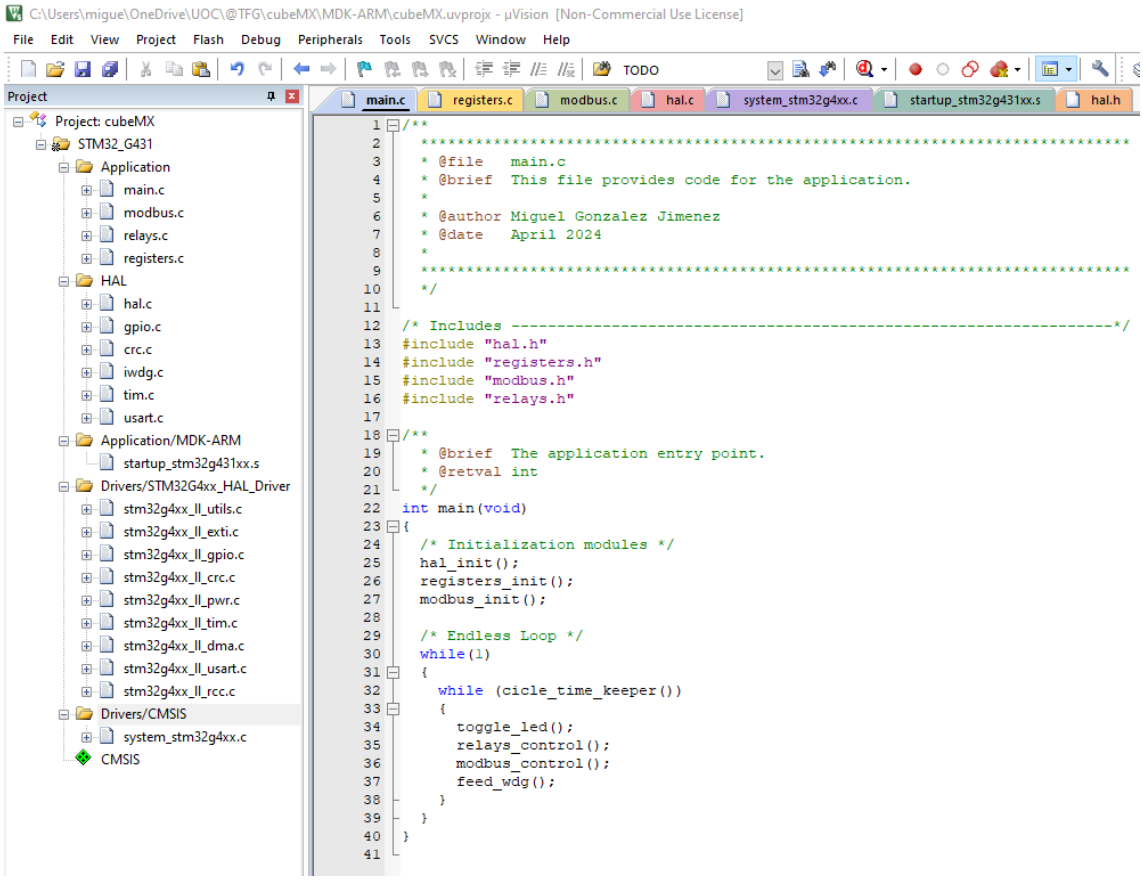


Figura 33: Estructura del proyecto en Keil uVision

En los siguiente parrafos se detalla en contenido de cada bloque:

Bloque Main. En la figura anterior se mostraba la estructura del proyecto dentro del IDE Keil uVision. En la parte derecha de la figura se muestra abierto el contenido del bloque Main, correspondiente al archivo “main.c”.

El código contenido en main.c se divide en, una cabecera de información, un apartado para los diferentes “*#include”* y finalmente la función de entrada al programa. En esta, hay dos partes diferenciadas que corresponden, por un lado, con la inicialización de los módulos de la aplicación y, por otro lado, el bucle infinito que realizará la lógica del programa.

El bucle infinito se ejecutará con una frecuencia específica; en la línea 32 se muestra una función llamada “cicle_time_keeper()” que no permitirá la entrada al bloque *while* que la llama, hasta que haya pasado el tiempo definido y que será el tiempo que determinará la frecuencia de refresco de las funciones.

Con cada ejecución del bucle infinito se realizará un cambio en el estado en el LED, que, aunque no podrá ser visible por el ojo humano, nos brindará información de la frecuencia de ejecución con la ayuda de un osciloscopio. Además de lo anterior, se ejecutarán las llamadas a los bloques Relays y Modbus para que realicen las tareas programadas. Finalmente se refresca el *watchdog*.

Bloque Relays. El bloque desarrollado en el archivo “relays.c” contendrá el código encargado de modificar el estado de los relés en función del valor de los registros “*set*” y “*reset”*, registros que son modificados por Modbus.

Se da prioridad al “*reset”*, esto significa que en caso de ordenes contradictorias al mismo tiempo, será la de “*reset*” la que prevalezca.

Finalmente, se llama a la función “*refresh_all_registers*” que se encargará de poner a 0 los registros de “*set”* y “*reset”* que ya han sido tratados, así como de modificar el valor del registro de lectura que indica el estado actual de los relés.

```c
/**
  ******************************************************************************
  * @file    relays.c
  * @brief   This file provides code for the relays oputputs.
  *
  * @author Miguel Gonzalez Jimenez
  * @date    April 2024
  *
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "relays.h"
#include "registers.h"

/* Private define ------------------------------------------------------------*/
#define REGISTER_LENGTH 16

/* Public code ---------------------------------------------------------------*/
/**
  * @brief relays_control.
  * It performs the set and reset of the relays, with reset having priority.
  * It also manages the updating the relay state variable
  * and clears the set and reset registers.
  */
void relays_control(void)
{
  uint16_t set_register = get_set_register();
  uint16_t reset_register = get_reset_register();

  for(uint8_t i = 0; i < REGISTER_LENGTH; i++)
  {
    if(set_register & (1<<i))
    {
      set_relay(i);
    }
    if(reset_register & (1<<i))
    {
      reset_relay(i);
    }
  }
  refresh_all_registers();
}
```

Figura 34: Contenido del archivo relays.c

Bloque Registers. Este bloque contiene una estructura con los registros que pueden ser leídos o escritos desde Modbus y que son tratados por la aplicación para modificar el estado de los relés. La estructura está definida de forma privada dentro del archivo “registers.c” con el fin de proteger su contenido y que solo pueda ser modificado por las funciones desarrolladas para tal caso dentro de este mismo bloque, publicadas en el correspondiente archivo “registers.h”.

En la estructura "*register_t*" están contenidos los registros "*set*" y "*reset*" que son los encargados de activar o desactivar relés según los bits activos en cada una de las variables, mientras que el registro "*state*" indica el ultimo estado de los relés y se comporta como un registro de solo lectura, por lo que no tendrá efectos sobre los relés si su valor es modificado por Modbus

Todas las variables mencionadas son de 16 bits sin signo y cada uno de los bits corresponde con uno de los relés, siendo el primer relé "*KA1*" el bit menos significativo LSB y el relé "*K16*" el bit más significativo MSB.

```c
1  /**
2    ******************************************************************************
3    * @file   registers.c
4    * @brief  This file provides code for the application registers.
5    *
6    * @author Miguel Gonzalez Jimenez
7    * @date   April 2024
8    *
9    ******************************************************************************
10   */
11
12 /* Includes ------------------------------------------------------------------*/
13 #include "registers.h"
14
15 /* Private define ------------------------------------------------------------*/
16 #define FALSE 0
17 #define TRUE  1
18
19 /* Private typedef -----------------------------------------------------------*/
20 typedef struct
21 {
22   uint16_t set;
23   uint16_t reset;
24   uint16_t state;
25 }register_t;
26
27 /* Private variables ---------------------------------------------------------*/
28 register_t relays;
29
30 /* Public code ---------------------------------------------------------------*/
31 /**
32   * @brief registers_init
33   */
34 void registers_init(void)
35 {
36   relays.set = 0;
37   relays.reset = 0;
38   relays.state = 0;
39 }
40 /**
41   * @brief get_set_register
42   * @retval relay_order.set register value
43   */
44 uint16_t get_set_register(void)
45 {
46   return relays.set;
47 }
48 /**
49   * @brief get_reset_register
50   * @retval relay_order.reset register value
51   */
52 uint16_t get_reset_register(void)
53 {
54   return relays.reset;
```

Figura 35: Extracto del archivo registers.c

3.1.2. Capa HAL

Esta capa es la encargada de abstraer la aplicación, del hardware utilizado. La utilización de la capa HAL es muy recomendable ya que, por un lado, facilita la legibilidad del código y, por otro lado, la portabilidad a otro hardware; a su vez, también es útil disponer de esta capa para poder realizar test unitarios sobre el código de la aplicación.

```c
/**
  ******************************************************************************
  * @file   hal.c
  * @brief  This file provides code for the hardware application layer.
  *
  * @author Miguel Gonzalez Jimenez
  * @date   April 2024
  *
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "hal.h"
#include "iwdg.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"
#include "crc.h"

/* Private variables ---------------------------------------------------------*/
modbus_tx_handler tx_handler;
modbus_rx_handler rx_handler;

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void USART2_IRQHandler(void);

/**
  * @brief HAL_init
  * @retval None
  */
void hal_init(void)
{
  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_SYSCFG);
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_PWR);

  /* System interrupt init*/
  NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_4);

  /* SysTick_IRQn interrupt configuration */
  NVIC_SetPriority(SysTick_IRQn, NVIC_EncodePriority(NVIC_GetPriorityGrouping(),15, 0));

```

Figura 36: Extracto del archivo hal.c

Como se muestra en la figura de arquitectura del software, la capa HAL está situada entre la de aplicación y los drivers de bajo nivel que proporciona el fabricante, es por ello, por lo que en esta capa se realizarán todas las “conexiones” entre la aplicación y los drivers. Lo harán mediante macros, funciones o *callbacks*.

