UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y desarrollo de librería de elementos gráficos para interfaz de usuario en sistemas empotrados para el control de grupos electrógenos

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Badiola Scarcella, Delfina Amaia

Tutor/a: Blanes Noguera, Juan Francisco

CURSO ACADÉMICO: 2022/2023

Dedicatoria

Quiero dedicar este trabajo de fin de carrera, que es tanto un logro como un nuevo inicio, a todas las personas que me han apoyado a lo largo de este camino.

En primer lugar, a mi familia, que ha sido un sostén constante y fundamental para alcanzar cada uno de mis objetivos. Han trabajado arduamente, sacrificándose, para ofrecerme la mejor educación posible. Mi gratitud hacia ellos es inmensa y siento que haber nacido en esta familia es el mayor obsequio. Ellos han hecho posible que sea la persona que soy hoy en día, por sus valores y su esfuerzo.

A mis amigos, que con su alegría y cariño han hecho los días largos de estudio más ligeros. Con los que he compartido experiencias únicas y siempre han estado dispuestos a ayudarme cuando más lo necesitaba. Vuestra amistad ha embellecido enormemente esta etapa de mi vida.

A mi pareja, que ha estado a mi lado en cada paso de estos últimos años, aportándome cariño y comprensión incondicionales. En especial, agradezco tu paciencia ante mis cambios de humor, mi ritmo irregular de sueño y por cuidar de mi alimentación.

Finalmente, y más significativamente, a mi hermana mayor, quien se ha convertido en mucho más que una hermana para mí. Edurne, tu ayuda, tu guía y tu incansable apoyo han sido la fuerza que me ha sostenido en los momentos de incertidumbre y cansancio. La generosidad con la que has dado todo por mí, ha dejado en mí una marca indeleble, que me recuerda que en este mundo hay amores que superan cualquier obstáculo. Este logro, aunque lleva mi nombre, es tan tuyo como mío.

Gracias a todos, desde el fondo de mi corazón. Este viaje no habría sido el mismo sin vosotros.

Agradecimientos

En esta importante etapa de mi vida académica y profesional, me gustaría expresar mi agradecimiento a todos los que han estado conmigo en este viaje.

A la empresa que me abrió sus puertas, agradezco la oportunidad que me brindaron para formar parte de su equipo. Su acogida y trato me hicieron sentir parte integrante de la familia corporativa. Su fe en mi capacidad y su apoyo constante han sido esenciales para mi crecimiento profesional y personal.

A Isabel, no tengo palabras suficientes para expresar mi gratitud. Tu ayuda, tu orientación y tu apoyo incansable han convertido estas prácticas en una experiencia única, enriquecedora e inolvidable. Gracias por alegrar los días de trabajo y por siempre dar apoyo con tu cariño. Tu influencia ha marcado mi formación profesional.

Gracias a todos desde lo más profundo de mi corazón. Este trabajo no solo representa el fin de mi carrera, sino también el amor, el esfuerzo y el apoyo de todas las personas que han creído en mí.

Resumen

El presente proyecto se enmarca dentro de la práctica realizada en la empresa Dismuntel, la cual se dedica al desarrollo de sistemas electrónicos a medida, incluyendo los encargados de controlar grupos electrógenos.

En este contexto, el enfoque del proyecto se centra en el desarrollo del HMI (Human Machine Interface), que es una interfaz de usuario o panel de control que conecta a las personas con una máquina, sistema o dispositivo. Aunque técnicamente el término puede aplicarse a cualquier sistema de E/S que permita la interacción del usuario con un computador, en este caso nos referiremos específicamente al contexto de los procesos industriales que controlan y monitorean máquinas de producción.

En particular, este Trabajo de Fin de Grado se enfocará en la interfaz gráfica, la cual desempeña un papel fundamental en cualquier sistema informático, incluidos los sistemas empotrados de control, ya que de ella depende la capacidad del usuario para operar. Además, un diseño y desarrollo adecuados de estas interfaces permiten su uso como librería en diferentes proyectos de software, lo que facilita la reutilización de componentes visuales.

Para llevar a cabo este proyecto se utilizarán los siguientes componentes de hardware: el microcontrolador RX671 de Renesas, un controlador LCD, la target board Rx671, un adaptador a LCD, y el propio LCD.

En cuanto al firmware, se utilizarán las siguientes herramientas: el entorno de desarrollo e2Studio, un entorno integrado proporcionado por Renesas basado en Eclipse; el compilador CCRx, para traducir código fuente en lenguaje C a lenguaje de máquina ejecutable por los microcontroladores de la familia Rx; y se utilizará la librería gráfica RAMTEX.

Finalmente, entre las herramientas utilizadas se encuentra ICON Edit, una herramienta de edición de iconos que se emplea para diseñar y crear iconos gráficos, utilizados en interfaces de usuario.

Palabras clave: sistemas empotrados, HMI, LCD, microcontrolador, librería gráfica, interfaz.

Abstract

This project is part of the internship at the company Dismuntel, which is dedicated to the development of customised electronic systems, including those in charge of controlling generators.

In this context, the focus of the project is on the development of the HMI (Human Machine Interface), which is a user interface or control panel that connects people with a machine, system or device. Although technically the term can be applied to any I/O system that allows user interaction with a computer, in this case we will specifically refer to the context of industrial processes that control and monitor production machines.

In particular, this Final Degree Project will focus on the graphical interface, which plays a fundamental role in any computer system, including embedded control systems, as the user's ability to operate depends on it. Furthermore, proper design and development of these interfaces allows them to be used as a library in different software projects, which facilitates the reuse of visual components.

The following hardware components will be used to carry out this project: the Renesas RX671 microcontroller, an LCD controller, the Rx671 target board, an LCD adapter and the LCD itself.

As for the firmware, the following tools will be used: the e2Studio development environment, an integrated environment provided by Renesas based on Eclipse; the CCRx compiler, to translate source code in C language to machine language executable by the microcontrollers of the Rx family; and the RAMTEX graphics library will be used.

Finally, among the tools used is ICON Edit, an icon editing tool used to design and create graphical icons used in user interfaces.

Keywords: embedded systems, HMI, LCD, microcontroller, graphics library, interface.

ÍNDICE GENERAL

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y desarrollo de librería de elementos gráficos para interfaz de usuario en sistemas empotrados para el control de grupos electrógenos

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Documento 1: Memoria

AUTOR/A: Badiola Scarcella, Delfina Amaia

Tutor/a: Blanes Noguera, Juan Francisco

CURSO ACADÉMICO: 2022-2023

Tabla de contenidos

Índice de figuras

Índice de tablas

1. Introducción

En la actualidad, los sistemas empotrados han surgido como una solución fundamental para mejorar la eficiencia y la precisión de los sistemas de control de máquinas. Estos sistemas están compuestos por una combinación de hardware y de software diseñado para ejecutar tareas de control y monitoreo complejas, garantizando el funcionamiento óptimo de los equipos y máquinas. Para la innovación y revolución del campo de la ingeniería, ha sido necesaria la integración de la tecnología de la información (todo lo relacionado con los ordenadores y el software) y la comunicación (a las formas en que las computadoras y otros dispositivos pueden comunicarse entre sí).

En la era actual, las interfaces gráficas para sistemas basados en computadoras son una parte integral del día a día de cualquier usuario de aplicaciones del siglo XXI. El entorno gráfico es usado en una multitud de aplicaciones por los usuarios, desde sistemas operativos, navegadores web, plataformas de redes sociales y aplicaciones de mensajería hasta aplicaciones móviles y cajeros automáticos, además de numerosas otras herramientas digitales. Más que nunca, estas interfaces se están convirtiendo en el medio principal de interacción entre los usuarios y la tecnología digital, haciendo que los sistemas sean más fáciles de usar y más accesibles.

En el sector industrial, las interfaces gráficas han llegado a desempeñar un papel vital a la hora de usar las máquinas. Dado que proporcionan un entorno intuitivo y accesible, se permite una visualización sencilla de los datos y facilitan el monitoreo y control en tiempo real de una variedad de sistemas. En este contexto, uno de los sectores en los que la interfaz de usuario es un requerimiento cada vez más demandado, es el de los grupos electrógenos con centrales de control basadas en sistemas empotrados. Estos sistemas de interfaz no sólo simplifican la operación de las máquinas, sino que también mejoran la eficiencia al permitir un control y monitoreo más preciso.

Los grupos electrógenos se han convertido en una solución imprescindible para asegurar el suministro de energía en situaciones críticas. Capaces de generar electricidad de manera autónoma, los grupos electrógenos proporcionan una fuente confiable de energía que es independiente de la red eléctrica convencional. Su uso se extiende a diversas industrias, lo que convierte a este proyecto en una iniciativa transversal. Esto significa que puede ser implementado en todo tipo de empresas, desde la construcción y la manufactura hasta el sector de la salud, por ejemplo, en situaciones de emergencia, donde la continuidad del suministro eléctrico es fundamental. Con el apoyo de los sistemas empotrados y las interfaces gráficas modernas, estos equipos pueden ser gestionados y controlados de manera más eficiente, optimizando así su rendimiento y confiabilidad.

1.1. Motivación y objetivos

Dismuntel es una empresa local que ofrece servicios relacionados con tecnología avanzada y cuenta con un fuerte sector de I+D, destinado a nuevos proyectos tecnológicos de ingeniería. Desde su fundación en 1997, ha logrado mejoras significativas en diversas áreas, como la gestión de energía, transporte, agricultura, fabricación de cerámica, pagos electrónicos y deportes, entre otros. A lo largo de estos años, ha participado y liderado una treintena de

proyectos de I+D financiados a nivel nacional e internacional. Actualmente cuenta con más de 70 trabajadores, disponen de una planta productiva de aproximadamente 1,900 m², está formada por una red clientelar de alrededor de 270 usuarios y está gestionando una cartera de más de 30 proyectos[1].

La empresa se especializa en llevar a cabo todos los procesos necesarios para transformar una idea en un producto tangible. Esto implica, en términos de diseño y desarrollo de montajes electrónicos, desde la concepción de la idea hasta el montaje final. Los servicios ofrecidos para lograr este objetivo incluyen: el desarrollo de la idea, la evaluación de las especificaciones, el diseño de PCB (sigla del inglés *Printed Circuit Board*, en español significa circuito impreso), la codificación, el ensamblaje e integración de componentes electrónicos, la verificación de dispositivos electrónicos, la producción y, finalmente, las pruebas y el empaque del producto final.[2]

También, se encargan de desarrollar y producir la electrónica de control de diversos dispositivos. Esto incluye el desarrollo de software, la configuración de la interfaz de usuario, la ejecución de tareas de procesamiento macro para el análisis de información, y el control de procesos para la toma de decisiones. En cuanto al hardware, diseñan, calculan y definen los componentes que conformarán el dispositivo final, de acuerdo con los requisitos de la normativa IEC.[3]

Dismuntel principalmente hace frente a sus procesos mediante pantallas que no siempre cubren las necesidades del usuario, por tener ciertas limitaciones que se mencionan a lo largo de este trabajo. Actualmente, el hardware es muy variado debido a la amplia gama de componentes y dispositivos que abarca, las diversas funcionalidades y propósitos para los que va a emplearse, la cantidad de formas y tamaños diferentes que pueden tener, la compatibilidad con otros componentes y dispositivos, y las adaptaciones para las que se va a utilizar.

Las pantallas de cristal líquido son un tipo de tecnología de visualización ampliamente utilizada en dispositivos electrónicos. Sin embargo, el control de una pantalla LCD (sigla del inglés *liquid-crystal display*, en español significa una pantalla de cristal líquido), se considera complejo debido a varios factores, como puede ser el control individual de cada píxel para conseguir una buena gestión y sincronización. También es importante controlar de manera adecuada la retroiluminación consiguiendo el equilibrio de brillo, contraste y uniformidad, pero esto también dificulta la implementación de estos dispositivos. De igual manera, es necesario diseñar de forma adecuada el circuito y escoger los controladores[1], para enviar señales adecuadas a los píxeles y gestionar la alimentación y control de la pantalla. Debido a todos estos factores, y a causa de la necesidad de cumplimentar las especificaciones y requerimientos de pantalla, la programación y configuración se compleja.

Como se ha mencionado, el hardware es muy variado, el control de una pantalla LCD es complejo y se requiere de ajustes específicos. En el mercado existen soluciones, pero no se ajustan de forma automática a todos los LCD, por lo que, en cada desarrollo de sistema empotrado hay que desarrollar una interfaz.

Atendiendo al contexto en el que se enmarca el proyecto y las razones que lo motivan, el objetivo global de este trabajo final de grado (de ahora en adelante TFG), es la creación de una capa de software de librería[2] que facilite al máximo la labor de desarrollar aplicaciones HMI (sigla

[1] Es un componente software que permite que el sistema operativo y un dispositivo se comuniquen entre sí.

[2] Es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.

del inglés *Human-Machine Interface*, en español significa Interfaz Humano-Máquina) en estos sistemas basados LCD's. Este objetivo ha sido desarrollado durante unas prácticas en la empresa de Dismuntel. Para lograrlo, se plantea una serie de subobjetivos que se articulan alrededor del objetivo principal:

- Caracterización de los parámetros de la pantalla
    - Determinación de las opciones de reloj.
    - Programación de los registros.
- Diseño de objetos gráficos
- Desarrollo de librería: funciones de uso del LCD

1.2. Conceptos previos

En esta sección se introducen algunos conceptos sobre los que se sustenta el proyecto y se describe brevemente el entorno de programación utilizado para la realización de este. En particular, se comienza introducciones una visión simplificada de la estructura del proyecto. Después, se describen los elementos que componen el hardware. Seguidamente, se define lo que es la interfaz de usuario, los tipos que hay y el tipo que se implementa en el proyecto. Finalmente, se describe brevemente el entorno de programación e2Studio utilizado para la programación y la herramienta IconEdit usada para la edición y creación de las imágenes.

1.2.1. Estructura de funcionamiento del proyecto

Previo a la explicación de los conceptos previos, se proporciona una visión simplificada de la estructura de funcionamiento del proyecto, en la figura 1. Esta estructura consiste en tres bloques principales: "aplicación", "microcontrolador" y "LCD", representados en azul oscuro. Entre los bloques de "aplicación" y "microcontrolador", se encuentra un bloque intermedio de color más claro denominado "librería", que aloja dos componentes adicionales, "funciones" y "ADDP. HW".

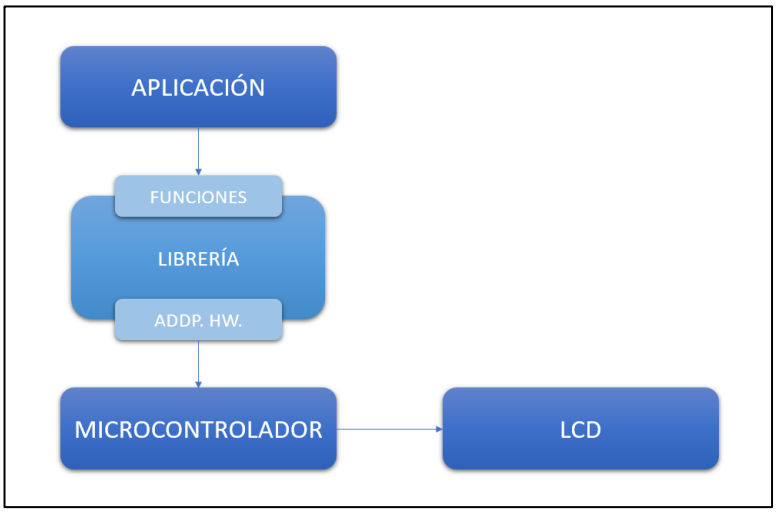


Figura 1. *Esquema del funcionamiento del proyecto*

La "aplicación" actúa como una plataforma diseñada para ejecutar operaciones específicas. La "librería", por otro lado, sirve de puente entre la "aplicación" y el "microcontrolador". Se vincula

con la "aplicación" a través de "funciones", las cuales son fragmentos de código que representan una tarea específica y facilitan la reutilización del código y la modularidad de la aplicación. Por ende, las "funciones", en este contexto, se utilizan para permitir la interacción efectiva entre la "aplicación" y la "librería". Para la conexión con el "microcontrolador", la "librería" emplea el "ADDP. HW", que esencialmente configura los registros para habilitar la funcionalidad del microcontrolador.

El microcontrolador es un dispositivo programable capaz de ejecutar las instrucciones deseadas. En este escenario, su función es transmitir la información a un LCD, que a su vez presenta la interfaz de usuario para grupos electrógenos en su pantalla.

1.2.2. Infraestructura hardware

En este apartado, se especifica cada uno de los componentes que integran el hardware del proyecto. Para brindar una perspectiva visual de cómo se ensambla el circuito en la práctica, se muestra una imagen real del circuito montado (véase figura 2).