Referente a los *callbacks*, son funciones que se pasan como argumento en un manejador y que son llamadas cuando se produce un evento, como podría ser la interrupción de un periférico. Se utilizará este tipo de mecanismo, por ejemplo, para la recepción y envío de bytes por la UART del Modbus, así como para *timers* que pudieran necesitarse.

Será en el archivo "*hal.c*" donde se capturen las interrupciones de los periféricos al incluir las funciones …IRQ_handler… que se necesiten.

3.2. Implementación Modbus

La implementación del Modbus se ha realizado en base a la especificación proporcionada por modbus.org [28], limitándola a las necesidades del proyecto.

La siguiente figura representa la máquina de estados que se ha diseñado e implementado para la gestión del Modbus. Como puede observarse en la siguiente figura, existen cuatro estados que de detallarán en cada uno de los correspondientes apartados.

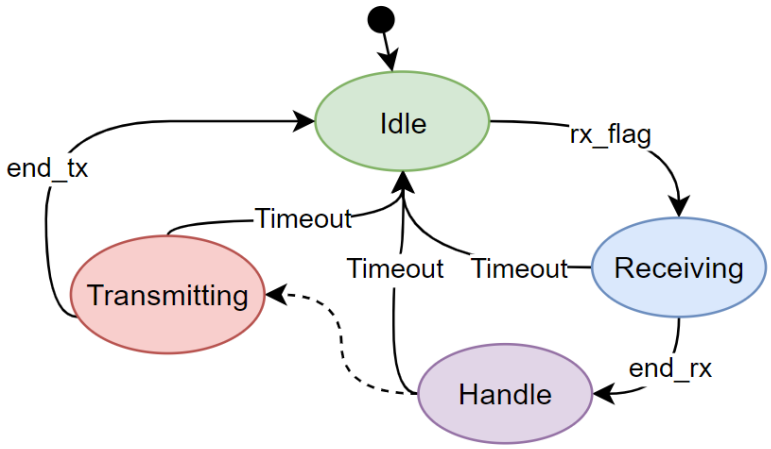


Figura 37: Máquina de estados diseñada para la implementación Modbus

3.2.1. Estado *Idle*.

El estado *Idle* u ocioso en su traducción al castellano, será el punto de entrada a la máquina de estados; permanecerá allí hasta recibir algún byte por el puerto UART, momento en el cual se activa "*rx_flag*", hecho que provoca un cambio al estado *Receiving* de la máquina.

3.2.2. Estado *Receiving.*

La máquina de estados permanecerá en este estado hasta finalizar la recepción de bytes, momento que será señalizado por la variable "*end_rx*". Será considerado el fin de recepción de la trama Modbus pasados más de 3,5 símbolos de tiempo sin recepción; dicha duración se calcula a continuación:

$$tiempo\ tx\ bit = \frac{1}{bps} = \frac{1}{57600} = 17{,}4\ \mu sec \tag{7}$$

$$nbit = bits_{datos} + bit_{start} + bit_{stop} + bit_{paridad} = 8 + 1 + 1 + 1 = 11 \tag{8}$$

$$t_simbolo = tiempo\ tx\ bit * nbits = 17{,}4 * 11 = 191{,}4\ \mu sec \tag{9}$$

$$t_{silencio} = t_{simbolo} * 3{,}5 = 669{,}9\ \mu sec \tag{10}$$

Por simplicidad en la implementación, se llamará cada milisegundo a la máquina de estados, tiempo que corresponde con la máxima granularidad obtenida por el *timer* SysTick, de este modo, si no es recibido ningún símbolo en el tiempo entre llamadas a la máquina, esto es 1 milisegundo, se considerará el fin de la recepción de la trama.

```
385 /**
386  * @brief Checks if the reception of bytes has finished.
387  * Working at 57600 baud, we will achieve a silence of 3.5 bytes after 669 microseconds,
388  * therefore this function will be called every millisecond.
389  */
390 void check_end_of_rx(void)
391 {
392   if(fsm.rx_flag == UP)
393   {
394     fsm.rx_flag = DOWN;
395   }
396   else
397   {
398     fsm.end_rx = TRUE;
399   }
400 }
```

Figura 38: Función check_end_of_rx

3.2.3. Estado *Handle.*

Una vez finalizada la recepción de una trama, el mensaje se comprobará, procesará y finalmente se preparará una respuesta si fuera necesario.

La función que procesará el mensaje realizará los siguientes pasos:

1. Comprobará que el mensaje va dirigido a su número de esclavo en el bus.
2. Comprobará la integridad del mensaje mediante la comprobación del CRC.
3. Comprobará que la función a realizar sea alguna de las implementadas.

A continuación, se muestra el código que realiza la función de manejo del mensaje:

```
77 /**
78  * @brief Handles Modbus messages after complete reception.
79  * Checks message integrity, determines function code, and dispatches to appropriate handler.
80  */
81 void handle_message(void)
82 {
83   uint8_t slave = fsm.rx_buffer[0] ;
84   uint8_t function_code;
85
86   /* Check if the message is for this slave and if CRC is correct */
87   if(fsm.rx_buffer[0] == fsm.slave_id && check_crc())
88   {
89     function_code = fsm.rx_buffer[1];
90     switch (function_code)
91     {
92       case READ_HOLDING_REG:  /* Function code for reading holding registers */
93         handle_read_holding_registers();
94         break;
95       case WRITE_SIGLE_REG:  /* Function code for writing a single register */
96         handle_write_single_register();
97         break;
98       case WRITE_MULTIPLE_REGS:  /* Function code for writing multiple registers */
99         handle_write_multiple_registers();
100        break;
101      default:  /* Unsupported function */
102        send_error_response(function_code, ILEGAL_FUNCTION);
103        break;
104    }
105  }
106  else /* No valid data to send as response */
107  {
108    fsm.bytes_to_send = 0;
109  }
110  /* Reset received bytes count and clear receive buffer */
111  fsm.received_bytes = 0;
112  clear_rx_buffer();
113 }
```

Figura 39: Función handle_message

Función handle_read_holding_registers. En esta función se procesarán los mensajes de lectura de registros; realizará las siguientes operaciones:

1. Extraerá la dirección del primer registro y el número de registros.
2. Comprobará que el último de los registros requerido esté dentro de los registros declarados.
3. Preparará la respuesta con el valor de los registros consultados, añadiéndole el CRC.
4. Calculará el número de bytes a transmitir para que el próximo estado gestione el envío.

A continuación, se muestra la función implementada:

```c
/**
 * @brief Handles the Read Holding Registers Modbus function (0x03).
 * This function processes a request to read one or more holding registers.
 * It validates the request, constructs the response, and prepares it for transmission.
 */
void handle_read_holding_registers(void)
{
  /* Extract the starting address and number of registers requested from the received buffer */
  uint16_t start_address = (fsm.rx_buffer[2] << 8) | fsm.rx_buffer[3];
  uint16_t num_registers = (fsm.rx_buffer[4] << 8) | fsm.rx_buffer[5];
  uint8_t i;

  /* Check if the requested registers are within the valid range */
  if (start_address + num_registers > MAX_REGISTERS)
  {
    send_error_response(fsm.rx_buffer[1], ILEGAL_DATA_ADDRESS);
    return;
  }

  /* Prepare the response buffer */
  fsm.tx_buffer[0] = fsm.slave_id;
  fsm.tx_buffer[1] = READ_HOLDING_REG;
  fsm.tx_buffer[2] = num_registers * 2;

  /* Fill the requested registers into the transmission buffer */
  for (i = 0; i < num_registers; i++)
  {
    uint16_t reg_value = read_holding_register(start_address + i);
    fsm.tx_buffer[3 + 2 * i] = (reg_value >> 8) & 0xFF;
    fsm.tx_buffer[4 + 2 * i] = reg_value & 0xFF;
  }

  /* Calculate the CRC for the response */
  uint16_t crc = modbus_crc_calculate(fsm.tx_buffer, 3 + 2 * num_registers);
  fsm.tx_buffer[3 + 2 * num_registers] = crc & 0xFF;        /* CRC low byte */
  fsm.tx_buffer[4 + 2 * num_registers] = (crc >> 8) & 0xFF; /* CRC high byte */

  /* Set the length of the response to send */
  fsm.bytes_to_send = 5 + 2 * num_registers;
  fsm.tx_i = 0;  /* Reset the transmission index */
}
```

Figura 40: Función handle_read_holding_registers

Función handle_write_single_register. En esta función se procesarán los mensajes de escritura de un registro. Realizará las siguientes operaciones:

1. Extraerá la dirección del registro.
2. Comprobará que el registro requerido esté dentro de los registros declarados.
3. Escribirá el valor en el registro solicitado.
4. Preparará la respuesta añadiéndole el CRC.
5. Calculará el número de bytes a transmitir para que el próximo estado gestione el envío.