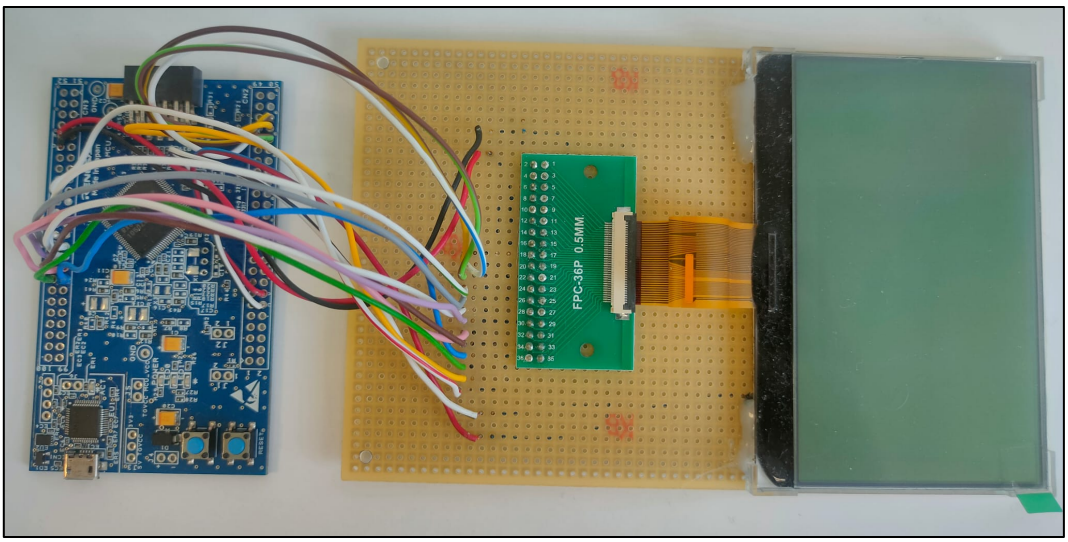


Figura 2. *Montaje del circuito*

Microcontrolador RX671*:* es un dispositivo de la familia de microcontroladores Renesas[3] RX que ofrece un alto rendimiento y una amplia gama de aplicaciones embebidas. Algunas características comunes de los microcontroladores RX incluyen una arquitectura de procesador de 32 bits, una unidad de punto flotante, una memoria flash integrada, periféricos de comunicación y E/S, y capacidades de temporización y control.

Controlador LCD*:* El ST7529 es un circuito integrado diseñado para controlar pantallas de cristal líquido. Proporciona la interfaz necesaria para controlar y gestionar el contenido que se muestra en una pantalla LCD.

Target board Rx671: La RTK5RX6710C00000BJ es una placa de desarrollo o *target board* diseñada específicamente para el microcontrolador Rx671 de Renesas. Proporciona un entorno adecuado para la programación y el desarrollo de aplicaciones basadas en ese microcontrolador.

Adaptador a LCD: es un dispositivo que se utiliza para convertir una señal de datos proveniente de una fuente, en una señal compatible con una pantalla LCD. Este adaptador

[3] Empresa especializada en microcontroladores, productos analógicos, entre otros.

específico utiliza un conector FFC (sigla del inglés *Flexible Flat Cable*, en español significa cable plano flexible) de 36 pines para la conexión.

LCD*:* El Displaytech 128240 es un tipo de pantalla LCD con una resolución de 128x240 píxeles. Se utiliza para producir imágenes y texto en dispositivos electrónicos.

1.2.3. Interfaces de usuario en sistemas empotrados

La interfaz de usuario es el medio por el cual se producen las comunicaciones de información entre seres humanos y máquinas. Permitiendo de manera sencilla y con rápida comprensión del usuario, el control del equipo electrónico. Existen varios tipos de interfaces de usuario, entre los cuales se incluyen:

Interfaz de línea de comandos (CLI): Es una de las interfaces más antiguas que se siguen usando hoy en día. Se fundamenta en la interacción con un ordenador o sistema operativo a través de comandos de textos por el usuario. En esta línea de texto se ingresan comandos para realizar acciones específicas, como ejecutar y gestionar programas o archivos.[4]

Cabe destacar que para el manejo de este tipo de interfaz es indispensable poseer un teclado. Este tipo de interfaz requiere que los operarios posean conocimientos sobre programación. Además, el control de grupos electrógenos se hace más complejo.

Interfaz gráfica de usuario (GUI): Es el tipo de interfaz más usado en la actualidad. La interfaz gráfica permite la interacción del usuario con la máquina mediante elementos visuales como objetos gráficos, iconos, ventanas y menús que representan la información.[5]

Para el manejo de la interfaz gráfica es necesario disponer, por lo menos, de un ordenador con pantalla y ratón. Para el control de grupos electrógenos puede ser requerida una placa de control que posea un LCD y una botonera. De este modo, se muestra por pantalla la información más relevante para que el operario de forma rápida y sencilla comprenda lo que está sucediendo y sea capaz de resolver el problema de una manera intuitiva con el panel de botones.

Interfaz de usuario táctil (TUI): La interfaz táctil de usuario permite la comunicación entre el usuario con un dispositivo electrónico mediante el sentido del tacto a través de una pantalla sensible. Suele ir acompañado de una interfaz gráfica de usuario que representa el panel de control que permite al humano interactuar con la máquina. [6]

En las condiciones de uso en las se debe trabajar en este proyecto, sería necesario manejar esta interfaz mediante guantes, lo que imposibilite el uso de ciertas pantallas. Además, es precioso que el operario entienda de manera clara lo que se muestra por pantalla y que sea evidente lo que debe pulsar, activar, desactivar o teclear para que todo funcione de manera adecuada. De esta manera, es posible que el hecho de utilizar este tipo de interfaz entorpezca la labor de los trabajadores más que facilitar el cometido.

Interfaz de usuario por voz (VUI) e interfaz de lenguaje natural (NLI):

La interfaz de usuario por voz permite la comunicación entre el usuario con un dispositivo electrónico a través de comandos de voz. Utilizan tecnología de reconocimiento por voz para procesar el lenguaje, transformando el habla del usuario en comandos. Los ejemplos más

conocidos son Alexa, Siri y Google Nest, son capaces de reconocer lo que dice el humano, ejecutando una acción u ofreciendo una respuesta.[7]

La interfaz de lenguaje natural permite la comunicación entre el usuario con un dispositivo electrónico mediante lenguaje natural, es decir, por medio del habla o la escritura, en lugar de contener comandos específicos o utilizar estructuras rígidas. También es capaz de identificar e interpretar acciones humanas, provenientes de movimientos y expresiones faciales. Aunque son interfaces muy veloces y sencillas de usar por el usuario, el hecho de agregar la comprensión de la máquina a este lenguaje aumenta significativamente la dificultad.[8]

El principal problema de emplear este tipo de interfaz es que son sensibles a entornos ruidosos. Los grupos electrógenos producen ruidos elevados durante su funcionamiento, generando interferencias con la interfaz de usuario por voz, lo que imposibilita el uso de esta interfaz en el proyecto.

Interfaz de realidad virtual y la interfaz de realidad aumentada:

La interfaz de realidad virtual permite al usuario sumergirse en un entorno generado por ordenador y que tenga la sensación de que es real, usando dispositivos como gafas y cascos de realidad virtual. Esta tecnología posibilita la interacción y exploración de entornos virtuales, tanto ficticios como recreaciones de la realidad.[9]

La interfaz de realidad aumentada permite la interacción entre ambientes virtuales y el mundo físico. Es decir, se superponen elementos virtuales en el mundo real, permitiendo la coexistencia e interacción con el entorno físico. Un claro ejemplo puede ser el juego Pokémon Go, en el que a través de la cámara del móvil se pueden observar criaturas virtuales que parecen estar en la realidad.[10]

El problema de estas interfaces es que necesitan espacios amplios, con buena iluminación, sin obstáculos o interferencias inalámbricas, y, si la actividad lo requiere, mandos o controladores específicos. Todo esto resulta absurdo para realizar el control y mantenimiento de grupos electrógenos, debido a que ni se cumplen las necesidades que requieren estas interfaces ni sería conveniente usarlas porque se necesita a un operario que vaya físicamente a comprobar las máquinas reales, no una simulación.

Decisión final:

Tras analizar cada uno de los tipos de interfaces de usuario y realizar una comparativa sobre el estudio de este proyecto, se concluye que la interfaz gráfica basada en tecnología LCD es la opción más acertada. La elección de la interfaz gráfica de usuario para el control de grupos electrógenos ofrece una solución que equilibra perfectamente el coste, la usabilidad y la eficiencia, por lo que se considera la opción más acertada en estas condiciones de uso. Adicionalmente el amplio rango de soluciones disponibles permite la elección de modelos adecuados para entornos agresivos como es el caso de los grupos electrógenos.

Uno de los factores determinantes de esta decisión es el aspecto económico. La implementación de una interfaz gráfica con sistemas basados en LCD en sistemas empotrados conlleva un coste, sin embargo, no es tan elevado en comparación con el de las demás opciones.

Esto permite una optimización del presupuesto sin sacrificar la calidad ni la funcionalidad del sistema.

Por otro lado, se destaca que la interfaz gráfica proporciona una visualización de la información intuitiva, clara y evidente, lo que facilita en gran medida su uso. Gracias a esta característica, es posible conocer rápidamente la situación del sistema y actuar de manera eficiente, evitando perder tiempo en descifrar interfaces complejas o en distracciones innecesarias. La facilidad de uso y lo intuitiva que es la interfaz gráfica no sólo ahorran tiempo, sino que también reducen la posibilidad de errores, lo que puede tener un impacto significativo en la seguridad y la eficiencia del sistema de control de grupos electrógenos.

1.2.4. Entorno de desarrollo y programación

El presente apartado muestra los dos entornos de desarrollo que se utilizan para la creación de la interfaz de usuario y para la programación del proyecto.

e2Studio: Es un entorno de desarrollo integrado (IDE) proporcionado por Renesas, basado en la plataforma Eclipse. Está diseñado para facilitar la programación y el desarrollo de firmware para los microcontroladores de Renesas. Al instalar esta aplicación, viene por defecto el compilador CCRx. Es un compilador propietario desarrollado por Renesas, específicamente diseñado para traducir el código fuente escrito en lenguaje de programación C a lenguaje de máquina que pueda ser ejecutado por el microcontrolador de la familia Rx.

IconEdit: Este entorno de desarrollo es una herramienta de edición de iconos que se utiliza para diseñar y crear iconos gráficos utilizados en interfaces de usuario, aplicaciones móviles y otros contextos donde se requieren representaciones visuales compactas.

2. Estudio de necesidades

Hasta la fecha actual, en la empresa Dismuntel, se utilizan pantallas de retroiluminación de 4 líneas por 20 dígitos para el control de grupos electrógenos. Estos dispositivos de visualización son ampliamente utilizados en la industria para mostrar información relevante. Cada pantalla cuenta con una matriz de 4 líneas y 20 columnas, lo que permite mostrar hasta 80 caracteres alfanuméricos, incluyendo letras, números, símbolos especiales y espacios en blanco. Además, algunas pantallas pueden mostrar gráficos simples, como líneas y formas básicas.

La forma en la que estas pantallas muestran información es mediante la activación selectiva de los segmentos individuales de cada dígito. Cada línea de la pantalla puede ser controlada de forma independiente, lo que brinda flexibilidad en la presentación de los caracteres. Sin embargo, estas pantallas tienen restricciones que perjudican a la presentación.

Dismuntel se enfrenta a diversos problemas al utilizar pantallas de retroiluminación de 4 líneas por 20 dígitos. Algunas restricciones pueden dificultar la presentación efectiva de información y limitar la capacidad de visualización de datos. Los problemas incluyen limitaciones de espacio, capacidad de visualización reducida, restricciones en la representación de gráficos y limitaciones de personalización. A continuación, se explica cada restricción de forma detallada.

1. Para empezar, estas pantallas tienen un tamaño reducido, lo que resulta en caracteres pequeños y poco legibles. La visualización de información de manera clara y precisa puede ser un desafío debido a estas limitaciones de espacio.
2. Además, poseen caracteres predefinidos en posiciones fijas, lo que limita la flexibilidad y personalización en la presentación de información. No permiten una amplia variedad de fuentes, estilos de texto o diseños de visualización, lo que puede restringir su adaptabilidad a las necesidades específicas.
3. La representación de gráficos complejos o imágenes detalladas no es adecuada en estas pantallas debido a su resolución restringida. Están diseñadas principalmente para la presentación de texto y caracteres alfanuméricos, lo que dificulta la representación precisa de gráficos más elaborados.
4. Por último, la capacidad de visualización simultánea está limitada en estas pantallas. Es necesario utilizar técnicas de desplazamiento y paginación para mostrar toda la información requerida, lo que puede afectar la eficiencia de la visualización de datos.

3. Planteamiento de la solución

La empresa Dismuntel se enfrenta a diversos problemas al utilizar pantallas de retroiluminación de 4 líneas por 20 dígitos, indicados en el apartado *2. Estudio de necesidades*, como limitaciones de espacio, capacidad de visualización restringida, limitaciones en la representación de gráficos y falta de personalización. Estos inconvenientes dificultan la presentación efectiva de información y la versatilidad de las pantallas.

En este TFG se propone una solución a estos problemas a través del uso de la pantalla Displaytech 128240. Esta pantalla ofrece mejoras significativas en términos de resolución, capacidad de visualización, representación de gráficos y flexibilidad en la presentación de información.

1. Con su alta resolución de 240x128 píxeles, la pantalla Displaytech 128240 ofrece una visualización más clara y detallada. A diferencia de las pantallas de 4x20, que disponen de caracteres más pequeños y menos resolutivos, este LCD puede exhibir una mayor cantidad de información de forma nítida y legible.
2. El Displaytech 128240 supera a las pantallas tradicionales en términos de flexibilidad. En lugar de limitarse a mostrar caracteres alfanuméricos en posiciones fijas, este LCD puede representar información pixel a pixel, lo que permite una amplia gama de fuentes, estilos de texto y diseños de visualización. Este grado de personalización hace que sea altamente adaptable a las necesidades específicas de cada aplicación.
3. Además de presentar texto, es capaz de representar gráficos complejos e imágenes detalladas gracias a su alta resolución. Esta capacidad de representar imágenes, gráficos y caracteres en cualquier área de la pantalla supera las limitaciones de las pantallas de 4x20, que están principalmente diseñadas para mostrar caracteres alfanuméricos en posiciones fijas.
4. Finalmente, debido a sus capacidades mejoradas de representación gráfica, el Displaytech 128240 puede mostrar más información simultáneamente. Al no estar restringida únicamente a caracteres alfanuméricos, puede visualizar una mayor cantidad de contenido en la misma pantalla, eliminando la necesidad de técnicas de desplazamiento y paginación.

En resumen, este TFG busca la integración del Displaytech 128240, proporcionando así una solución efectiva a los problemas que enfrenta Dismuntel con las pantallas de retroiluminación de 4 líneas por 20 dígitos.

4. Diseño y desarrollo de la solución

Para llevar a cabo este TFG, se requiere la instalación de dos entornos de desarrollo, e2Studio e IconEdit. Al descargar e2Studio, se obtiene el compilador[4] CCRx, necesario para el correcto funcionamiento del proyecto.

Asimismo, es fundamental contar con las licencias correspondientes para ambos programas. Sin ellas, en e2Studio no se dispondrá de las definiciones, funciones y configuraciones necesarias para desarrollar el código. Del mismo modo, sin la clave de licencia de IconEdit, no se podrá acceder a todas las funciones que ofrece.

Para el montaje del hardware, es indispensable contar con todos los componentes electrónicos mencionados en el apartado *1.2. Conocimientos previos*. También se requiere disponer de cables adecuados para la correcta unión de los elementos, así como de un soldador para fortalecer las conexiones y evitar posibles errores en el ensamblaje. Además, es necesario contar con un cable USB para establecer la conexión entre el dispositivo y el ordenador, y contar con un multímetro para realizar las comprobaciones y verificaciones necesarias en el proceso.

4.1. Conocimientos previos

Para poder hacer uso de los entornos de programación, las librerías y llevar a cabo el montaje de manera adecuada, es fundamental contar con conocimientos en programación y componentes electrónicos. Especialmente se requiere un dominio del lenguaje de programación C/C++, utilizado en e2Studio y una comprensión de los fundamentos electrónicos.

De este modo, es un requisito esencial contar con conocimientos en informática, para la programación, y en electrónica, para el montaje y la comprensión de los componentes utilizados. Estos conocimientos son indispensables para lograr un desarrollo y montaje exitosos, así como para abordar de manera efectiva cualquier desafío que surja durante el proceso.

4.2. Estructura del desarrollo realizado

Los desarrollos realizados en este trabajo se dividen en 2 bloques, siendo cada uno de ellos objeto de un capítulo de este TFG.

Previamente a abordar cada uno de estos 2 capítulos, se introducen otros dos capítulos en los que se especifican los fundamentos necesarios para poder llevar a cabo la propuesta y la integración del hardware. Estos dos sirven de introducción para ser capaces de entender los capítulos principales

En el capítulo de fundamentos se habla acerca del software y del hardware empleado.

[4] Es un programa que traduce código escrito en un lenguaje de programación de alto nivel en lenguaje de máquina.

- Software: Es necesario conocer los entornos de desarrollo que se utilizan para la creación de la interfaz gráfica, que permite interactuar de manera efectiva con el sistema de control. No solo se trata de manejar herramientas, sino de entender su funcionalidad y potencial, para crear una interacción efectiva con el sistema de control.

- Hardware: Es crucial conocer los componentes electrónicos involucrados y sus especificaciones. Este conocimiento, es indispensable para diseñar sistemas eficientes, duraderos y fiables, y para identificar y resolver posibles fallos. Además, proporciona las claves para seleccionar los componentes más adecuados en términos de rendimiento, coste y eficiencia energética, asegurando así la viabilidad y el éxito del proyecto.

Por otro lado, en el segundo capítulo de integración del hardware, se habla acerca de las configuraciones de la placa y del controlador:

- Configuración de la Target board para Rx671 en e2Studio: Esta placa de desarrollo es uno de los componentes hardware que requiere configurarse dentro del entorno de desarrollo de e2Studio. Para ello, es necesario operar en el modo de trabajo denominado "Smart Configurator", ya que facilita unas herramientas y opciones que permiten un ajuste preciso del sistema de hardware. Además, permite crear código automáticamente incorporando los cambios de las opciones de configuración seleccionadas.

- Configuración del controlador st7529 en e2Studio: Gracias a este controlador, se permite la configuración de los parámetros iniciales que serán los encargados de mejorar la visualización de la pantalla. Para ello, es indispensable contar con la librería adecuada para dicho controlador.

Los capítulos principales, que contienen el desarrollo del proyecto, son:

- Desarrollo gráfico de widget con IconEdit

A partir de los fundamentos descritos en los apartados anteriores, se explica cómo se crean y diseñan iconos gráficos que constituyen la interfaz del usuario. Se detalla cada uno de los gráficos utilizados, así como la razón por la que se utiliza el tipo de gráfico para cada tipo de valor que se desea mostrar por pantalla.

- Programación e integración

En este capítulo se habla acerca de los archivos de las librerías que son configurados durante el desarrollo del proyecto y se comenta cuáles han sido las modificaciones realizadas en cada uno. Además, también se explica el código del proyecto principal, para que sea sencillo de comprender el funcionamiento de la interfaz gráfica de usuario.

5. Fundamentos necesarios

En este apartado se explican los fundamentos en IconEdit y e2Studio necesarios para crear la interfaz de usuario. También es indispensable mostrar los componentes electrónicos con los que se trabaja y sus especificaciones.

Cabe mencionar que, para la realización de este TFG, la alumna ha invertido tiempo en aprender el uso de las herramientas de desarrollo modernas para sistemas empotrados. Esto es debido a que, con los conocimientos aprendidos tras finalizar la carrera, no fueron suficientes para desarrollar este proyecto. Por ello, en este punto, se proporciona gran parte del conocimiento adquirido para la comprensión del proyecto.

5.1. Software

Se comparte a continuación información sobre los entornos de desarrollo empleados.

5.1.1. e2Studio

En la presente sección se proporcionan dos guías rápidas: para la creación de un nuevo proyecto y para conocer el entorno de desarrollo.

5.1.1.1. Guía para la creación de un nuevo proyecto.

El primer paso para la creación de un nuevo proyecto es pulsar en el menú principal la opción “File”, escoger la opción “New” y por último hacer clic sobre “C/C++ Project” (véase figura 3).

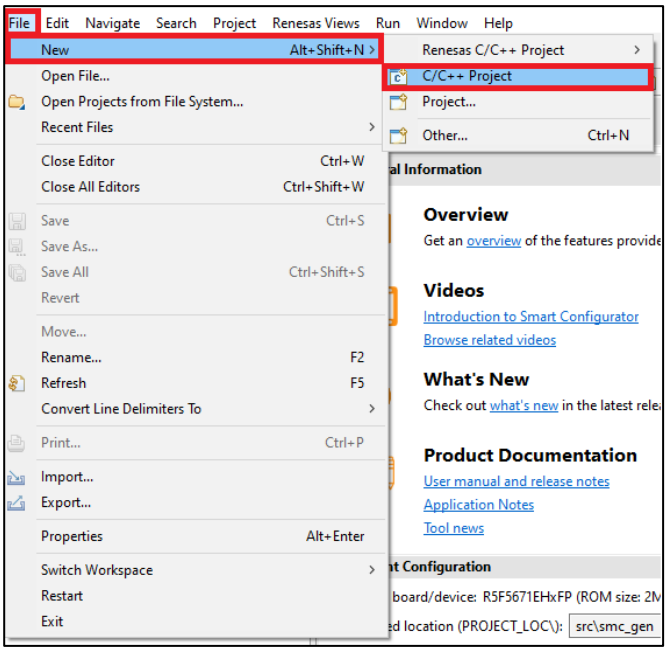


Figura 3. *Creación de un nuevo proyecto*

A continuación, se abre una ventana emergente en la que se debe seleccionar, en la parte izquierda, la opción “Renesas RX”. Después, en la parte derecha aparecerán una serie de

opciones, por lo que, se deberá presionar la opción “Renesas CC-RX C/C++ Executable Project”. Por último, se debe pulsar el botón inferior “Next (véase figura 4).

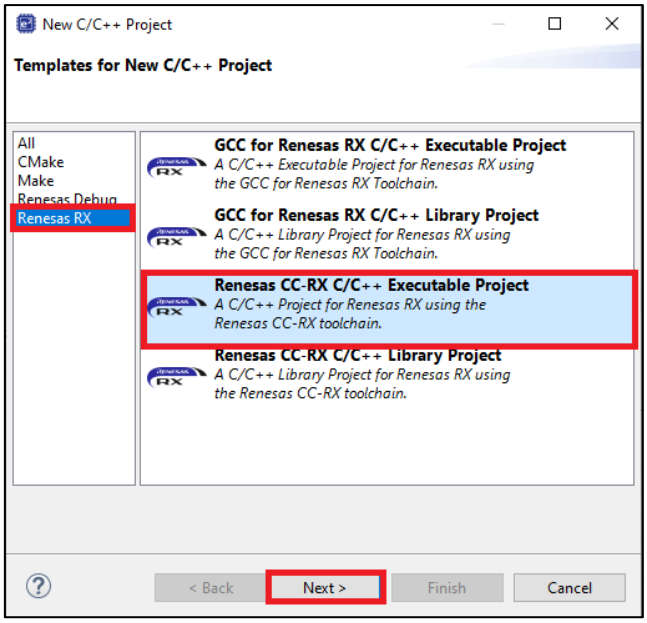


Figura 4. *Selección del tipo de proyecto de la familia RX*

Seguidamente, se debe introducir el nombre del proyecto y la ruta del proyecto (siempre y cuando se desee cambiar). Para hacer esto último, se debe quitar la opción de “User default location” y seleccionar la nueva ubicación en la que guardar el fichero. Por último, se pulsa el botón “Next” para continuar con las configuraciones (véase figura 5).

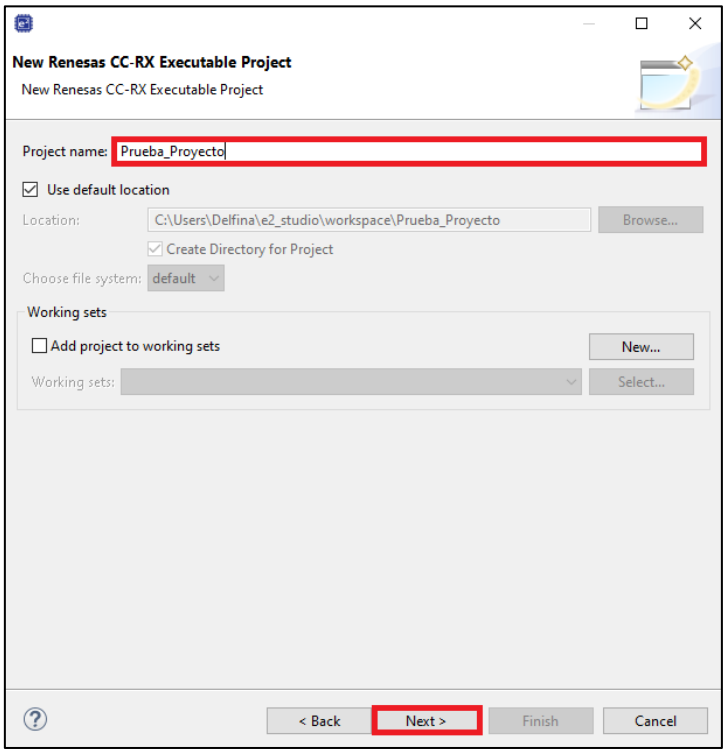


Figura 5. *Ubicación y nombre del proyecto*

Acto seguido, se debe escoger el compilador CC-RX. Para ello, en el campo “Toolchain” seleccionaremos la opción “Renesas CC-RX” y en el campo “Toolchain Version” la opción que muestre la versión más reciente de este compilador, en este caso “v3.05.00”. Para seleccionar el dispositivo “R5F5671EHxFP”, empleado en este TFG, se debe hace clic sobre los tres puntos situados a la derecha del campo “Target Devices”. Tras pulsar se abre una nueva ventana emergente en la que se harán las siguientes elecciones: “RX600” > “RX671” > “RX671 - 100pin” >“R5F5671EHxFP”. Después se presiona “OK” para guardar. Para finalizar, se debe seleccionar “E2 Lite (RX)” en el apartado de “Configurations” y hacer clic sobre el botón “Next” (véase figura 6).

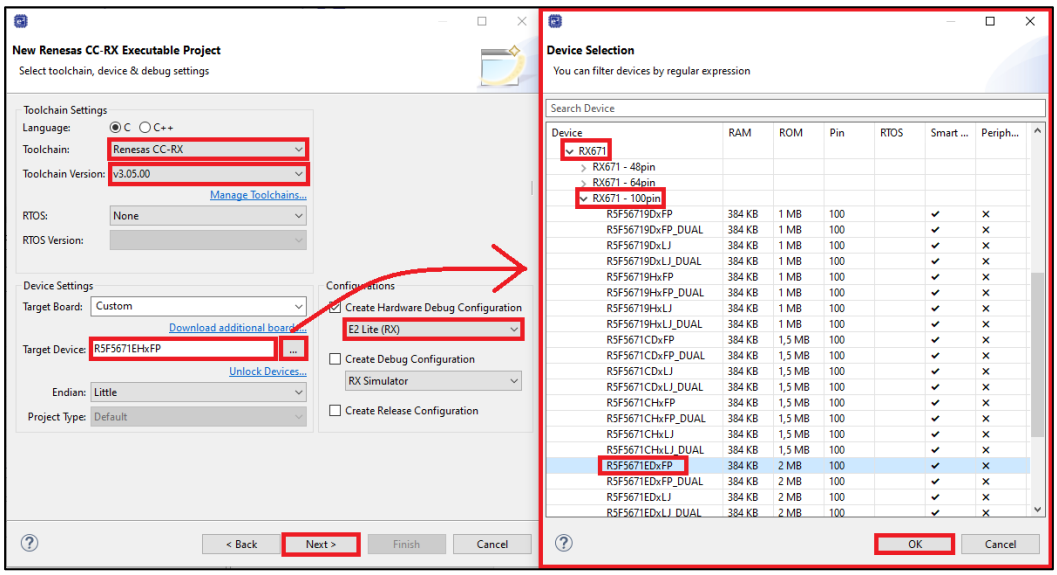


Figura 6. *Selección de compilador, versión y dispositivo*

El último paso en esta ventana emergente nos da la opción de habilitar “Use Smart Configurator”. Si se quiere activar, se debe marcar y pulsar “Finish” (véase figura 7).

Esta opción sirve para realizan las configuraciones relacionadas con el microcontrolador. Es uno de los modos de los que dispone el programa e2Studio.

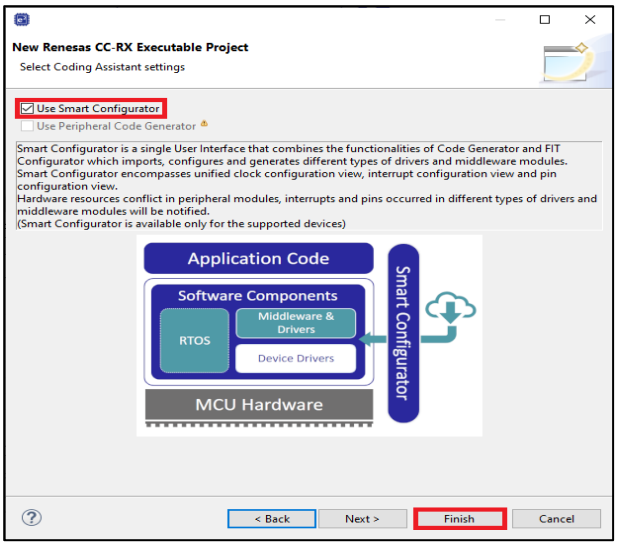


Figura 7. *Permitir uso del Smart Configurator*

Siguiendo estos pasos, se consigue la creación del proyecto en e2Studio, con la placa correspondiente para este TFG.

5.1.1.2. Guía para conocer el entorno de desarrollo

Tras la creación del nuevo proyecto, el entorno de desarrollo inicialmente se organiza en cinco secciones. Estas comprenden diversas ventanas que tienen la capacidad de mostrarse u ocultarse, permitiendo en el mismo espacio diferentes acciones. La apariencia inicial de la aplicación al comenzar una sesión de trabajo se ilustra en la figura 8.

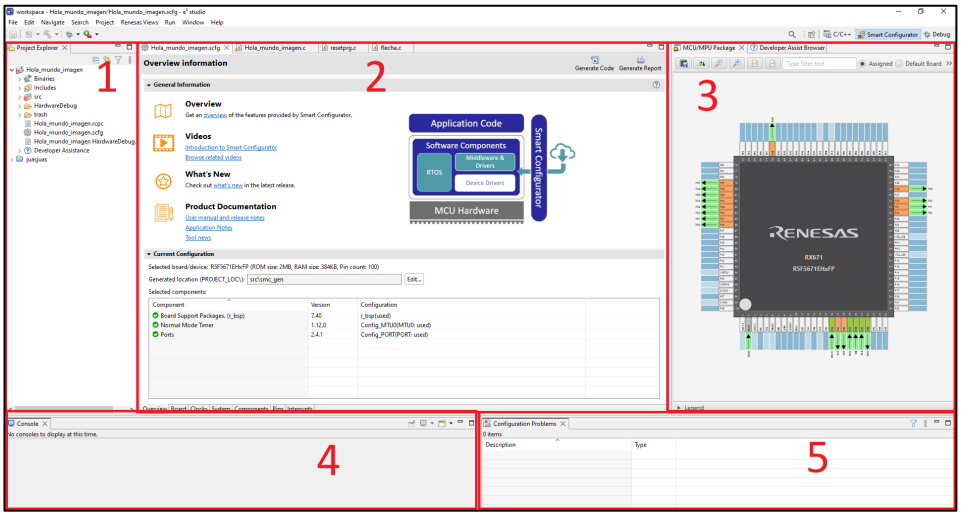


Figura 8. *Visita general de e2Studio*

En la parte superior se encuentra la ventana principal, que contiene el menú principal, la barra de herramientas en la parte izquierda y el modo de trabajo en la parte derecha.