A continuación, se muestra la función implementada:

```c
/**
 * @brief Handles the Write Single Register Modbus function (0x06).
 * This function processes a request to write a single holding register.
 */
void handle_write_single_register(void)
{
  /* Extract the register address and value to write from the received buffer */
  uint16_t register_address = (fsm.rx_buffer[2] << 8) | fsm.rx_buffer[3];
  uint16_t register_value = (fsm.rx_buffer[4] << 8) | fsm.rx_buffer[5];

  /* Check if the register address is within the valid range */
  if (register_address > MAX_REGISTERS)
  {
    send_error_response(fsm.rx_buffer[1], ILEGAL_DATA_ADDRESS);
    return;
  }

  /* Write the value to the register */
  if (set_register_by_position(register_address, register_value))
  {
    /* If the write operation was successful, prepare the response that echoes the request */
    fsm.tx_buffer[0] = fsm.slave_id;
    fsm.tx_buffer[1] = WRITE_SIGLE_REG;
    fsm.tx_buffer[2] = fsm.rx_buffer[2];
    fsm.tx_buffer[3] = fsm.rx_buffer[3];
    fsm.tx_buffer[4] = fsm.rx_buffer[4];
    fsm.tx_buffer[5] = fsm.rx_buffer[5];

    /* Calculate the CRC of the response */
    uint16_t crc = modbus_crc_calculate(fsm.tx_buffer, 6);
    fsm.tx_buffer[6] = crc & 0xFF;        /* CRC low byte */
    fsm.tx_buffer[7] = (crc >> 8) & 0xFF; /* CRC high byte */

    /* Set the length of the response to send */
    fsm.bytes_to_send = 8;
    fsm.tx_i = 0;  /* Reset the transmission index */
  }
  else/* If the write operation fails for any reason, send an appropriate error code */
  {
    send_error_response(fsm.rx_buffer[1], WRITE_ERRROR);
  }
}
```

Figura 41: Función handle_write_single_register

Función handle_write_multiple_registers. De forma análoga a las anteriores funciones, en este caso se realizarán comprobaciones para no escribir fuera de la estructura de datos de los registros y se modificarán los valores de los registros requeridos, preparando finalmente la trama de respuesta.

Más concretamente:

1. Extraerá la dirección del primer registro y el número de registros.
2. Comprobará que el último de los registros requerido esté dentro de los registros declarados.
3. Escribirá el valor en los registros solicitados.
4. Preparará la respuesta añadiéndole el CRC.
5. Calculará el número de bytes a transmitir para que el próximo estado gestione el envío.

A continuación, se muestra la función implementada en el código:

```c
/**
 * @brief Handles the Write Multiple Registers Modbus function (0x10).
 * This function processes a request to write values to multiple holding registers.
 */
void handle_write_multiple_registers(void)
{
  /* Extract the starting address and the number of registers to write from the received buffer */
  uint16_t start_address = (fsm.rx_buffer[2] << 8) | fsm.rx_buffer[3];
  uint16_t num_registers = (fsm.rx_buffer[4] << 8) | fsm.rx_buffer[5];
  uint8_t byte_count = fsm.rx_buffer[6];
  uint8_t i;

  /* If the request is out of range, send an error response */
  if (start_address + num_registers > MAX_REGISTERS)
  {
    send_error_response(fsm.rx_buffer[1], ILEGAL_DATA_ADDRESS);
    return;
  }

  for (i = 0; i < num_registers; i++)
  {
    uint16_t value = (fsm.rx_buffer[7 + 2 * i] << 8) | fsm.rx_buffer[8 + 2 * i];
    if (!write_holding_register(start_address + i, value))
    {
      /* If a write operation fails, send an error response and terminate the loop */
      send_error_response(fsm.rx_buffer[1], WRITE_ERRROR);
      return;
    }
  }

  /* If all writes are successful, prepare the acknowledgment response that echoes the request */
  fsm.tx_buffer[0] = fsm.slave_id;
  fsm.tx_buffer[1] = WRITE_MULTIPLE_REGS;
  fsm.tx_buffer[2] = (start_address >> 8) & 0xFF;
  fsm.tx_buffer[3] = start_address & 0xFF;
  fsm.tx_buffer[4] = (num_registers >> 8) & 0xFF;
  fsm.tx_buffer[5] = num_registers & 0xFF;

  /* Calculate and append CRC */
  uint16_t crc = modbus_crc_calculate(fsm.tx_buffer, 6);
  fsm.tx_buffer[6] = crc & 0xFF;        /* CRC low byte */
  fsm.tx_buffer[7] = (crc >> 8) & 0xFF; /* CRC high byte */

  /* Set the length of the response to send */
  fsm.bytes_to_send = 8;
  fsm.tx_i = 0;  /* Reset the transmission index */
}
```

Figura 42: Función handle_write_multiple_registers

Manejo de errores. Una vez mostradas las funciones implementadas para el procesado de los mensajes Modbus, se hace necesario contemplar los casos de excepción; tal como se especifica en el protocolo Modbus, hay definidos unos mensajes de error que serán devueltos al *Master* del bus en caso de excepción.

El formato del mensaje de error estará formado por los siguientes campos:

1. *Slave ID*. Número de esclavo al que va dirigido el mensaje.
2. Función. En este campo se añadirá al valor de la función solicitada por el *Master* el valor 0x80.
3. Código de error. Se introduce el código de error detectado; habiéndose implementado los siguientes:
    - 0x01. Función ilegal. En este caso, el *Master* ha intentado realizar una función que no ha sido implementada en este dispositivo.
    - 0x02. Dirección de datos ilegal. En este caso, el *Master* ha intentado leer o escribir fuera del espacio de registros.
    - 0x04. Error de escritura. En este caso, el módulo de registros no ha podido hacer la escritura del registro. Hay que destacar que tal y como se ha realizado la implementación, este caso no podrá darse salvo error en la codificación.
4. CRC. Como toda trama de Modbus, deberá incorporar dos bytes con el CRC calculado.

La implementación de la función se realiza de la siguiente forma:

```c
/**
 * @brief Sends an error response with a given function code and error code.
 * This function constructs an error response message according to the Modbus protocol.
 * @param function The function code from the request that caused the error.
 * @param error_code The specific Modbus exception code to be returned.
 */
void send_error_response(uint8_t function, uint8_t error_code)
{
  fsm.tx_buffer[0] = fsm.slave_id;
  fsm.tx_buffer[1] = function | MODBUS_ERROR;
  fsm.tx_buffer[2] = error_code;

  /* Calculate CRC for the error response */
  uint16_t crc = modbus_crc_calculate(fsm.tx_buffer, 3);
  fsm.tx_buffer[3] = crc & 0xFF;        /* CRC low byte */
  fsm.tx_buffer[4] = (crc >> 8) & 0xFF; /* CRC high byte */

  /* Set the length of the response to be sent, including the CRC */
  fsm.bytes_to_send = 5;
  fsm.tx_i = 0;   /* Reset the transmission index */
}
```

Figura 43: Función send_error_response

3.2.4. Estado *Transmitting.*

Finalmente, el estado *Transmitting* completará la operación solicitada por Modbus. Su finalidad será la de iniciar la transmisión enviando el primer byte, iniciando de esta forma el proceso de envío que irá transmitiendo un byte tras otro mediante las interrupciones generadas por la UART.

Una vez se detecta que el mensaje se ha enviado completamente se activará el *flag* "*end_tx*" para señalizar el fin del envió a la máquina de estados.

Se debe tener en cuenta que, en el caso de no necesitar transmitir una respuesta, se tendría un tamaño de bytes para enviar igual a 0 y se activará el *flag* "*end_tx*" para señalizar este hecho a la máquina de estados. A continuación, se muestra la implementación.

```c
/**
 * @brief Handles the transmission process.
 * Initiates the sending of response bytes and handles transmission completion.
 */
void transmitting_handle(void)
{
  if (fsm.bytes_to_send > 0 && fsm.tx_i == 0)
  {
    /* Start transmitting the first byte if there are bytes to send */
    tx_byte(fsm.tx_buffer[fsm.tx_i++]);
    fsm.bytes_to_send--;
  }

  if (fsm.bytes_to_send == 0)
  {
    /* Reset index for future transmissions and signal that the transmission has ended */
    fsm.tx_i = 0;
    fsm.end_tx = TRUE;
  }
}
```

Figura 44: Función transmitting_handle

4. Resultados

4.1. Montaje del prototipo

Recibidos los componentes electrónicos, se realizó el montaje de estos sobre el PCB. En primer lugar, se comenzó por el montaje del microcontrolador por ser el componente con mayor dificultar debido a su gran número de pines y reducido espaciado entre ellos.

Una vez montado el microcontrolador, se continuó con los componentes SMT y posteriormente con los componentes THT.

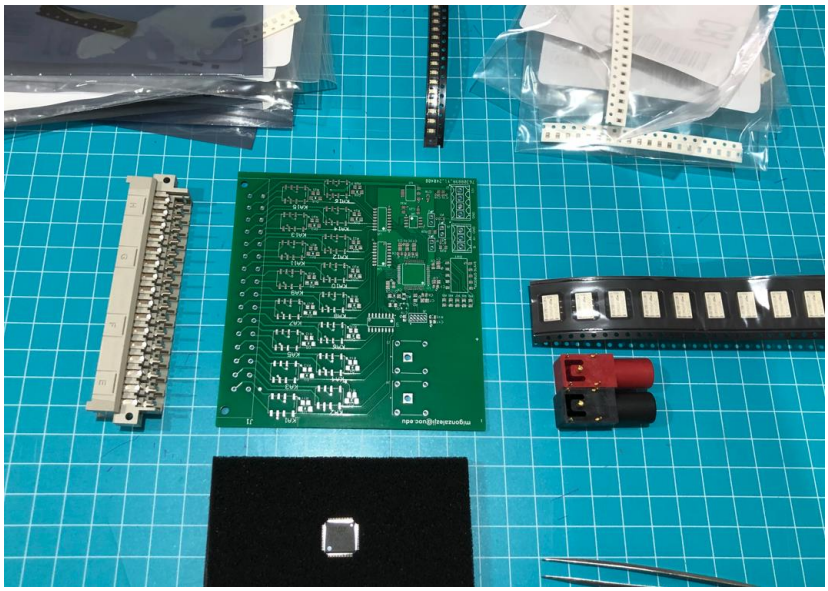


Figura 45: PCB y componentes

Finalizado el montaje, se obtiene el PCBA que se muestra en la siguiente figura.

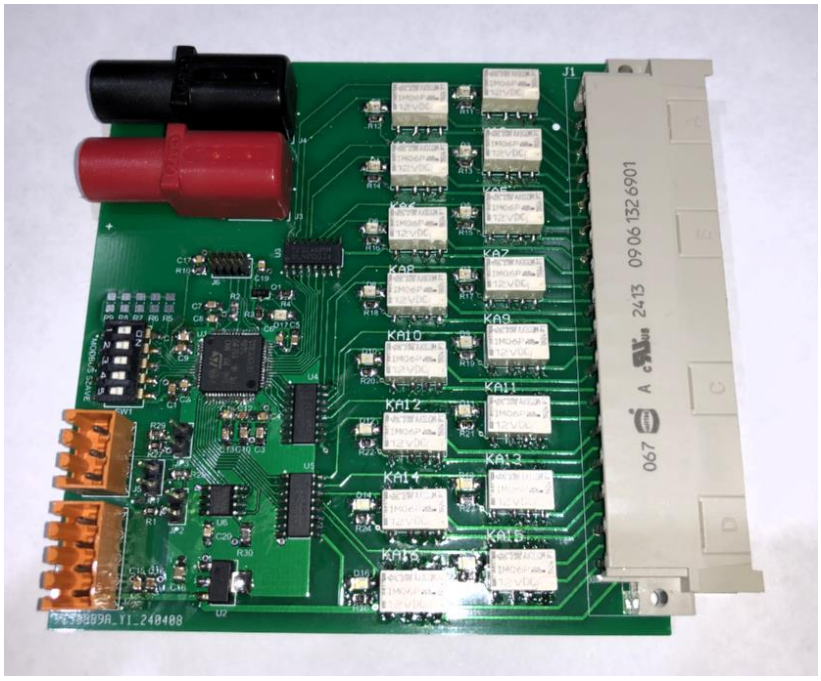


Figura 46: PCBA

4.2. Verificación del prototipo

Para verificar el correcto diseño y montaje del prototipo se realizan las siguientes comprobaciones que se detallan en los próximos párrafos.

Activación de todos los relés. En este caso se activan los relés modificando a mano el valor del registro "*set*" a 0xffff; el nuevo valor se ve reflejado en el registro "*state*".

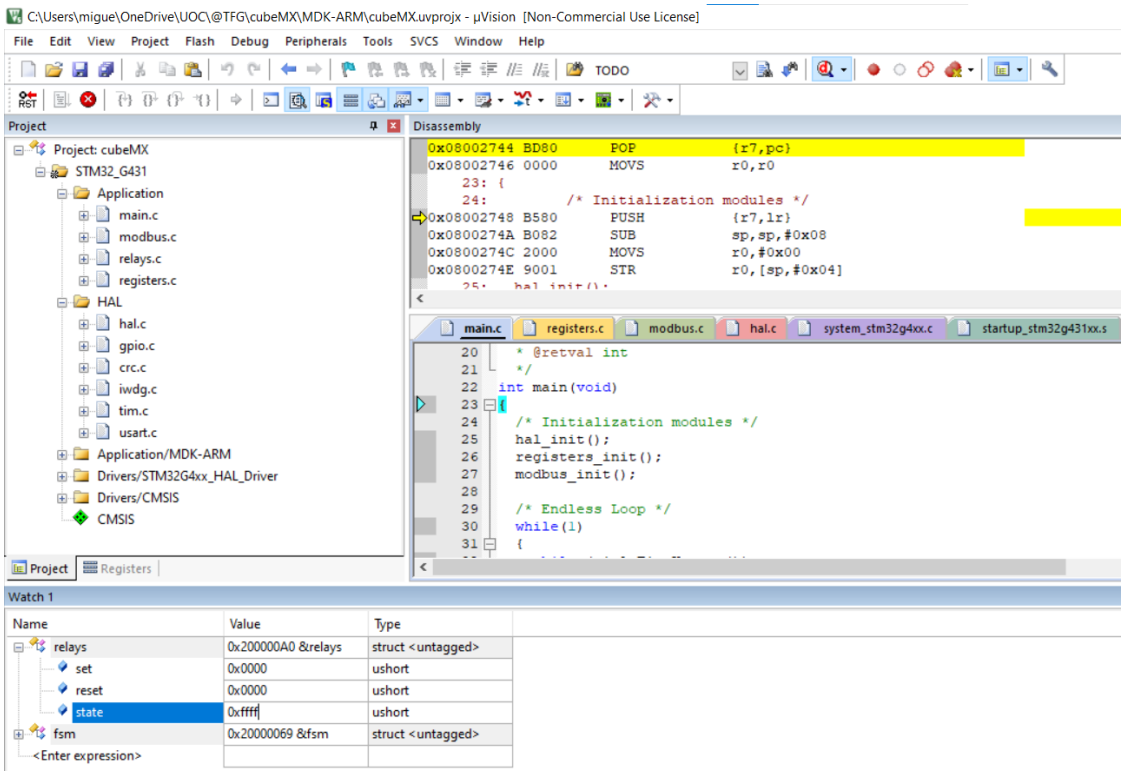


Figura 47: IDE keil uVision en modo depuración

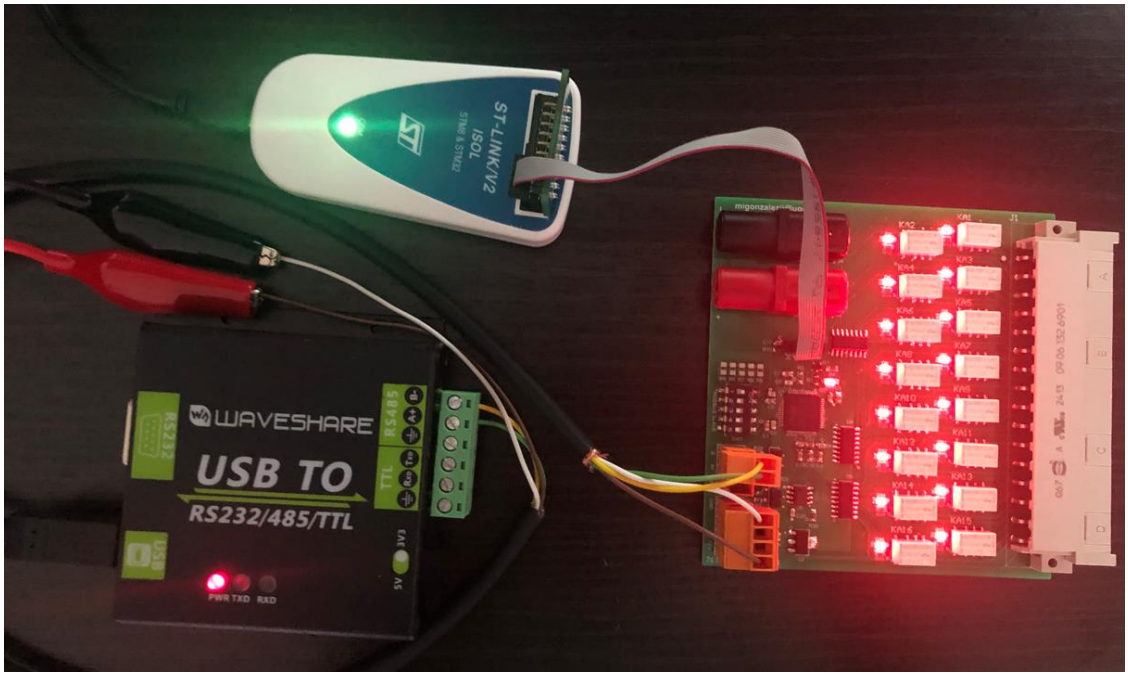


Figura 48: Activación de todos los relés

Continuidad entre conectores al activar los relés. En este caso, se trata de activar los relés individualmente y comprobar la continuidad en cada uno de los dos contactos.