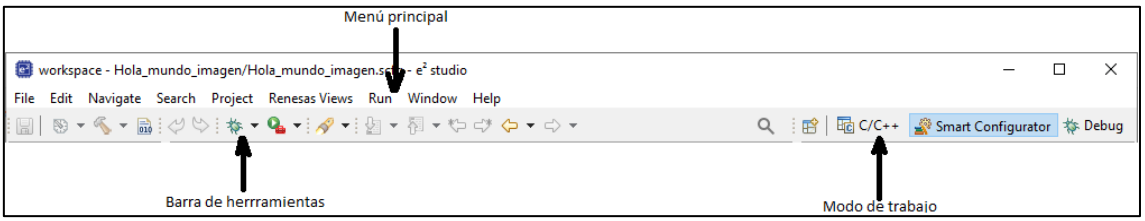


Figura 9. *Componentes de la ventana principal*

El menú principal brinda acceso a todas las funciones y facilita la personalización del programa. Cuenta con los menús estándar presentes en cualquier aplicación de Windows, como “File”, “Edit”, “Search”, “Help, entre otros, además de otros específicos de este software, como puede ser la opción “Renesas Views”. La barra de herramientas está diseñada para acelerar la ejecución de las tareas más habituales, accesibles desde el menú principal, mejorando así la eficiencia de uso del entorno. Al situar el ratón sobre los íconos, se despliegan cuadros de ayuda que proporcionan información sobre la funcionalidad de cada uno. Finalmente, el modo de trabajo, tal como sugiere su denominación, permite cambiar la forma en la que se está trabajando, presentando la información más relevante de acuerdo a la opción seleccionada en las ventanas 2 y 3 de la figura 8.

En la parte inferior izquierda de la ventana principal, se ubica la ventana de navegación de proyectos. En el ejemplo mostrado en la figura 10, se pueden observar dos proyectos:

“Hola_mundo_imagen”, que representa el proyecto completo, y “Proyecto_nuevo”, que es un proyecto recién creado.

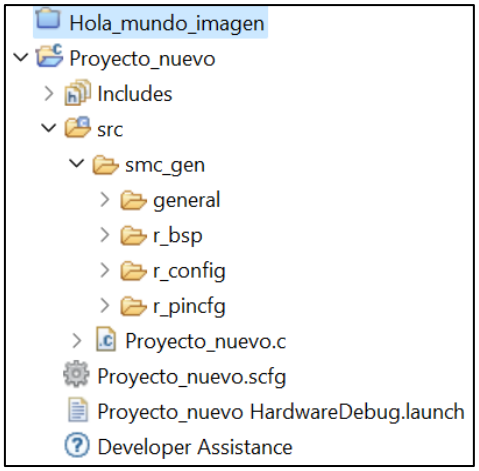


Figura 10. *Carpetas que componen un proyecto nuevo*

Al desplegar un proyecto recién creado (en este caso, “Proyecto_nuevo”), se aprecian los archivos que poseen la información básica. Un archivo clave es aquel que lleva el nombre del proyecto con la extensión “.scfg” (como en este caso “Proyecto_nuevo.scfg”), contendiendo la configuración para el microcontrolador (véase figura 10).

En la carpeta “src” existe la subcarpeta “smc_gen”, donde se localizan las carpetas predeterminadas que albergan los componentes y configuraciones iniciales. De igual forma, se ubica el segundo archivo a resaltar, el cual lleva el nombre del proyecto con la extensión “.c”. Este representa el archivo fundamental del proyecto, siendo el que se utiliza y modifica con mayor frecuencia (véase figura 10).

Cuando se pone en marcha el proyecto se genera la carpeta “HardwareDebug” y la carpeta “trash”, la cual almacena las últimas modificaciones registradas del proyecto. En el interior de la carpeta “src”, se incorpora la subcarpeta “gclcd”, que contiene librerías, y la carpeta “imagen”, que aloja los símbolos e imágenes que se emplearán para el desarrollo de la interfaz de usuario (véase figura 11).

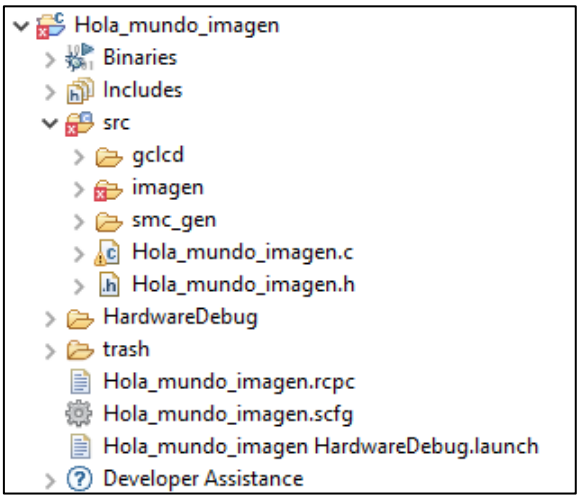


Figura 11. *Carpetas principales del proyecto completo*

A continuación, se mencionan las carpetas y los archivos de mayor relevancia que contiene la carpeta “src”. En la carpeta “inc”, destacan los archivos “gdisp.h”, “gdisphw.h” y “st7529.h”, que contienen definiciones cruciales. Dentro de la carpeta “st7529”, se hallan diversos archivos y tres carpetas adicionales, destacando los archivos “gdispcfg.h”, “ghwioinit.c”, “bussim.c”, “sgtypes.h” y “ghwinitctrl.c”. Cada uno de ellos alberga definiciones, funciones o datos que se emplean en el proyecto, por lo que resulta esencial mantener un seguimiento de estos archivos (véase figura 12).

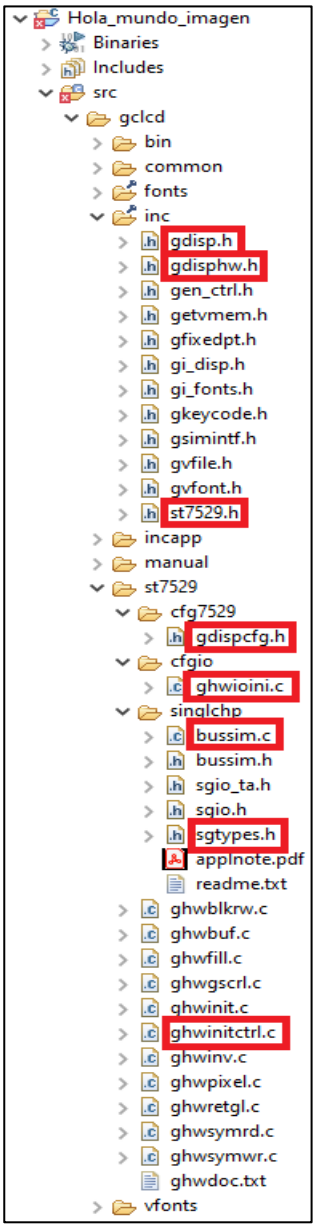


Figura 12. *Archivos más relevantes*

En la ventana 2 mostrada en la figura 8, se encuentra el área de trabajo de e2Studio, que inicialmente muestra el archivo con terminación “scfg”, en el que se pueden modificar diversas configuraciones.

En la ventana 3 indicada en la figura 8, se localiza la ventana que inicialmente muestra la visualización del microcontrolador con la configuración de sus pines.

Se hablará de ambas ventanas simultáneamente, ya que el contenido de ellas depende, entre otras cosas, del modo de trabajo que esté seleccionado.

En este entorno de desarrollo existen tres modos de trabajo: “C/C++”, es el modo en el que se modifica el código; “Smart Configurator” (opción utilizada en la creación del proyecto), en el que se realizan las configuraciones relacionadas con el microcontrolador; y, por último, “Debug”, es el estado en el que se encuentra el programa cuando se está ejecutando el proyecto.

Inicialmente el proyecto se encuentra en modo “Smart Configurator”. En el área de trabajo, en la parte inferior izquierda, se muestran una serie de opciones: “Overview”, “Board”, “Clocks”, “System”, “Components”, “Pins” e “Interrupts” (véase figura 13).

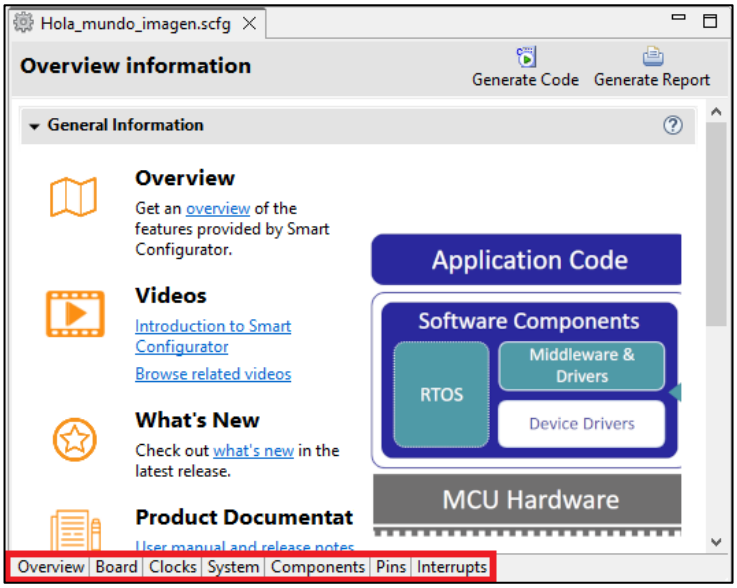


Figura 13. Opciones de configuración del Smart Configurator

Al inicio, el proyecto se encuentra en la opción “Overview”, que muestra información al usuario principiante, como enlaces a vídeos y a documentos para saber cómo funciona el programa.

La opción denominada "Board" indica el tipo de dispositivo con el que se trabaja. Si durante la fase de creación del proyecto se eligió incorrectamente el dispositivo, puede rectificarse a través de esta función.

Por otro lado, la opción "Clocks" permite ajustar los relojes del sistema para adaptarse a las necesidades del proyecto (véase figura 14). Esto incluye la frecuencia del reloj principal, la selección de las fuentes de reloj para diferentes periféricos y la configuración de los divisores de reloj para obtener las frecuencias de reloj deseadas. También permite habilitar o deshabilitar relojes para conservar energía. Adicionalmente, al utilizar “Smart Configurator” para configurar los relojes, se genera automáticamente el código necesario para implementar estas configuraciones de reloj en su programa, ahorrando tiempo y reduciendo la posibilidad de errores.

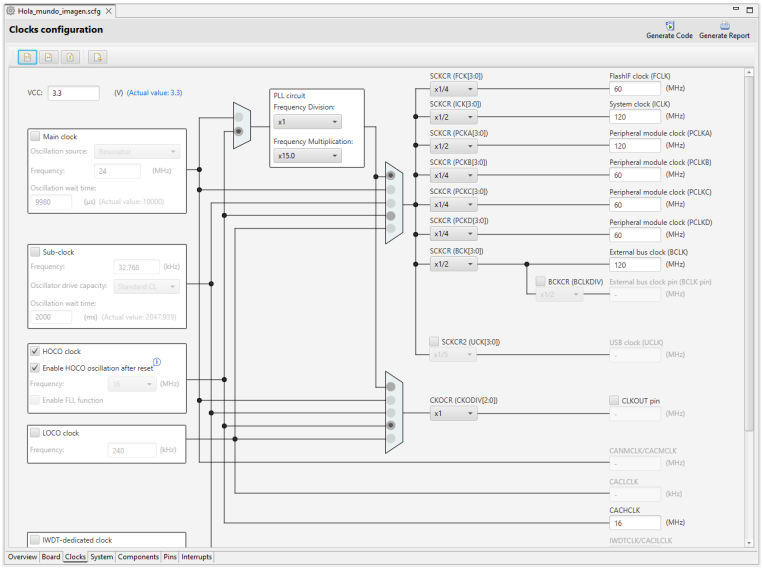


Figura 14. *Smart Configurator, configuración de relojes*

En la cuarta opción, “System”, se encuentran las tres interfaces de depuración que se pueden seleccionar (véase figura 15).

- “Unused”: empleada cuando no se planea utilizar ninguna interfaz de depuración.
- “FINE” (*Emulator debug function*): seleccionada si se planea utilizar la interfaz de depuración del emulador “FINE”. “FINE” es una interfaz propietaria de Renesas que permite una gran cantidad de funcionalidades de depuración, incluyendo la capacidad de detenerse en puntos de interrupción, visualizar y modificar el contenido de la memoria y los registros, y trazar la ejecución del programa.
- “JTAG” (sigla del inglés *Joint Test Action Group*): escogida si se planea utilizar la interfaz de depuración “JTAG”. “JTAG” es una norma común para la interfaz de prueba y depuración que se utiliza en muchos microcontroladores y otros chips. “JTAG” permite la depuración en circuito, lo que significa que puede depurar el microcontrolador mientras está en el circuito y funcionando.

Para el proyecto que se desarrolla durante este TFG, se selecciona la opción de “JTAG”.

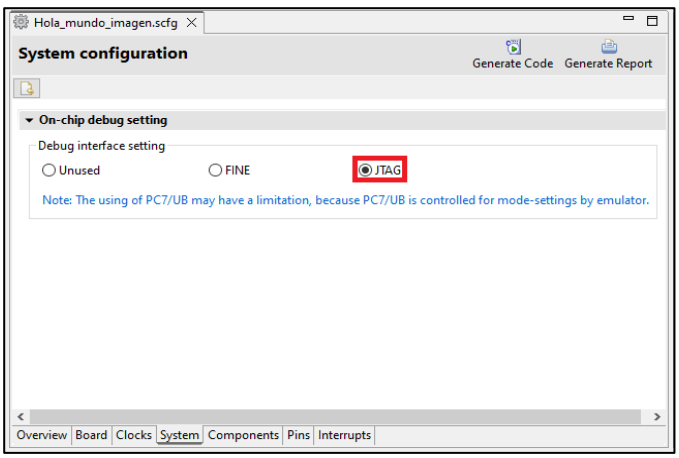


Figura 15. *Smart configurator, interfaces de depuración*

La siguiente opción, “Components”, es extremadamente útil para gestionar y configurar los componentes de software. Estos componentes pueden ser controladores de dispositivos, middleware, bibliotecas y otros paquetes de software. Además, una vez se termina la selección configuración, se genera automáticamente el código necesario para la inicialización y uso de dichos elementos.

Al hacer clic sobre cualquier componente, aparecen los parámetros de este elemento, que pueden ser modificados.

Para incorporar componentes adicionales al proyecto, se debe hacer clic en el ícono representado por un signo más (+) de color verde ubicado en la sección "Components" (este ícono está marcado en rojo en la figura 16, en la esquina superior izquierda). Al seleccionarlo se abrirá una ventana en el lado derecho de la pantalla, en la que se pueden filtrar los tipos de componentes según lo ingresado por el usuario. Luego, se debe escoger el componente que mejor se ajusta a las necesidades del proyecto y se presiona el botón "Finish". Así, el nuevo componente aparecerá en la sección izquierda de la pantalla.

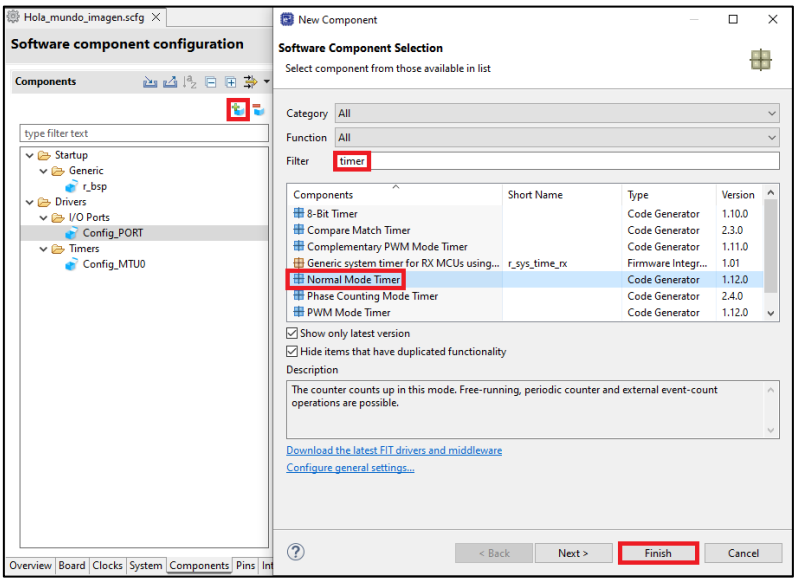


Figura 16. *Selección de nuevos componentes*

La opción "Pins" se utiliza para la configuración de los pines del microcontrolador. Este configurador de pines permite asignar funciones a los pines del chip, definiendo su modo de funcionamiento (como entradas o salidas, por ejemplo) y otras características, como resistencias pull-up/pull-down[5], modo de manejo de la interrupción…

Por otro lado, la opción "Interrupts" se utiliza para configurar las interrupciones del microcontrolador. Las interrupciones son señales que detienen temporalmente el flujo normal de la ejecución del programa para realizar alguna tarea específica. A través de esta opción, se pueden configurar las prioridades de las interrupciones, habilitar o deshabilitar ciertas interrupciones, y vincularlas a funciones específicas que se deben ejecutar cuando se produce la interrupción.

[5] Estados que disponen las resistencias cuando el pin se encuentra en resposo. Pull-up: establecen un estado HIGH. Pull-down: establecen un estado LOW.