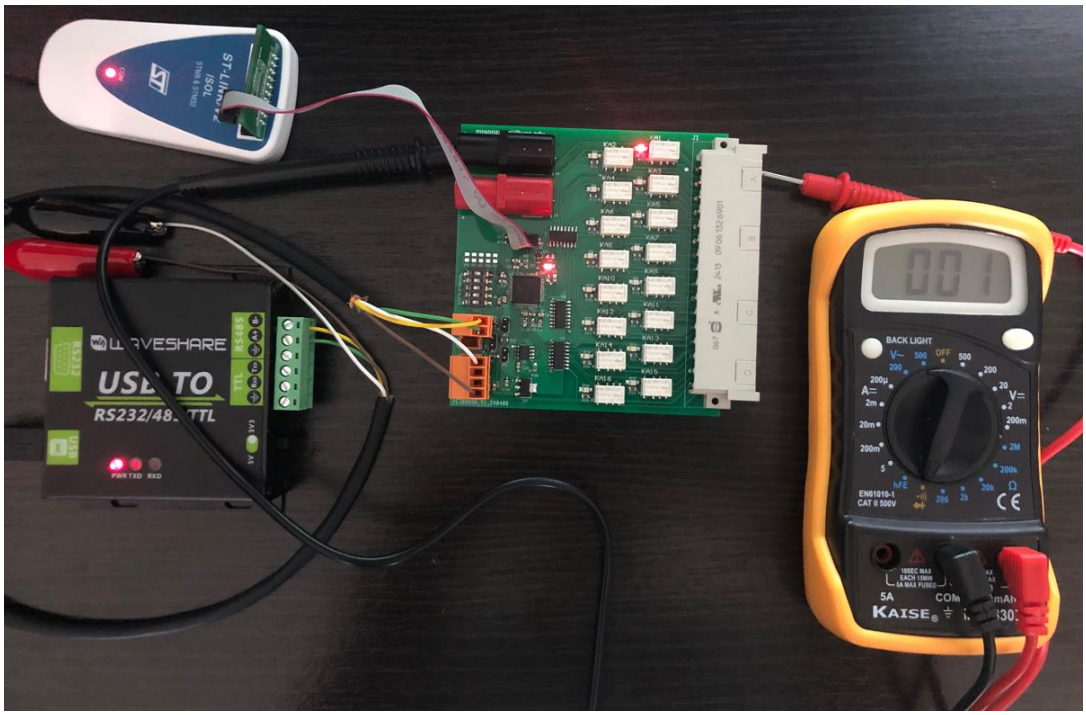


Figura 49: Comprobación de continuidad

Comprobación de comunicación Modbus. En este caso, al comenzar la comunicación por Modbus verificamos implícitamente la correcta configuración del ID y configuración UART, entre otros. Como se puede observar en la figura inferior, se realiza la lectura de los tres registros de forma simultánea y se escribe un valor 0x0230 en el registro "*set*" (registro 0), lo que provoca la activación de los relés 5, 6 y 10, como puede comprobarse en la lectura del registro "*state*":

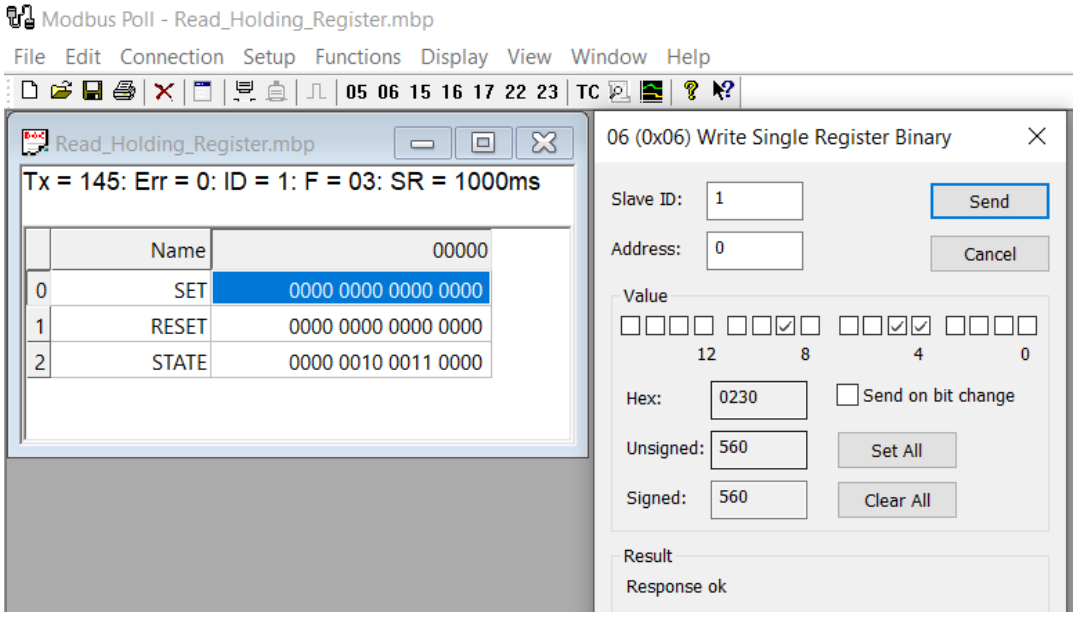


Figura 50: Comunicación Modbus

Se comprueba que efectivamente los relés seteados mediante el registro Modbus correspondiente, son activados:

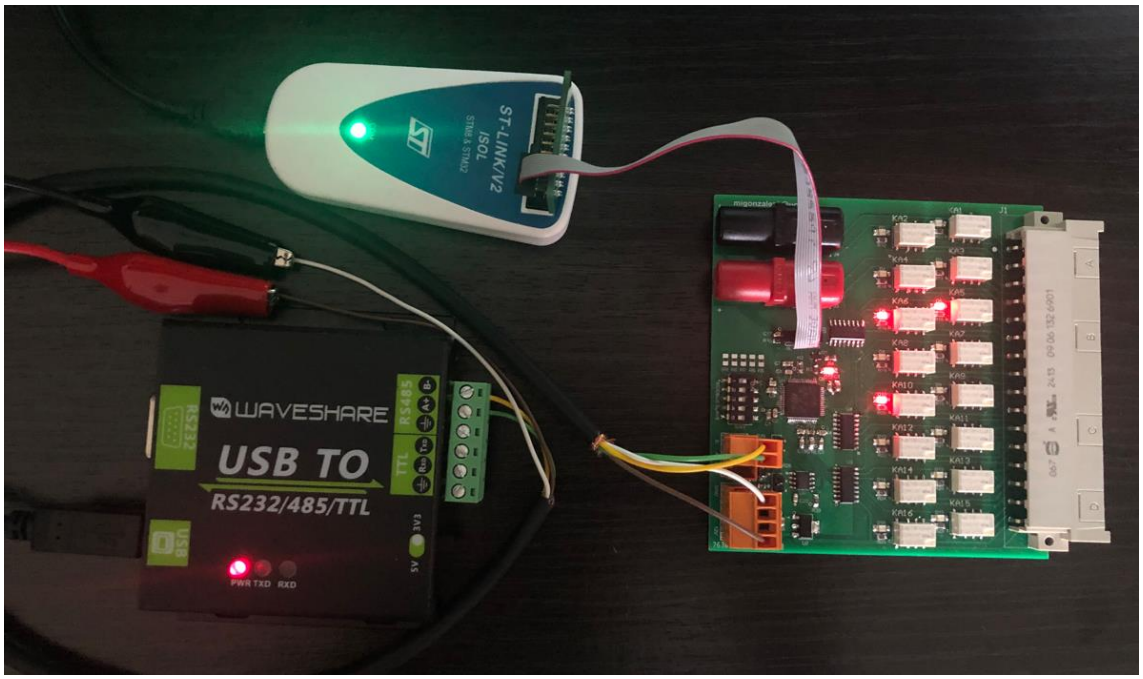


Figura 51: Modbus, activación relés 5, 6 y 10

<u>Comprobación del registro *reset*</u>. En la comprobación anterior se activaron algunos relés y en este caso, de forma análoga se desactivarán algunos de ellos.

Se comprueba que efectivamente los relés reseteados mediante el registro Modbus correspondiente, son desactivados en el circuito.

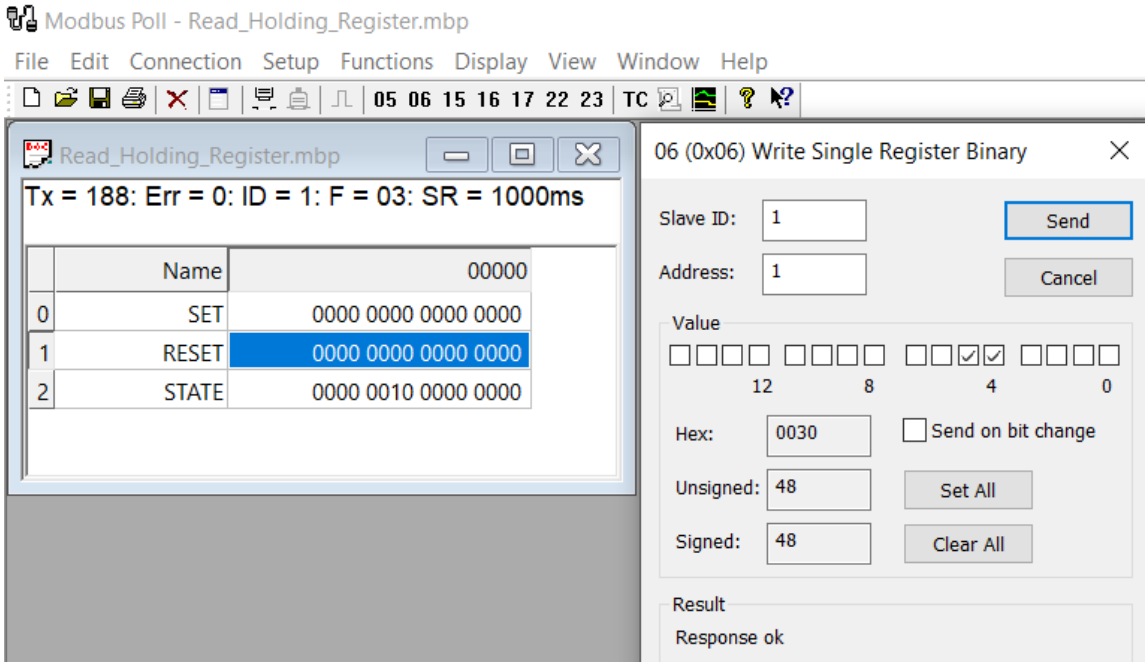


Figura 52: Modbus, Write Single Register reset

Escritura de múltiples registros. En este caso, se desea comprobar la función Modbus Write Multiple Registers y para ello se escribirán ordenes de "*set*" y "*reset*" de forma simultánea:

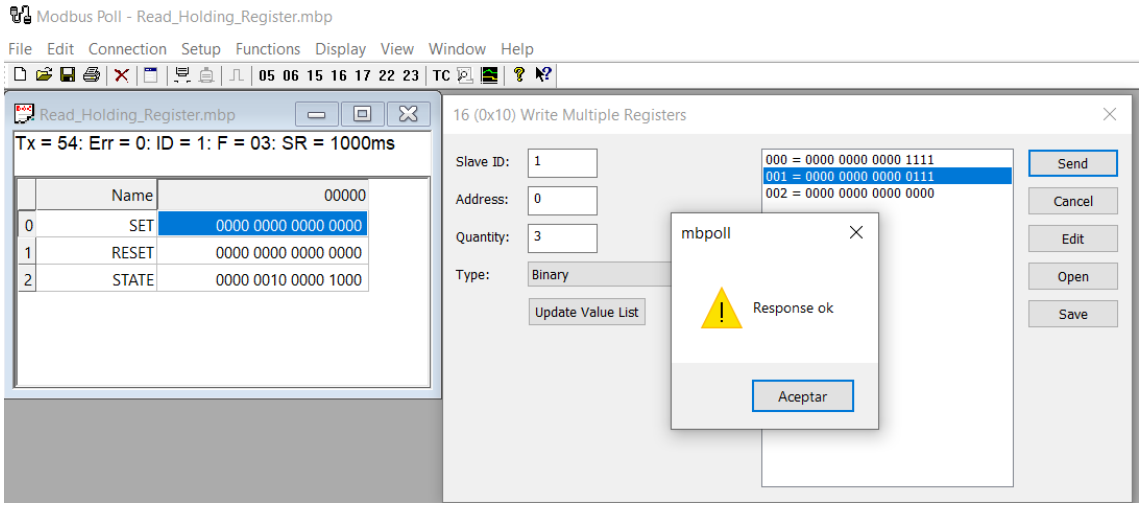


Figura 53: Modbus, Write Multiple Registers

Escritura del registro *state*. En esta comprobación se pretende comprobar que efectivamente un cambio sobre este registro no tendrá ningún efecto en el estado real de los relés:

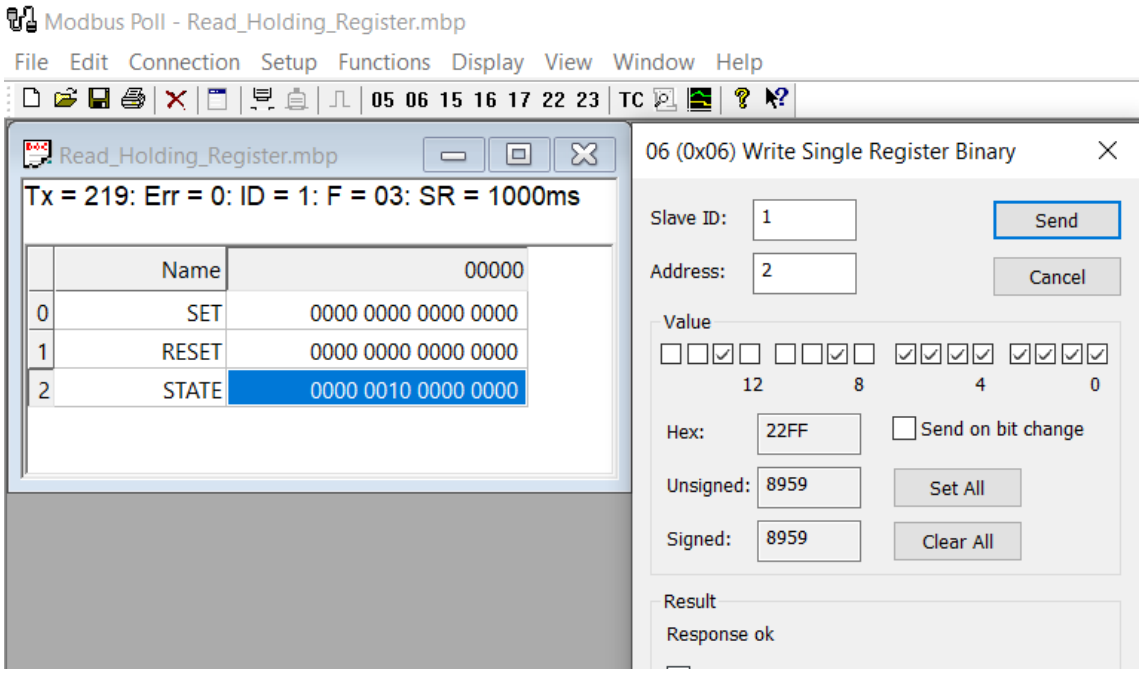


Figura 54: Modbus, escritura registro *state*

Error 01, Illegal Function. Esta prueba consiste en realizar una de las operaciones que no han sido implementadas en el circuito, como lo es "Read Input Registers", esperando que la respuesta Modbus indique esta situación:

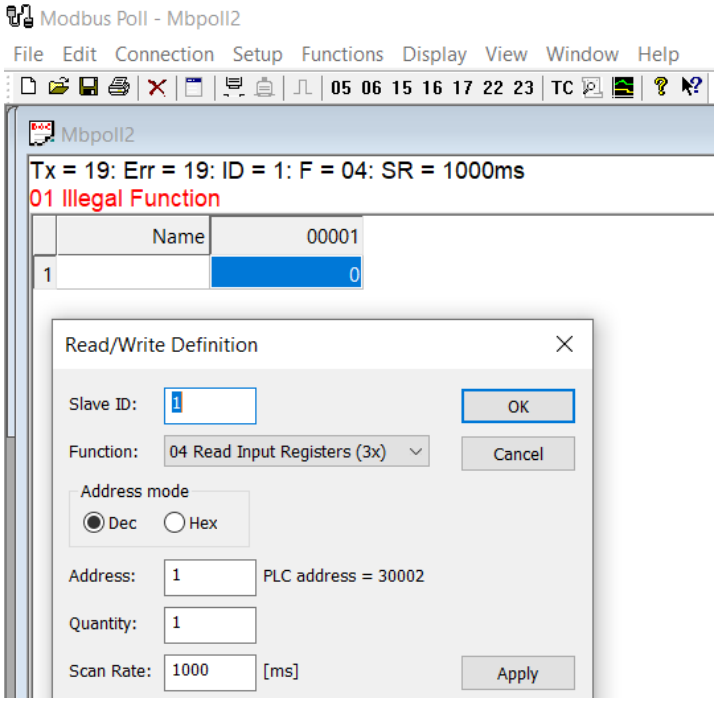


Figura 55: Modbus, intento de función ilegal

Error 02, Illegal Data Address. La idea de esta prueba es comprobar que el firmware detecta correctamente el intento de lectura de un registro fuera del mapa de registros, para ello se intenta tanto intentar leer 4 registros desde el registro 0, como directamente el registro 4:

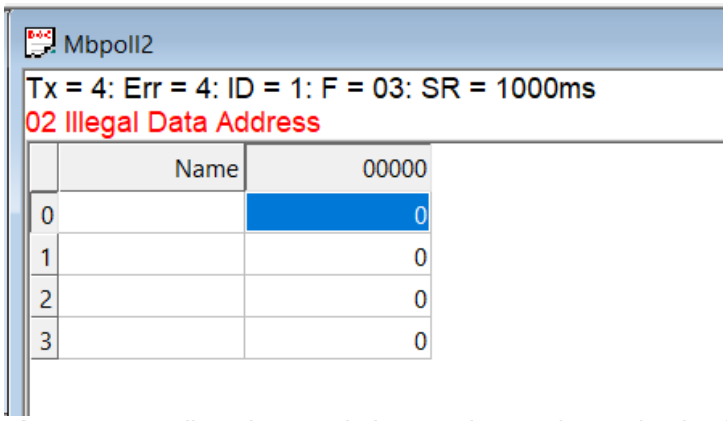


Figura 56: Modbus, intento de lectura de 4 registros desde el registro 0

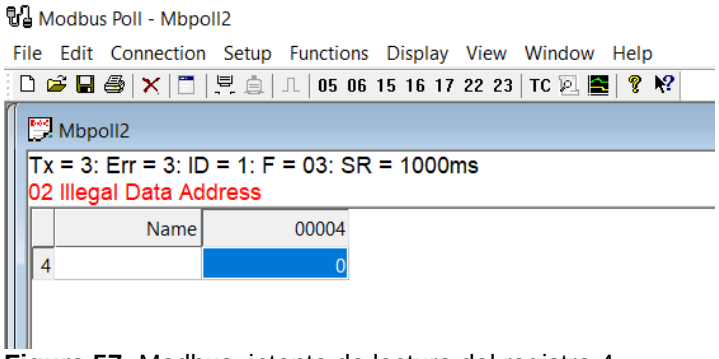


Figura 57: Modbus, intento de lectura del registro 4

Tasa de error. Esta comprobación consiste en aumentar la frecuencia de consultas Modbus, consultando cada 10 milisegundos para aumentar la tasa de envío y tras pasar los 10 minutos comunicando -que permite el software Modbus Poll en su periodo de evaluación-, se consiguen enviar 9554 paquetes sin ningún error:

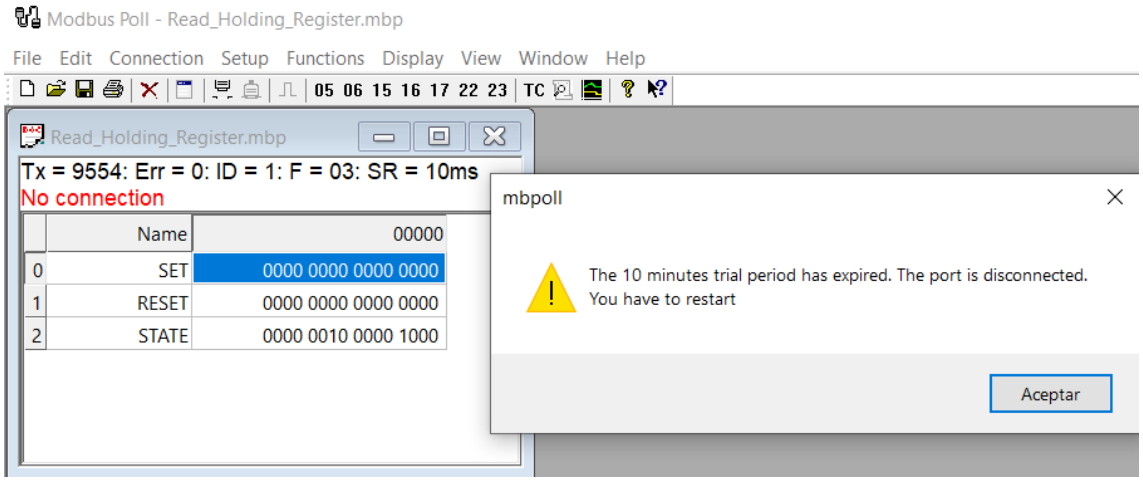


Figura 58: Modbus, 10 minutos de conexión

5. Conclusiones y trabajos futuros

5.1. Cumplimiento de los objetivos iniciales

Siguiendo la planificación detallada en el apartado correspondiente, se han conseguido los objetivos marcados en cada hito y que fueron detallados en la redacción de las correspondientes PEC a modo de entrega parcial del proyecto. Estas PEC concluyeron de la siguiente forma:

- PEC2. Con el diseño del PCB finalizado y la creación de un proyecto *firmware* base.
- PEC3. Con el aprovisionamiento de los materiales, fabricación del prototipo, diseño del programa de control e implementación de Modbus y verificación del prototipo **con resultado exitoso**.
- PEC4. Con la documentación finalizada.

En la evaluación de incidencias y riesgos se consideró la posibilidad de necesitar retrabajar o rediseñar el prototipo realizado en la entrega de la **PEC3**, sin embargo, el diseño ha podido desarrollarse según estaba contemplado en el plan inicial, en parte gracias a las acciones tomadas, fruto del análisis de riesgos, como han sido, anticipar las tareas, o crear la base del proyecto *firmware* antes de dar por finalizado el diseño del PCB con la entrega de la **PEC2**.

Con el hecho de converger cada una de las acciones realizadas, como son, el diseño del hardware, el diseño del software, así como el resto de las operaciones, entre ellas, el aprovisionamiento de materiales y montaje del prototipo, se consigue obtener un circuito **totalmente funcional que satisface todos los objetivos marcados**.

5.2. Líneas de trabajo futuras

Durante el desarrollo y diseño del hardware y software han ido apareciendo posibles mejoras a implementar de cara a ampliar las aplicaciones en las que el circuito puede ser utilizado. De detallan a continuación:

Mayor rango de alimentación. Pese a que el circuito desarrollado tiene prevista su alimentación a través de una línea de 12 V, hubiera sido muy interesante de cara a ser utilizando en otras aplicaciones, disponer de un mayor rango, como podría ser de 12 V a 24 V. Ello obligaría posiblemente a realizar dos cambios importantes, por un lado, integrar un convertidor DC-DC, y por otro lado, muy posiblemente convendría bajar la tensión de alimentación de los relés a 5 V, para asegurar su estabilidad mediante el árbol de alimentaciones que cuelguen del convertidor.

Velocidad del bus configurable. El circuito se ha diseñado para trabajar con un bus a 57.600 *baudios*, sin embargo, para ser utilizado a otra velocidad requeriría cambiar el *firmware;* una solución para dar mayor flexibilidad podría ser tener una lista de velocidades, las que son típicamente estándar en los puertos de comunicación, configurables mediante Dip Switch.

Mayor rango de esclavo en el bus. El circuito actual dispone de 32 identificadores de esclavo diferentes; al igual que los puntos anteriores de mejora, si se desea que el circuito pueda ser utilizado más allá de una plataforma de test *Offline,* podría convenir aumentar hasta 256, esto se podría realizar modificando el Dip Switch aumentándolo hasta los 8 bits.

Taladros de sujeción. El circuito se ha diseñado para ser fijado mediante *brackets* añadidos al conector. Podría interesar añadirle unos taladros para la fijación mediante separador con tornillo o similar.

6. Glosario

3D: Tres dimensiones.

AOI (Automated Optical Inspection): Sistema de inspección que utiliza cámaras para detectar defectos en productos electrónicos durante su fabricación.

ARM: Arquitectura de microprocesadores utilizada ampliamente en dispositivos móviles y embebidos.

Array: Matriz de componentes.

CRC (Cyclic Redundancy Check): Método de detección de errores en la trama de comunicación mediante la generación de una secuencia de verificación.

Baudios: Unidad de medida de la velocidad de transmisión de símbolos en una señal de comunicación, equivalente a un símbolo por segundo.

BOM (Bill of Materials): Lista detallada de los materiales, componentes y cantidades necesarias para fabricar un producto.

Bottom: Contraparte del top, se refiere a la capa inferior o base del PCB.

Buck: Tipo de convertidor DC-DC que reduce el voltaje de entrada a un voltaje de salida más bajo.

Build to Print: Método de fabricación que persigue la máxima semejanza entre unidades.

Compilación: Proceso de transformar el código fuente de un programa en código objeto o ejecutable.

Datasheet: Documento técnico que proporciona especificaciones y detalles sobre un componente electrónico.

Diagrama de Gantt: Herramienta grafica para representar visualmente un cronograma.

Dip Switch: Conjunto de interruptores diminutos usados para configurar ajustes en dispositivos electrónicos.

Drivers: Software que permite al sistema operativo interactuar con un hardware específico.

DUT (Device Under Test): Dispositivo o componente electrónico que está siendo probado para verificar su funcionamiento.

EOL (End of Life): Término utilizado para indicar cuando un producto o componente ya no se fabricará ni soportará.

ESD (Electrostatic Discharge): Descarga electrostática, fenómeno que puede dañar componentes electrónicos.

Esquemático: Diagrama que representa los componentes y conexiones en un circuito electrónico.

FCT (Functional Circuit Test): Prueba que verifica que un PCB o dispositivo electrónico funcione correctamente bajo condiciones operativas normales.

FDCAN (Flexible Data-rate Controller Area Network): Versión mejorada del protocolo CAN que permite mayores tasas de transmisión de datos.

Firmware: Programa de software que controla las funciones de hardware de un dispositivo específico.

Fixture: Dispositivo utilizado para sujetar, soportar o posicionar un objeto durante procesos de fabricación o prueba.

Flag: Indicador utilizado en programación y sistemas digitales para señalar una condición o estado específico.

Footprint: Huella o patrón de pines de un componente electrónico en un PCB.

Gerber: Formato estándar de archivos utilizado en la fabricación de placas de circuito impreso (PCBs).

GND (Ground): Referencia de voltaje cero en un circuito eléctrico, comúnmente conocida como tierra.

Grid: Cuadrícula usada para organizar visualmente elementos en una interfaz o diseño.

ICT (In-Circuit Test): Método de prueba que verifica la funcionalidad de componentes electrónicos ensamblados en un PCB.

IDE (Integrated Development Environment): Ambiente de desarrollo integrado que proporciona herramientas para la programación.