Prosiguiendo con las ventanas que componen la vista general de e2Studio, la tercera ventana mostrada en la figura 8, es inicialmente la visualización del microcontrolador con la configuración de sus pines. Como se aprecia en la figura 17, el microcontrolador consta de 100 pines, cada uno con su respectiva conexión y dirección (entrada o salida) claramente identificadas. Esta visualización se vuelve especialmente beneficiosa ya que proporciona una perspectiva rápida y detallada de la configuración del microcontrolador, lo que facilita la verificación de las conexiones y contribuye a una más efectiva detección de posibles errores o inconsistencias en la configuración.

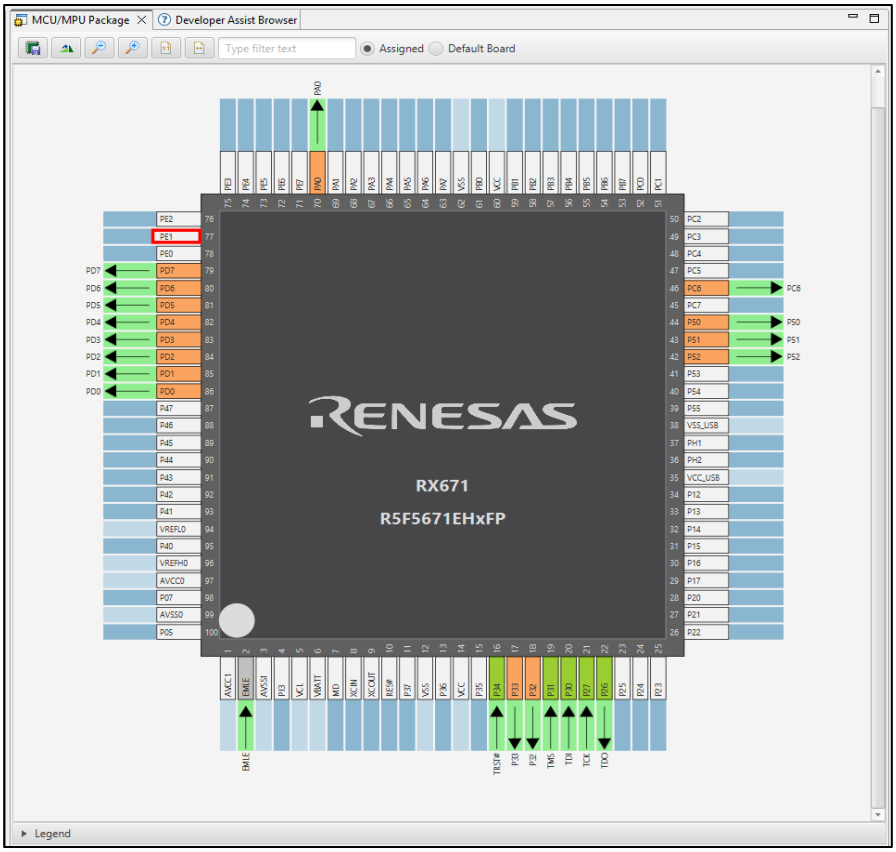


Figura 17. *Smart Configurator, visualización del microcontrolador*

El entorno de desarrollo e2Studio permite cambiar el modo de trabajo a "C/C++", que es el espacio en el que se lleva a cabo la edición y modificación del código. Este cambio de modo puede realizarse seleccionando la opción "C/C++" en la barra de modos de trabajo, o simplemente haciendo clic en cualquier archivo que no sea el archivo con el nombre del proyecto y la extensión "scfg". Al hacerlo, el área de trabajo mostrará el código del archivo seleccionado con sintaxis coloreada (se pintan las palabras con diferentes colores para distinguir entre los diversos elementos del lenguaje de programación).

Es importante destacar que, al pasar a este modo, ya no se visualiza el microcontrolador en la parte derecha de la pantalla como ocurre en el modo "Smart Configurator". En su lugar, se presenta una ventana denominada "Outline". Esta ventana ofrece una representación simplificada del código fuente del archivo abierto en ese momento. De forma concreta, presenta una organización jerárquica de todas las funciones, variables y estructuras que se definen en el archivo, facilitando así la comprensión y navegación a través del código (véase figura 18).

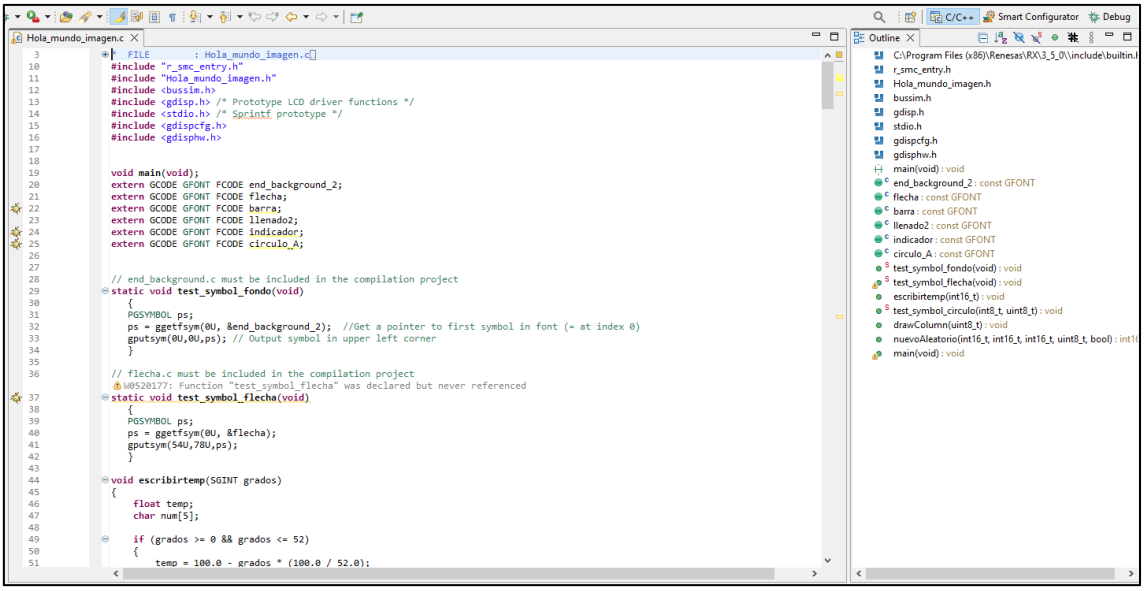


Figura 18. *Modo de trabajo C/C++, edición de textos*

Una vez completado el proceso de escritura del programa, el siguiente paso es la compilación, un paso crucial para identificar y corregir posibles errores en el código. Esta tarea se realiza a través del IDE, que ejecuta el compilador y emite una serie de mensajes (errores, advertencias e información adicional) que se presentan en la ventana de consola, identificada como la ventana número 4 en la figura 8. Este mecanismo permite al usuario no sólo entender qué ocurre durante la compilación o ejecución del programa, sino también facilita la localización y corrección de los errores que pudieran haber surgido en el código.

Además de la ventana de consola, el IDE proporciona una herramienta adicional conocida como Problemas de Configuración, que se identifica como la ventana número 5 en la figura 8. Este recurso es invaluable cuando se trata de rastrear problemas específicos relacionados con la configuración del proyecto. Como rutas incorrectas, versiones de bibliotecas incompatibles o cualquier otro problema de configuración, estos se mostrarán en la lista de Problemas de Configuración. Esto puede ser extremadamente útil para detectar y solucionar problemas que pueden impedir que el proyecto se compile o se ejecute correctamente.

Una vez resueltos todos los errores, llega el momento de pasar a ejecutar el programa. En la barra de herramientas, se encuentra disponible un ícono representado por un escarabajo verde, que ofrece la opción de "Debug". Al seleccionar este ícono, se despliegan tres opciones, siendo necesario elegir "Debug Configurations" (véase figura 19).

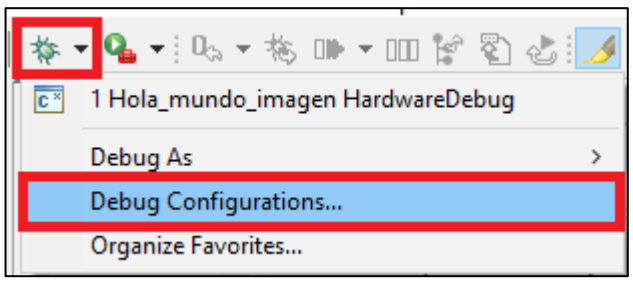


Figura 19. *Primer paso para poder entrar en modo Debug*

Al seleccionar esta opción, se abre una ventana en la cual, en la parte izquierda, se encuentra la sección "Renesas GDB Hardware Debugging". En dicha sección, es importante seleccionar el proyecto que se desea ejecutar, el cual se mostrará con el nombre del proyecto seguido de un espacio y la terminación "HardwareDebug". Posteriormente, en la parte superior derecha, se debe hacer clic en "Debugger", representado por el mencionado ícono del escarabajo verde. A continuación, se procede a la sección de "Connection Settings" y, dentro de "Connection Type", se elige la opción "Fine" del menú desplegable. Una vez realizados estos pasos, se selecciona el botón "Apply" después, se presiona el botón "Debug" para iniciar la ejecución del programa (véase figura 20).

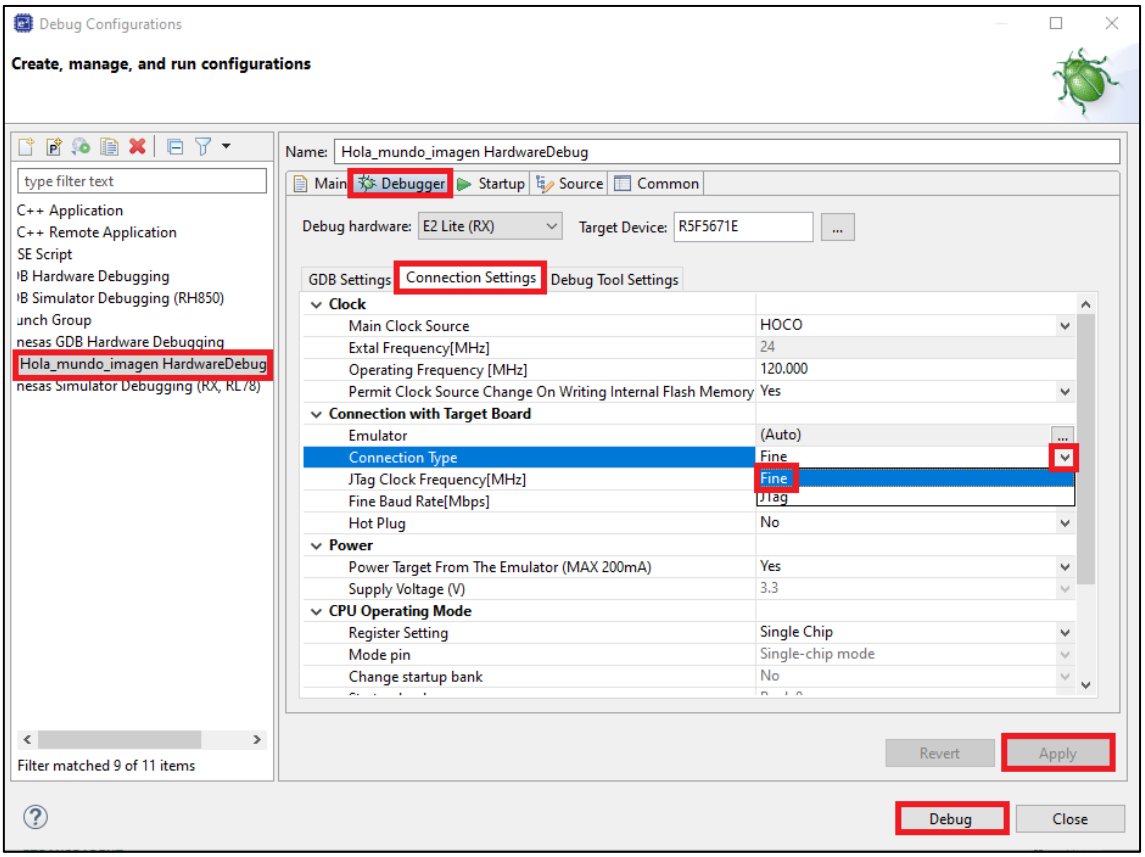


Figura 20. *Configuración del modo Debug*

Al ejecutar el proyecto, el programa entra en el modo "Debug". En este modo, al igual que en el modo de trabajo "C/C++", el área de trabajo muestra el código del archivo seleccionado. Sin embargo, en la ventana derecha ya no se visualiza el microcontrolador, en su lugar aparece una nueva ventana que ofrece múltiples acciones. Estas acciones son: "Variables", "Breakpoints", "Project Explorer", "Expressions" y "I/O Registers". Cada una de estas acciones proporciona funcionalidades específicas para el proceso de depuración y análisis del programa en ejecución (véase figura 21).

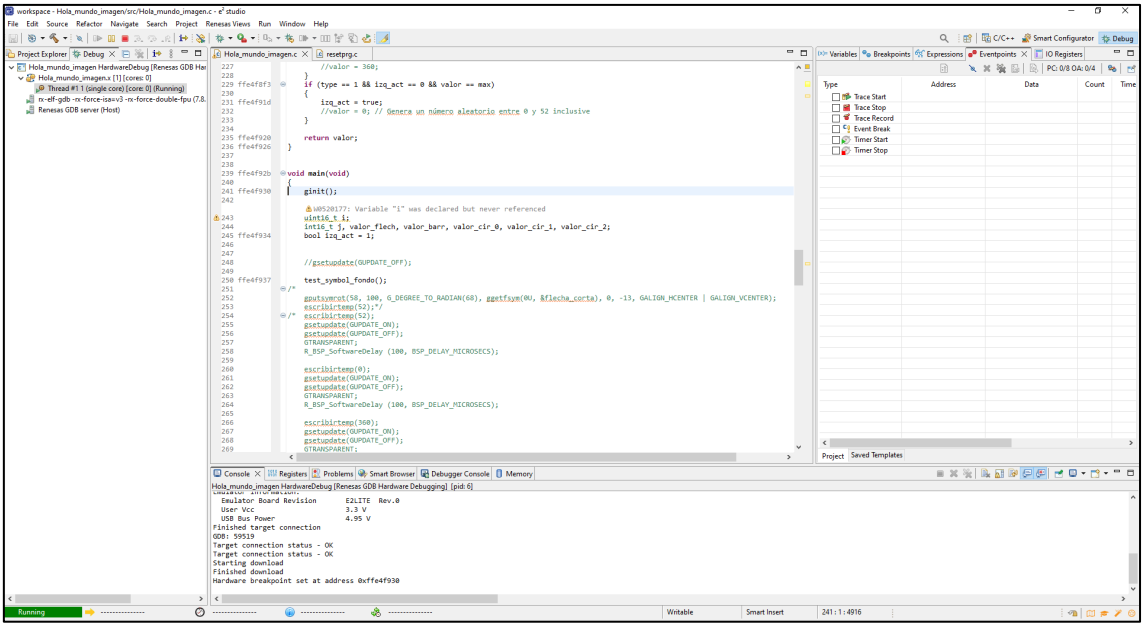


Figura 21. *Modo de trabajo Debug, ejecución del proyecto*

Siguiendo estos pasos, se consigue la comprensión de cómo trabajar en el entorno de trabajo.

5.1.2. IconEdit

Al igual que el apartado anterior, se va a hacer una guía acerca del entorno de desarrollo, para saber lo esencial para crear una imagen.[11] [12]

5.1.2.1. Guía de las acciones más relevantes

Al ingresar al programa IconEdit, el entorno de desarrollo consta de una ventana principal en la parte superior que incluye el menú principal, la barra de herramientas principal y las fichas de selección de proyectos. A continuación, se encuentran las pestañas de la barra de selección de modo de edición, seguidas de la barra de herramientas de edición y la paleta de colores (véase figura 22). El área restante de la pantalla es el espacio de trabajo de ICON Edit, donde se visualiza y se puede editar el símbolo en proceso (véase figura 23).

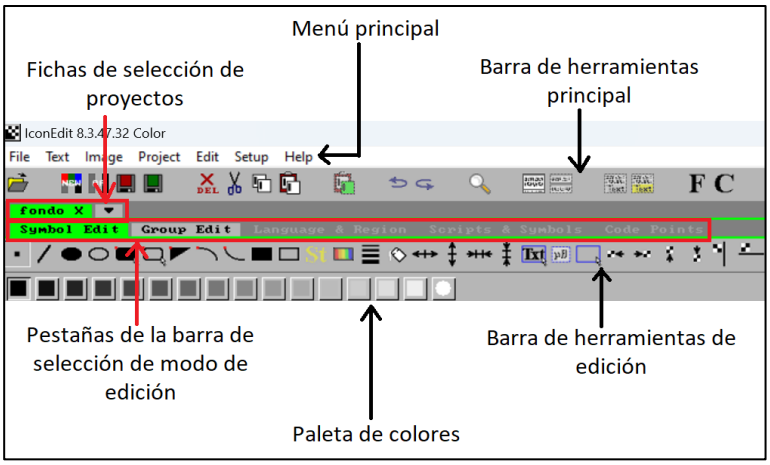


Figura 22. *Componentes de la ventana principal de IconEdit*

La figura 23 muestra la visión general del entorno de desarrollo al seleccionar las dimensiones de 240 de ancho por 128 de alto, y al haber seleccionado la opción de crear un símbolo.

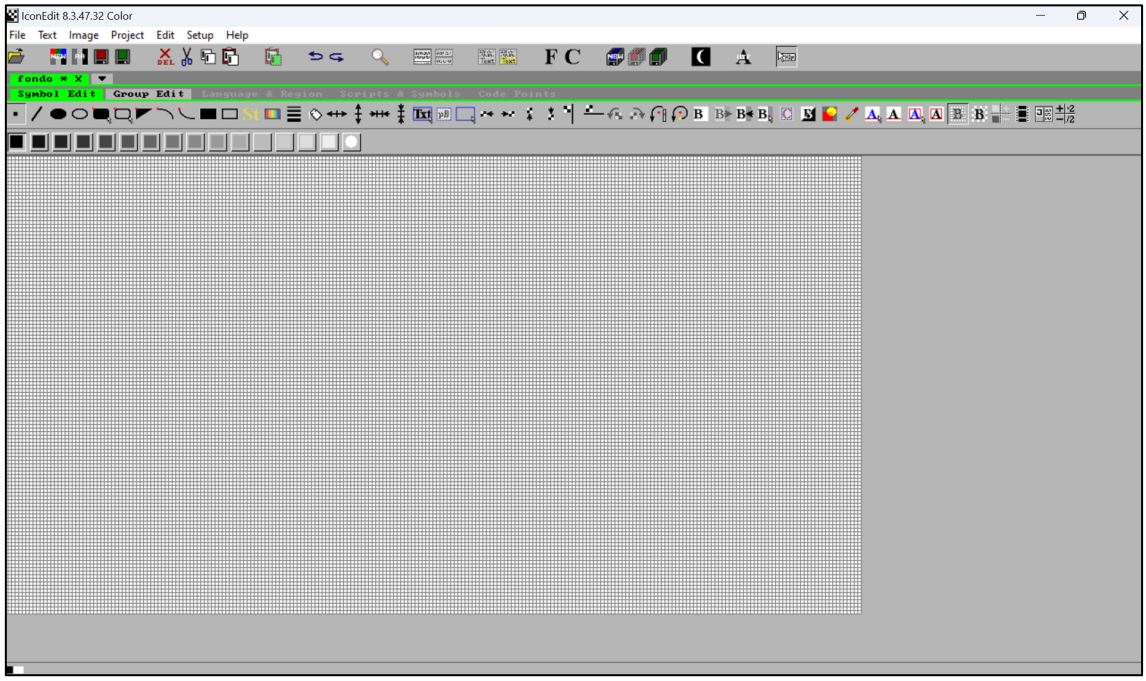


Figura 23. *Visita general de IconEdit*

Al seleccionar la segunda opción “Group Edit” en las pestañas de la barra de selección de modo de edición, se puede ver, que permite crear más de un símbolo en el mismo proyecto (véase figura 24). Esto permite a la hora de programar, al introducir un único archivo .c y .sym dentro del proyecto de e2Studio, se puede elegir cualquiera de los símbolos creados. Hay que tener en cuenta, que todos tendrán la misma dimensión.

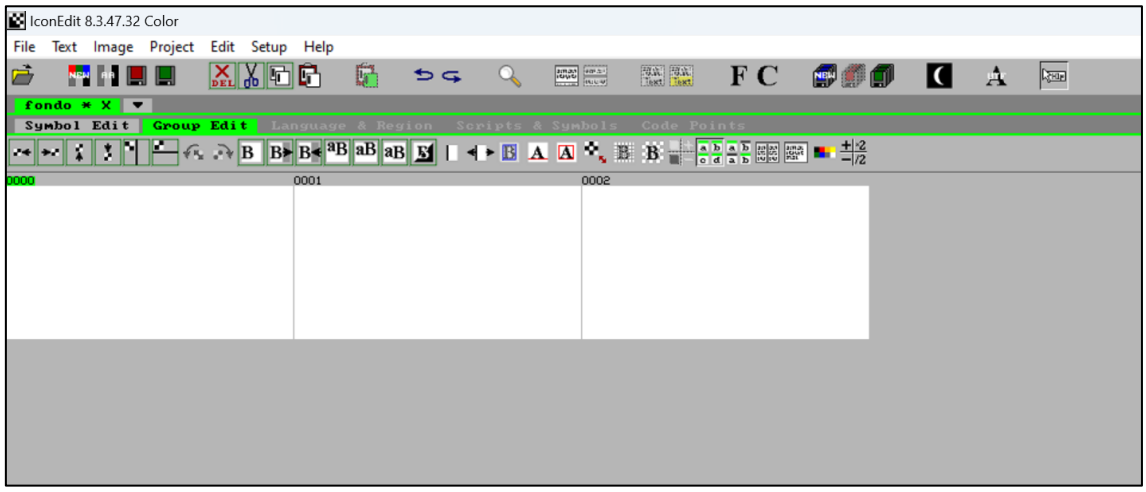


Figura 24. *Grupo de símbolos*

A continuación, se muestran en la tabla 1 las herramientas más usadas en dicho programa.

	Eliminar
	Copiar
	Pegar
	Deshacer
	Rehacer
	Insertar varios símbolos en el signo de intercalación
	Dibujar píxel a píxel
	Dibujar línea
	Elipse hueca
	Dibujar figura rectangular
	Rellenar
	Escribir
	Una vez pulsado el botón de “Escribir”, en la barra de herramientas principal aparece esta opción. Al hacer clic sobre ese icono, se puede modificar elementos relacionadas con el texto (véase figura 25). *
	Editar en fotograma o mover fotogram
	Espejo
	Rotar alrededor de un punto
	Mostrar cuadrícula de píxeles
	Cambiar el tamaño de píxel mostrado

Tabla 1. *Herramientas IconEdit*

* Al pulsar el icono decimotercero, se accede a una ventana dedicada a la edición del texto (véase figura 25). En la parte superior derecha de la ventana, se encuentra la opción para seleccionar el tipo de letra deseado. En la parte inferior izquierda, se proporciona la posibilidad de elegir la posición inicial de la primera letra y ajustar el tamaño de la fuente. En el panel derecho, se ofrecen opciones adicionales para aplicar semi negrita, negrita o cursiva. Una vez que se han realizado todas las selecciones, se confirman los ajustes haciendo clic en el botón "Ok".

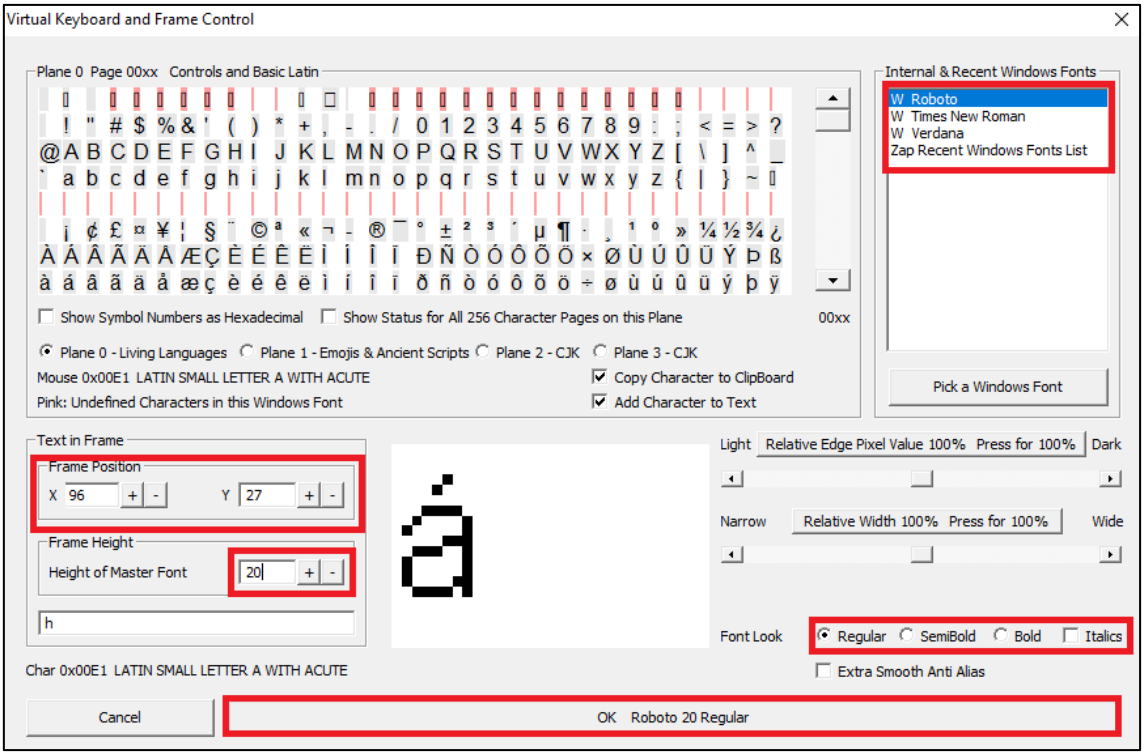


Figura 25. *Configuración de texto, IconEdit*

Una vez, se posee toda esta información, lo más recomendable es abrir el documento "IconEditQuickGuide.pdf" que aparece en la carpeta de "iconedit", y realizar algún ejemplo de los que aparece, para entrar en contacto con el programa.

5.2. Hardware

En los siguientes apartados, se desglosan y analizan detalladamente cada uno de los componentes de hardware que conforman el proyecto.

5.2.1. Microcontrolador RX671

El microcontrolador RX671 es un microcontrolador (MCU) de 32 bits basado en el núcleo de CPU RXv3[6] (véase figura 26). Algunas de sus características más importantes son:

- Tiene una frecuencia de operación máxima de 120 MHz.
- Es compatible con la unidad de protección de memoria (MPU) y con las interfaces de depuración JTAG y FINE.
- Cuenta con un diseño y arquitectura de bajo consumo, operando desde una única fuente de alimentación de 2.7 a 3.6 V
- Dispone de memoria flash de código interno que admite versiones de hasta 2 Mbytes de ROM.
- Cuenta con memoria flash de datos interna de 8 Kbytes.

[6] Un tipo de controlador que comercializa la empresa Renesas

- Adicionalmente, tiene 384 Kbytes de SRAM interna y 4 Kbytes de RAM en espera.
- Respecto al espacio de dirección externa, dispone de buses para transferencia de datos a alta velocidad (frecuencia operativa máxima de 60 MHz), 8 áreas de CS y espacio de bus de 8 o 16 bits seleccionable por área. Además, tiene un área de SDRAM independiente de 128 Mbytes.
- Incorpora una función de cifrado y hasta 114 pines para puertos de E/S generales.
- La versión D puede operar en un rango de temperatura de -40°C a +85°C y la versión G de -40°C a +105°C.[13]

Típicamente, las áreas de CS (sigla del inglés Chip Select, en español significa "Selección de Chip") y SDRAM (sigla del inglés Synchronous Dynamic Random Access Memory, en español significa Memoria Dinámica de Acceso Aleatorio Síncrona") se utilizan para direccionar espacios de memoria externa o periféricos. El área QSPI se refiere a una interfaz *Quad SPI* (sigla del inglés *Serial Peripheral Interface*, en español significa Interfaz Periférico en Serie), que permite comunicación de alta velocidad con dispositivos de memoria flash.

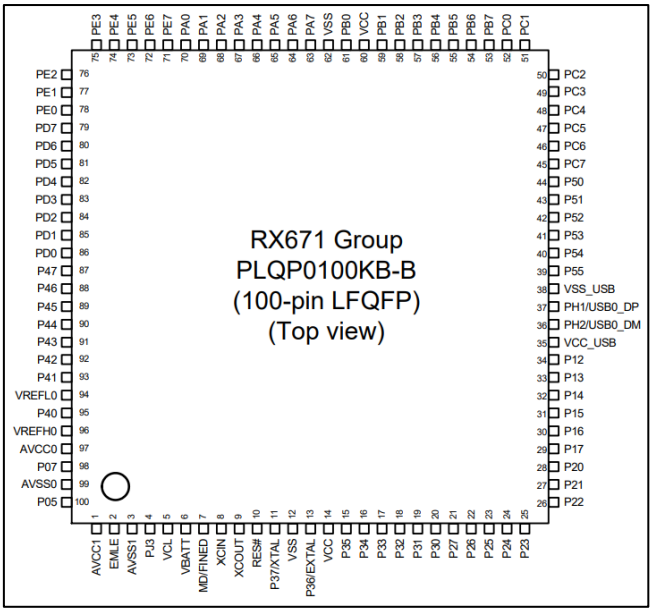


Figura 26. *Asignación de cada pin*

5.2.2. Controlador st7529

El ST7529 es un controlador y driver LSI (sigla del inglés Large Scale Integration, en español significa Integración a Gran Escala) para sistemas de pantalla de cristal líquido (LCD) de matriz de puntos gráficos a escala de grises. Este chip se puede conectar directamente con un microprocesador y acepta datos de visualización del Interfaz Periférico en Serie (SPI), 8-bit paralelo, que almacena en una memoria RAM de datos de visualización en el chip. Se realizan operaciones de lectura y escritura de la memoria RAM de datos de visualización sin un reloj operativo externo, lo que minimiza el consumo de energía. Además, contiene circuitos de

suministro de energía necesarios para impulsar el cristal líquido, lo que permite hacer un sistema de visualización con el menor número de componentes[14].

Entre las características del ST7529 se incluyen:

- Salida de circuitos del driver con 255 segmentos / 160 salidas comunes, con una resolución de pantalla máxima de 255 x 160.
- Soporta diferentes relaciones de deber para el display parcial, y permite mover la ventana parcial y desplazar los datos.
- Interfaz de microprocesador con una interfaz paralela bidireccional de 8 bits con serie 8080.
- RAM de datos de visualización en chip con una capacidad de hasta 204000 bits.
- Un rango de tensión de funcionamiento que va de 2.4 a 3.3V, y una tensión de impulsión LCD de 3.76 a 18.0V.
- Un voltaje de conducción LCD (EEPROM) para almacenar el valor de ajuste de contraste para una mejor visualización.

El controlador ST7529 ofrece dos modos para mostrar datos de escala de grises de 32 bits: el modo 2bytes 3pixels (2B3P) y el modo 3bytes 3pixels (3B3P). Para el proyecto, se emplea el modo 2B3P para visualización en escala de grises de 32 bits.

5.2.3. Placa de desarrollo RTK5RX6710C00000BJ (Target board para Rx671)

La RTK5RX6710C00000BJ es una placa de destino para el microcontrolador RX671 de Renesas (véase figura 27). Es una herramienta de evaluación y desarrollo que permite una entrada fácil a la evaluación, el prototipado y el desarrollo para este microcontrolador[15].

Las especificaciones de la placa son las siguientes:

- MCU de evaluación: Número de parte R5F5671EHDFP, paquete 100-pin LFQFP, memoria integrada 2-MB ROM, 384-KB RAM, 8-KB memoria flash de datos.
- Tamaño de la placa: 54.0 mm x 90.0 mm, grosor: 1.6 mm.
- Circuito de suministro de energía: Conector USB de 5V de entrada, IC de suministro de energía de 5V de entrada, salida de 3.3-V, cabecera de suministro de energía externa de 3.3V de entrada, 2 pines x 1.
- Corriente máxima: 200 mA.
- LEDs: indicador de alimentación verde x 1, usuario verde x 2, LED ACT verde x 1.
- USB: Micro-B.

Con esta placa se pueden realizar tareas como la programación del MCU de Renesas, la depuración del código del usuario, y la implementación de circuitos de usuario para

conmutadores y LEDs. Además, proporciona aplicaciones de muestra y muestras de código de inicialización de funciones periféricas.

El conector es de tipo USB micro-B tanto para el IDE (sigla del inglés *Integrated Development Environment*, en español significa Entorno de Desarrollo Integrado) como para el Programador Flash Renesas (RFP). Se conecta al ordenador mediante un cable USB. Cuando la fuente de energía está encendida, se suministra energía al producto al conectar el cable.