Idle: Estado de inactividad o espera en un sistema o dispositivo.

ITA (Interface Test Adapter): Adaptador que conecta un dispositivo bajo prueba (DUT) a un sistema de prueba (Receiver).

LL Drivers (Low Level Drivers): Interfaz software de bajo nivel para interactuar con el hardware.

Jumper: Puente o conector que se usa para cerrar circuitos eléctricos temporalmente.

Keil: Empresa que desarrolla software de desarrollo para microcontroladores, conocida por su IDE Keil uVision.

Label: Etiqueta o identificador usado en programación para nombrar partes de un código o componentes en un diseño.

LDO (Low Dropout Regulator): Regulador de voltaje que puede operar con una diferencia muy pequeña entre el voltaje de entrada y de salida.

LED (Light Emitting Diode): Diodo que emite luz cuando se le aplica voltaje.

LSB (Least Significant Bit): Bit menos significativo en un número binario, que representa el valor más bajo.

Macros: Instrucciones para el preprocesador en lenguaje C.

Mass interconexion: Sistema de interconexión utilizado en pruebas de ensamblaje modular y automatizado.

Master: Dispositivo que controla la operación de otros dispositivos en un sistema de comunicación.

Mbps (Megabits per second): Unidad de medida de la velocidad de transferencia de datos, equivalente a un millón de bits por segundo.

Modbus: Protocolo de comunicación serial utilizado en sistemas de automatización industrial.

MSB (Most Significant Bit): Bit más significativo en un número binario, que representa el valor más alto.

Offline: Fuera de línea.

Online: En línea.

PCB (Printed Circuit Board): Tarjeta utilizada para conectar eléctricamente componentes electrónicos.

PCBA (Printed Circuit Board Assembly): Tarjeta de circuito impreso montada con los componentes electrónicos.

PDF (Portable Document Format): Formato de archivo utilizado para presentar documentos de manera consistente en diferentes dispositivos y plataformas.

PEC. Prueba de evaluación continua.

Receiver: En un sistema de test, bloque de conexionado en el lado de la instrumentación.

Renderizar: Proceso de generar una imagen a partir de un modelo por medio de un software computacional.

ROHS (Restriction of Hazardous Substances): Regulación que limita el uso de ciertas sustancias peligrosas en dispositivos electrónicos.

RS485: Bus estándar de comunicación serial.

$R_{\theta JA}$ (Junction-to-ambient thermal resistance): Resistencia térmica desde la unión del transistor hasta el ambiente circundante.

SMT (Surface-Mount Technology): Tecnología de montaje de componentes electrónicos en superficie.

Socket: Conector soldado sobre PCB.

Software: Conjunto de programas y sistemas operativos que permiten realizar tareas específicas en un dispositivo.

SOT-223: Paquete estándar para transistores y otros componentes electrónicos, reconocido por su pequeña forma y eficiencia térmica.

SPI (Solder Paste Inspection): Método de inspección utilizado para verificar la correcta aplicación de la pasta de soldadura en PCBs.

STM32CubeMX: Software del fabricante STMicroelectronics para la generación de código para microcontroladores STM32.

SysTick (System Timer): Temporizador de systema integrado en la familia de microcontroladores ARM Cortex-M

SWD (Serial Wire Debug): Interfaz de depuración que proporciona una manera de comunicarse con un microcontrolador con un mínimo de pines.

THT (Through-Hole Technology): Tecnología de montaje de componentes electrónicos con agujero pasante.

Timeout: Periodo de tiempo predefinido durante el cual se espera que se complete una operación.

Timer: Dispositivo o función que mide y controla el tiempo en un sistema electrónico.

Top: Capa superior del PCB.

TTL (Transistor-Transistor Logic): Tecnología de lógica digital donde los niveles de voltaje específicos representan estados lógicos

UART (Universal Asynchronous Receiver/Transmitter): Componente de hardware que maneja la comunicación serial en microcontroladores.

USB (Universal Serial Bus): Estándar de conexión para periféricos de ordenadores y otros dispositivos.

uVision: Entorno de desarrollo integrado específicamente para microcontroladores ARM.

VPC (Virginal Panel Corporation): Fabricante de soluciones en conexión de alta calidad.

Watchdog: Temporizador de vigilancia para recuperar errores de software.

7. Bibliografía

[1] 6TL Engineering, empresa que desarrolla sistemas de pruebas automatizadas. Disponible en: https://6tlengineering.com (última consulta el 8 de junio de 2024).

[2] IPTE, proveedor de soluciones de automatización industrial. Disponible en: https://www.ipte.com (última consulta el 8 de junio de 2024).

[3] EIIT, soluciones de ingeniería e integración tecnológica. Disponible en: https://www.eiit.com (última consulta el 8 de junio de 2024).

[4] 6TL Engineering, plataformas automatizadas. Disponible en: https://6tlengineering.com/product-category/automated-platforms (última consulta el 8 de junio de 2024).

[5] Altium Designer, software de diseño PCB. Disponible en: https://www.altium.com/es/altium-designer (última consulta el 8 de junio de 2024).

[6] ARM, herramientas y software Keil uVision. Disponible en: https://developer.arm.com/Tools%20and%20Software/Keil%20MDK (última consulta el 8 de junio de 2024).

[7] RS Components, distribuidor de componentes electrónicos. Disponible en: https://es.rs-online.com/web (última consulta el 8 de junio de 2024).

[8] Farnell, proveedor de componentes electrónicos. Disponible en: https://es.farnell.com (última consulta el 8 de junio de 2024).

[9] Mouser Electronics, distribuidor de componentes electrónicos. Disponible en: https://www.mouser.es (última consulta el 8 de junio de 2024).

[10] STMicroelectronics, herramienta STM32CubeMX. Disponible en: https://www.st.com/content/st_com/en/stm32cubemx.html (última consulta el 8 de junio de 2024).

[11] Modbus Tools, software Modbus Poll. Disponible en: https://www.modbustools.com/modbus_poll.html (última consulta el 8 de junio de 2024).

[12] STMicroelectronics, página principal. Disponible en: https://www.st.com/content/st_com/en.html (última consulta el 8 de junio de 2024).

[13] STMicroelectronics, microcontrolador STM32G431RB. Disponible en: https://www.st.com/en/microcontrollers-microprocessors/stm32g431rb.html (última consulta el 8 de junio de 2024).

[14] TE Connectivity, productos Axicom. Disponible en: https://www.te.com/en/products/brands/axicom.html (última consulta el 8 de junio de 2024).

[15] 6TL Engineering, matriz 8x16 DPST. Disponible en: https://6tlengineering.com/product/8x16-ch-dpst-2a-250vac-matrix (última consulta el 8 de junio de 2024).

[16] National Instruments, modelo PXI-2527. Disponible en: https://www.ni.com/es-es/shop/model/pxi-2527.html (última consulta el 8 de junio de 2024).

[17] Harting, conectores PCB DIN 41612. Disponible en: https://b2b.harting.com/ebusiness/es/PCB-connectors-DIN-41612/34716 (última consulta el 8 de junio de 2024).

[18] Cliff Electronics, especificaciones del producto FCR7350. Disponible en: https://www.cliffuk.co.uk/products/testleads/sockets/fcr7350.pdf (última consulta el 8 de junio de 2024).

[19] Weidmuller, conectores para circuito impreso OMNIMATE Signal. Disponible en: https://www.weidmuller.es/es/productos/connectivity/bornes_y_conectores_para_circuito_impreso/omnimate_signal.jsp (última consulta el 8 de junio de 2024).

[20] STMicroelectronics, *datasheet* del LD1117. Disponible en: https://www.st.com/resource/en/datasheet/ld1117.pdf (última consulta el 8 de junio de 2024).

[21] Texas Instruments, especificaciones del ULN2003A. Disponible en: https://www.ti.com/product/ULN2003A/part-details/ULN2003ADR (última consulta el 8 de junio de 2024).

[22] STMicroelectronics, *datasheet* del ST3485EB. Disponible en: https://www.st.com/resource/en/datasheet/st3485eb.pdf (última consulta el 8 de junio de 2024).

[23] STMicroelectronics, manual de usuario um1817. Disponible en: https://www.st.com/resource/en/user_manual/um1819-demonstration-firmware-for-stm32091ceval-board-stmicroelectronics.pdf (última consulta el 8 de junio de 2024).

[24] Mouser Electronics, *datasheet* del producto Lite-On. Disponible en: https://www.mouser.es/datasheet/2/239/lite-on_lites10847-1-1737242.pdf (última consulta el 8 de junio de 2024).

[25] TE Connectivity, *datasheet* del producto 5-1462039-6. Disponible en: https://www.te.com/en/product-5-1462039-6.datasheet.pdf (última consulta el 8 de junio de 2024).

[26] Texas Instruments, datasheet del ULN2003A. Disponible en: https://www.ti.com/lit/ds/symlink/uln2003a.pdf (última consulta el 8 de junio de 2024).

[27] JLCPCB, página principal. Disponible en: https://jlcpcb.com/ (última consulta el 8 de junio de 2024).

[28] Modbus, protocolo de aplicación Modbus V1.1b3. Disponible en: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf (última consulta el 8 de junio de 2024).

[29] Embedded C Coding Standard, free book. Disponible en: https://barrgroup.com/sites/default/files/barr_c_coding_standard_2018.pdf (última consulta 8 de junio de 2024).