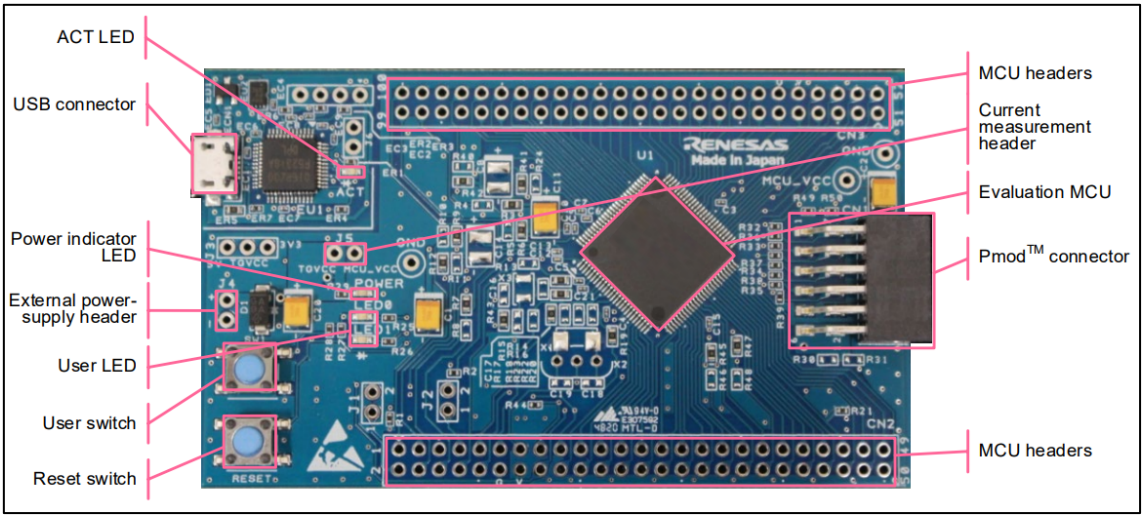


Figura 27. *Disposición de la placa (lado superior)*

5.2.4. Adaptador para LCD (o placa convertidora para LCD)

El adaptador a LCD, también conocido como placa convertidora FPC/FFC (FPC: signa del inglés *Flexible Printed Circuit*, en español significa circuitos impresos flexibles), se utiliza ampliamente en aplicaciones electrónicas de alta densidad. Esta placa es compatible con PCB de un solo lado, cuenta con 36 pines y tiene un paso de 0,5 mm a 2,54 mm. Su tamaño total es de aproximadamente 48 x 26 x 3,5 mm[16].

Es importante tener en cuenta que la placa PCB es solo un convertidor y los tamaños mencionados se miden manualmente, por lo que, puede haber un margen de error de +/- 0,1 pulgadas. La placa es fabricada por Sourcing Map y tiene un peso de aproximadamente 10 gramos.

5.2.5. Pantalla de cristal líquido Displaytech 128240

El Displaytech 128240 es una pantalla LCD con una resolución de 240 x 128 puntos (Dots). Tiene un tamaño de punto de 0.325 x 0.325 mm y un área de visualización de 92.0 mm x 53.0 mm. El módulo en sí mide 98.7 mm x 67.7 mm x 9.5 mm[17].

Este LCD utiliza tecnología FSTN (sigla del inglés *Film compensated STN*, en español significa neumático súper trenzado con compensación de película) y tiene un modo polarizador transflectivo. El ángulo de visión es de 6/12 en punto. Cuenta con retroiluminación LED de color blanco, la cual es alimentada por una fuente de alimentación externa.

El controlador del LCD es el ST7529-G. El método de conducción es de 1/144 Duty, 1/12 Bias. Duty en el contexto de la electrónica se refiere al ciclo de trabajo específico de un dispositivo de accionamiento, es decir, este término hace referencia al tiempo durante el cual un dispositivo está activado (o "conduciendo") en relación con el período total del ciclo. Por otro lado, Bias se

refiere a una corriente o voltaje aplicado para establecer un punto de operación determinado en un dispositivo electrónico, en español, Bias se traduce como polarización.

Es importante tener en cuenta que la pantalla se divide en dos partes (véase figura 28) y esto puede causar problemas. Es necesario encontrar la configuración correcta para mostrar de manera adecuada la información en el display.

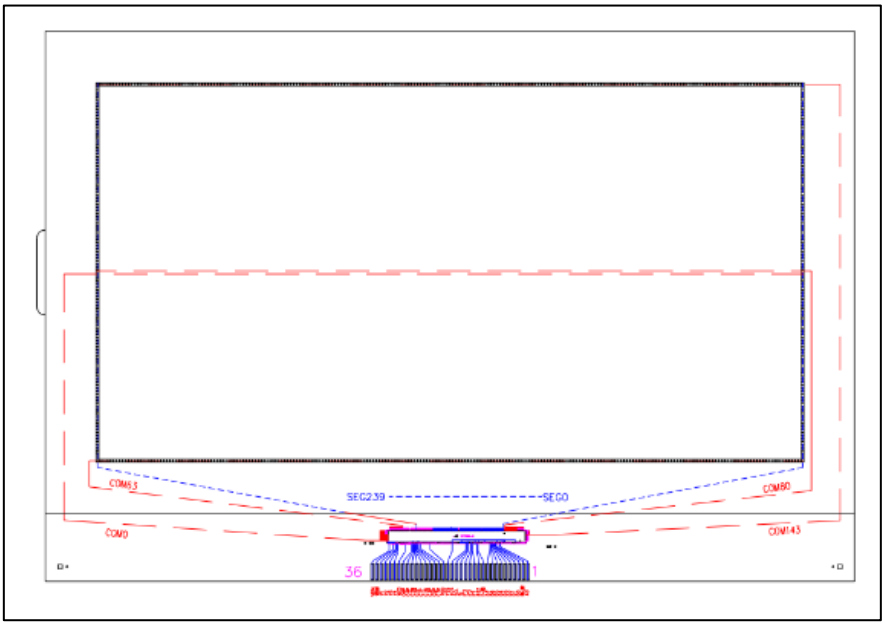


Figura 28. *División de la pantalla*

6. Integración del hardware

La integración del hardware es un aspecto vital en el desarrollo y la ejecución exitosa de cualquier proyecto tecnológico. Funciona como la plataforma fundamental para las operaciones del sistema y su manejo efectivo puede determinar el rendimiento general, la eficiencia y la adaptabilidad del sistema. Una integración de hardware bien ejecutada permite que los diferentes componentes de un sistema interactúen en una sinergia eficiente, lo que se traduce en un rendimiento optimizado y una funcionalidad ampliada.

A continuación, se abordarán los aspectos técnicos de la configuración de la Tarjeta RX671 y del controlador st7529 en el entorno de desarrollo e2Studio, siendo ambos componentes claves en el marco de la integración de hardware en este proyecto.

6.1. Configuración de la Target board para Rx671 en e2Studio

Para configurar la Tarjeta RX671 en el entorno de desarrollo e2Studio, es necesario operar en el modo de trabajo denominado "Smart Configurator". Este modo facilita una serie de herramientas y opciones que permiten una personalización y ajuste precisos del sistema de hardware.

Dentro de este marco de trabajo, existen dos aspectos principales a modificar: "Clocks" y "Components", explicados en el punto *5.1.1.2. Guía para conocer el entorno de desarrollo*. En el segundo, "Components", se realizan modificaciones más específicas que incluyen ajustes en los pines del microcontrolador y el temporizador del mismo.

6.1.1. Modificando "Clocks"

Para configurar adecuadamente la placa objetivo RX671 en el entorno de desarrollo e2Studio, es imprescindible ajustar correctamente las especificaciones relativas a los relojes del sistema, disponibles en el manual de usuario de la serie RX600 de la familia RX[18]. Estos ajustes permiten adaptar las capacidades del microcontrolador a las necesidades específicas del proyecto.

Es importante notar que el ajuste del sintetizador de frecuencia PLL (del inglés *phase-locked loop),* o Bucle de Fase Bloqueada, es un elemento clave en esta configuración. Este dispositivo genera una señal de salida cuya fase está relacionada con la señal de entrada. En el caso de la placa RX671, la señal de entrada proviene del Oscilador en Chip de Alta Velocidad (HOCO), que es la fuente principal de las señales de reloj del sistema.

Siguiendo las indicaciones contenidas en las páginas 310 y 311 del citado manual, se debe activar el HOCO y vincularlo con el sintetizador de frecuencia PLL. A partir de este vínculo, se establecen los valores de las distintas frecuencias de reloj: la del sistema interno (ICLK), las de los relojes periféricos A, B, C, y D (PCLKA, PCLKB, PCLKC, PCLKD), la del reloj de flash (FCLK), y la del reloj del bus (BCLK). Los valores adecuados para estos relojes son los que se muestran en la figura 29.

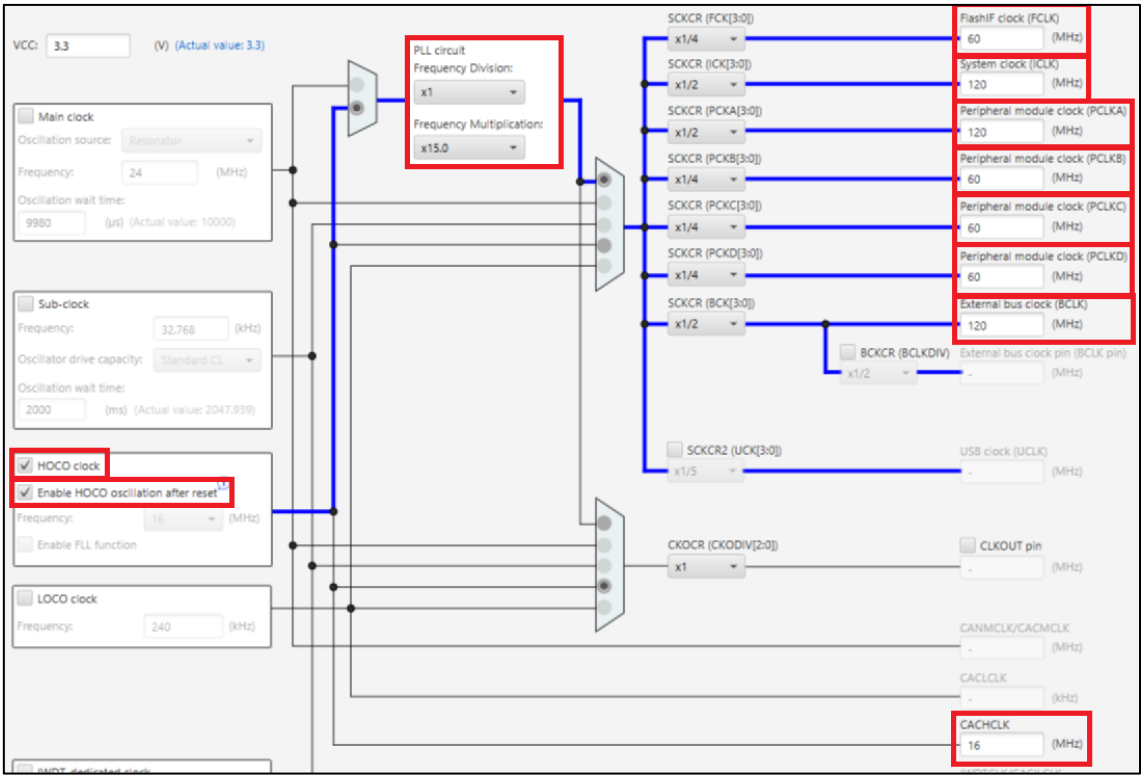


Figura 29. *Configuración de los relojes*

Los valores de las frecuencias de funcionamiento del reloj del sistema interno (ICLK), de los relojes periféricos A (PCLKA), B (PCLKB), C (PCLKC) y D (PCLKD), del reloj de flash (FCLK) y del reloj del bus (BCLK), están indicados en la página 310 del manual mencionado anteriormente (véase figura 30).

Operating frequency*1	• ICLK: 120 MHz (max)*2 • PCLKA: 120 MHz (max) • PCLKB: 60 MHz (max) • PCLKC: 60 MHz (max) • PCLKD: 60 MHz (max) • FCLK: 4 MHz to 60 MHz (for programming and erasing the code flash memory and data flash memory) 60 MHz (max) (for reading from the data flash memory) • BCLK: 120 MHz (max) • BCLK pin output: 60 MHz (max) • SDCLK pin output: 60 MHz (max) • UCLK: 48 MHz • CLKOUT pin output: 40 MHz (max) • CACCLK: Same as the clock from respective oscillators. • CANMCLK: 24 MHz (max) • RTCSCLK: 32.768 kHz • RTCMCLK: 1 kHz to 16 MHz • REMSCLK: 32.768 kHz • VBATCLK: 32.768 kHz • IWDTCLK: 120 kHz • JTAGTCK: 10 MHz (max)

Figura 30. *Valores de las frecuencias de funcionamiento*

En caso de que se decida configurar la frecuencia del ICLK para que supere los 60 MHz, es fundamental modificar el valor del registro ROMWT (véase al final de la página 311 del manual o en la figura 31). Este registro ajusta el tiempo de espera para acceder a la memoria ROM, y

requiere ser adecuado a las velocidades superiores para garantizar una correcta lectura de la memoria.

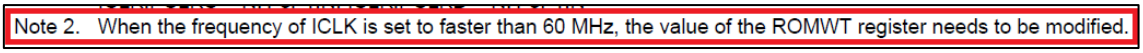


Figura 31. *Modificación del registro ROMWT al superar 60MHz de frecuencia del ICLK*

El control de las frecuencias de los distintos relojes del sistema se realiza mediante el registro SCKCR (sigla del inglés *System Clock Control Register*). La configuración de este registro permite establecer las divisiones de frecuencia de los relojes internos del sistema en relación al reloj principal. Por tanto, la velocidad de funcionamiento del microcontrolador y sus periféricos puede ser ajustada para optimizar tanto rendimiento como consumo energético.

Para alcanzar los valores de frecuencia mencionados, es crucial ajustar el SCKCR de manera que los relojes que tengan una frecuencia de 60 MHz tengan una relación de 1/4 respecto al reloj principal, y los que tengan una frecuencia de 120 MHz, una relación de 1/2.

Además, será necesario establecer la frecuencia de multiplicación del PLL en “x15.0” para obtener los resultados deseados. Por último, no olvidar ajustar el HOCO para que la CACHCLK sea de 16 MHz.

Siguiendo estos pasos, se garantiza una configuración correcta y eficiente de la placa RX671 en función de los requisitos del proyecto.

6.1.2. Modificando "Components"

En esta sección se abordará la configuración de ciertos elementos bajo la opción "Components", utilizando el modo de trabajo "Smart Configurator", más específicamente, los puertos y el temporizador (timer). Se realizará un análisis detallado de cómo ajustar estos componentes de manera eficiente para adaptarlos a los requisitos específicos del sistema que se está desarrollando.

6.1.2.1. Puertos

Para avanzar en la configuración de la placa objetivo RX671, es indispensable añadir dos nuevos componentes, tal como se describe en punto *5.1.1.2. Guía para conocer el entorno de desarrollo*. El primer componente que se debe incorporar corresponde a los puertos.

Para realizar las configuraciones necesarias para asegurar que estos puertos se ajusten a las necesidades específicas del proyecto en desarrollo, se debe seleccionar el componente "Config_PORT" que se muestra en el panel izquierdo (véase figura 32). Al hacer clic sobre este componente, los parámetros correspondientes se mostrarán en el panel de la derecha. Posteriormente, hay que seleccionar los puertos A (PORTA), C (PORTC), D (PORTD), 3 (PORT3) y 5 (PORT5).

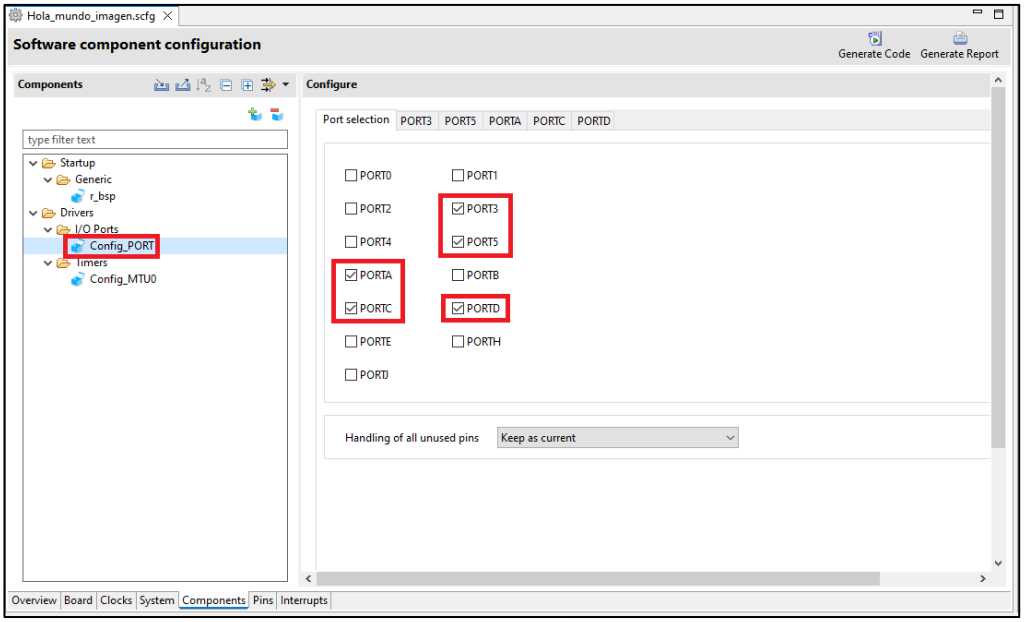


Figura 32. *Configuración de los puertos*

A continuación, se modifican las salidas en los puertos correspondientes, para que la configuración sea la misma que en este proyecto. Para ello, en la parte superior del área de desarrollo de e2Studio, aparecen los nombres de los puertos anteriormente añadidos.

Se presiona el primero de ellos, el puerto 3 (PORT3), cuyos pines P32 y P33 se corresponden a los LED0 y LED1 respectivamente. Por consiguiente, estos dos pines deben ser ajustados como salidas (Out) y establecerse inicialmente en un estado inactivo, es decir, la opción “Output 1” no se selecciona. La configuración de este puerto puede verse en la figura 33 (este es el único que se muestra, porque el resto de puertos tienen las mismas opciones).

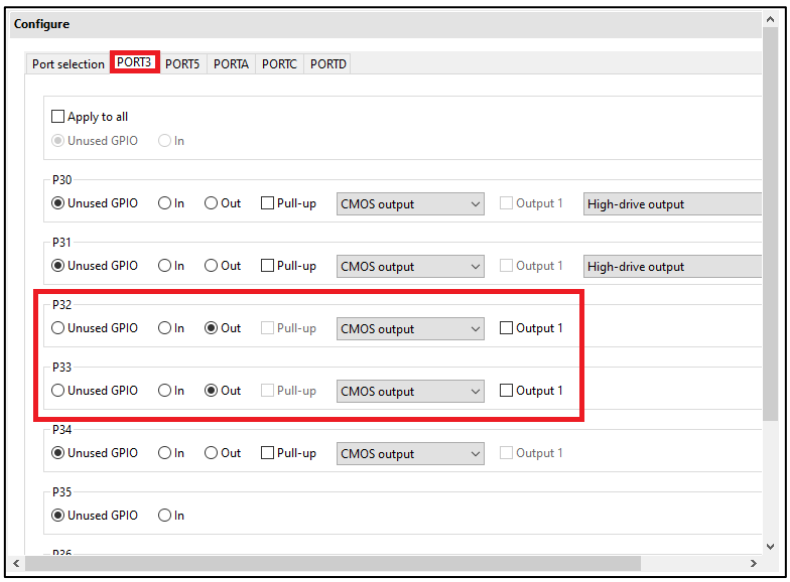


Figura 33. *Configuración del puerto 3*

El siguiente puerto que aparece es el puerto 5 (PORT5), cuyos pines P50, P51 y P52 deben establecerse como salidas (seleccionando la opción “Out”). Estos pines son usados para las funciones de escritura (WR), reinicio (RST) y lectura (RD) respectivamente. Inicialmente, estos

se dejan en un estado activo (seleccionando Output 1), ya que funcionan cuando se encuentran en estado bajo.

El siguiente paso implica el manejo del pin PA0 en el puerto A (PORTA). Este pin se configura como salida, seleccionando "Out" y se inicia en un estado inactivo, es decir, no se selecciona "Output 1".

En el puerto C (PORTC), el pin PC6 se configura como salida, seleccionando "Out". Este pin desempeña la función de Chip Select (XCS o pin de selección de chip) y se establece inicialmente en un estado activo, seleccionando "Output 1". Este pin se activa a nivel bajo, lo que permite la entrada/salida de datos. En caso de estar en nivel alto, dicha acción no sería posible.

Finalmente, en el puerto D (PORTD), los pines desde PD0 hasta PD7 deben ser configurados como salidas, seleccionando "Out". Estos pines se utilizan para las operaciones de lectura y escritura de valores y se deben iniciar en un estado inactivo, es decir, no se selecciona "Output 1".

6.1.2.2. Timer

Por último, se debe incorporar el segundo componente, que corresponde al temporizador. Para la correcta configuración del temporizador, en el caso del proyecto desarrollado en este TFG, se deben realizar las siguientes elecciones. Primero, se debe hacer clic sobre el componente creado “Config_MTU0” y se abrirán las configuraciones a la derecha (véase figura 34). A continuación, se debe escoger como “counter clear source” el TGRA0 y en “counter clock selection” la opción de PCLK/64. Después, en las configuraciones generales del registro, se modifica el valor de TGRA0 a 10 ms, como se puede observar a su derecha, el valor escogido es igual al valor actual. Por último, en el apartado de ajustes de interrupción, se selecciona la primera opción “Enable TGRA input capture/compare match interrupt (TGIA0)” y se selecciona el nivel de prioridad a 1.

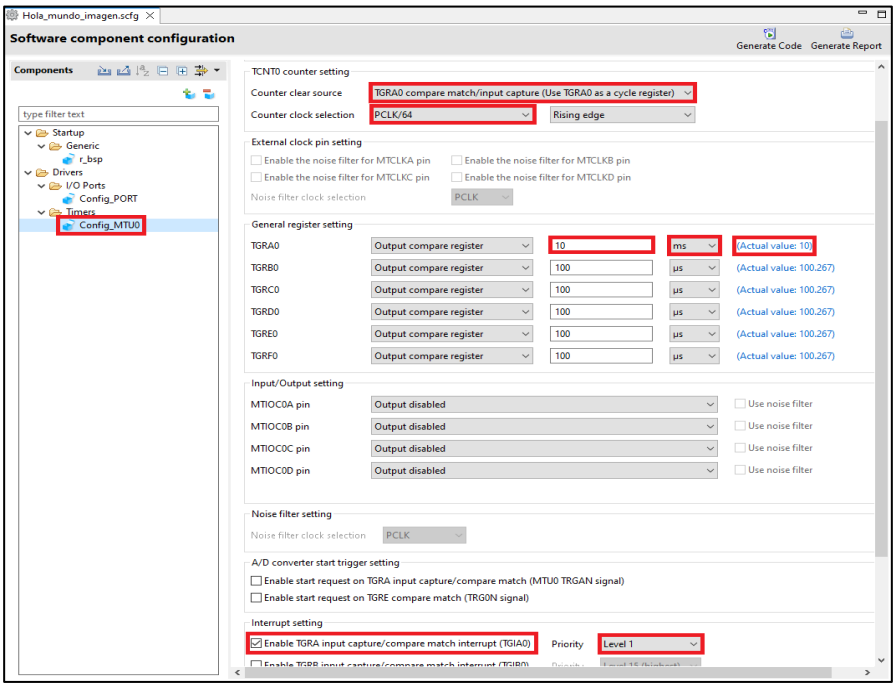


Figura 34. *Configuración del timer*

6.2. Configuración del controlador st7529 en e2Studio

Para la correcta configuración del controlador ST7529, es indispensable contar con la librería apropiada para dicho controlador, tal como se ilustra en la figura 12. En este caso, se está hablando de la carpeta "st7529", ubicada dentro de la carpeta "inc", la cual engloba todo lo necesario para la correcta utilización de este controlador.

Además, resulta crucial incorporar la biblioteca completa en el proyecto e implementar las modificaciones pertinentes para garantizar un funcionamiento óptimo. En otras palabras, se debe establecer las configuraciones correctas tanto para la pantalla que se está utilizando como para el controlador ST7529.

Para abordar las modificaciones realizadas, es preciso introducir algunos conceptos clave. Primero, es fundamental entender que existen dos tipos de elementos en este contexto: los comandos y los datos.

Los comandos son instrucciones que le dicen al controlador qué operación llevar a cabo. Estas operaciones pueden variar desde inicializar el controlador, hasta configurar parámetros específicos de la pantalla, entre otros.

Los datos, por otro lado, son la información específica que se le da al controlador para ejecutar esos comandos. Por ejemplo, si un comando es configurar el tamaño de la pantalla, los datos serían las dimensiones específicas de dicha pantalla.

Es esencial diferenciar estos dos elementos, ya que son vitales para el correcto funcionamiento del sistema.

En este sentido, se explicarán y mostrarán los comandos de mayor importancia. La tabla 2 expone cómo acceder a los comandos "Ext In" (para desactivar la instrucción de extensión) y "Ext Out" (para habilitar la instrucción de extensión). La única diferencia entre estos dos comandos es el estado del dígito D0: si está a nivel bajo, estará en "Ext In"; si está a nivel alto, estará en "Ext Out".

Ext=0 or Ext=1

Index	*Command*	*A0*	*RD*	*WR*	*D7*	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*	*Function*	*Hex*	*Parameter*
1	Ext In	0	1	0	0	0	1	1	0	0	0	0	Ext=0 Set	30	None
2	Ext Out	0	1	0	0	0	1	1	0	0	0	1	Ext=1 Set	31	None

Tabla 2. *Comandos 1, Ext = 0 y Ext = 1*

Dentro de los comandos en los que está deshabilitada la instrucción de extensión (“Ext In”), hay todos los que aparecen en la tabla 2. Como se puede observar, cada comando tiene una función asociada y un número correspondiente en hexadecimal, según los valores de los pines A0, RD, WR y desde D7 hasta D0.

Para la adecuada configuración de estos comandos, es necesario consultar el documento del controlador ST7529 (a partir de la página 38 se encuentran estas configuraciones).

Se va a poner especial atención en los comandos que se incluyen en la inicialización del proyecto: “DISCTRL”, “SLPOUT”, “OSCON”, “PWRCTRL”, “VOLCTRL”, “DISNOR”, “DISINV”, “COMSCN”, “DATSDR”, “LASET” y “CASET”.

Dentro de los comandos en los que está habilitada la instrucción de extensión (“Ext Out”), este proyecto se centrará únicamente en la inicialización del comando “ANASET”.

Index	Command	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0	Function	Hex	Parameter
1	DISON	0	1	0	1	0	1	0	1	1	1	1	Display On	AF	None
2	DISOFF	0	1	0	1	0	1	0	1	1	1	0	Display Off	AE	None
3	DISNOR	0	1	0	1	0	1	0	0	1	1	0	Normal Display	A6	None
4	DISINV	0	1	0	1	0	1	0	0	1	1	1	Inverse Display	A7	None
5	COMSCN	0	1	0	1	0	1	1	1	0	1	1	COM Scan Direction	BB	1 byte
6	DISCTRL	0	1	0	1	1	0	0	1	0	1	0	Display Control	CA	3 bytes
7	SLPIN	0	1	0	1	0	0	1	0	1	0	1	Sleep In	95	None
8	SLPOUT	0	1	0	1	0	0	1	0	1	0	0	Sleep Out	94	None
9	LASET	0	1	0	0	1	1	1	0	1	0	1	Line Address Set	75	2 bytes
10	CASET	0	1	0	0	0	0	1	0	1	0	1	Column Address Set	15	2 bytes
11	DATSDR	0	1	0	1	0	1	1	1	1	0	0	Data Scan Direction	BC	3 bytes
12	RAMWR	0	1	0	0	1	0	1	1	1	0	0	Writing to Memory	5C	Data
13	RAMRD	0	1	0	0	1	0	1	1	1	0	1	Reading from Memory	5D	Data
14	PTLIN	0	1	0	1	0	1	0	1	0	0	0	Partial display in	A8	2 bytes
15	PTLOUT	0	1	0	1	0	1	0	1	0	0	1	Partial display out	A9	None
16	RMWIN	0	1	0	1	1	1	0	0	0	0	0	Read and Modify Write	E0	None
17	RMWOUT	0	1	0	1	1	1	0	1	1	1	0	RMW end	EE	None
18	ASCSET	0	1	0	1	0	1	0	1	0	1	0	Area Scroll Set	AA	4 bytes
19	SCSTART	0	1	0	1	0	1	0	1	0	1	1	Scroll Start Set	AB	1 byte
20	OSCON	0	1	0	1	1	0	1	0	0	0	1	Internal OSC on	D1	None
21	OSCOFF	0	1	0	1	1	0	1	0	0	1	0	Internal OSC off	D2	None
22	PWRCTRL	0	1	0	0	0	1	0	0	0	0	0	Power Control	20	1 byte
23	VOLCTRL	0	1	0	1	0	0	0	0	0	0	1	EC control	81	2 bytes
24	VOLUP	0	1	0	1	1	0	1	0	1	1	0	EC increase 1	D6	None
25	VOLDOWN	0	1	0	1	1	0	1	0	1	1	1	EC decrease 1	D7	None
26	RESERVED	0	1	0	1	0	0	0	0	0	1	0	Not Use	82	0
27	EPSRRD1	0	1	0	0	1	1	1	1	1	0	0	READ Register1	7C	None
28	EPSRRD2	0	1	0	0	1	1	1	1	1	0	1	READ Register2	7D	None
29	NOP	0	1	0	0	0	1	0	0	1	0	1	NOP Instruction	25	None
30	STREAD	0	0	1	Read Data								Status Read		
31	EPINT	0	1	0	0	0	0	0	0	1	1	1	Initial code(1)	07	1 byte

Tabla 3. *Comandos 3, cuando Ext = 0*

Ext=1

Index	Command	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0	Function	Hex	Parameter
1	Gray 1 Set	0	1	0	0	0	1	0	0	0	0	0	FRAME 1 Gray PWM Set	20	16 bytes
2	Gray 2 Set	0	1	0	0	0	1	0	0	0	0	1	FRAME 2 Gray PWM Set	21	16 bytes
3	ANASET	0	1	0	0	0	1	1	0	0	1	0	Analog Circuit Set	32	3 bytes
4	SWINT	0	1	0	0	0	1	1	0	1	0	0	Software Initial	34	None
5	EPCTIN	0	1	0	1	1	0	0	1	1	0	1	Control EEPROM	CD	1 byte
6	EPCOUT	0	1	0	1	1	0	0	1	1	0	0	Cancel EEPROM	CC	None
7	EPMWR	0	1	0	1	1	1	1	1	1	0	0	Write to EEPROM	FC	None
8	EPMRD	0	1	0	1	1	1	1	1	1	0	1	Read from EEPROM	FD	None

Tabla 4. *Comandos 2, cuando Ext = 1*

6.2.1. Comandos más nombrados a lo largo del proyecto

A continuación, se describen los comandos más empleados para el desarrollo de este TFG de las tablas 3 y 4:

"DISCTRL" o Control de Pantalla (valor en hexadecimal CA) - Byte de Parámetro: 3

Este comando y los parámetros siguientes se utilizan para configurar las características relacionadas con el tiempo de visualización (véase tabla 5). Este comando debe ser seleccionado antes de usar SLPOUT. No es recomendable cambiar este comando mientras la pantalla está encendida.

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0	Función
Comando	0	1	0	1	1	0	0	1	0	1	0	
Byte de Parámetro 1 (PB1)	1	1	0	*	*	*	0	0	CLD	0	0	Proporción de división de CL, patrón de conducción F1 y F2.
Byte de Parámetro 2 (PB2)	1	1	0	*	*	DT5	DT4	DT3	DT2	DT1	DT0	Deber de conducción (Drive duty)
Byte de Parámetro 3 (PB3)	1	1	0	*	*	*	FI	LF3	LF2	LF1	LF0	Valor de conjunto inverso de FR

Tabla 5. *Comando de control de pantalla*

PB1 especifica la proporción de división de CL.

CLD: Proporción de división de CL. Se utilizan para cambiar el número de etapas de división del reloj externo o interno.

CLD=0: no dividir, CLD=1: 2 divisiones.

PB2 especifica ciclo de trabajo (proveniente del inglés *duty cycle*). Inicial: 00H

(Número de líneas de visualización)/4-1 = DT5 x 25 + DT4 x 24 + DT3 x 23 + DT2 x 22 + DT1 x 21 + DT0 x 20

En el caso de este TFG, en el datasheet del LCD se especifica que debe tener 1/144 Duty.

$$\frac{144}{4} - 1 = 35 \rightarrow en\ hexadecimal\ 23$$

$$\rightarrow en\ binario\ (1,0,0,0,1,1) = (DT5, DT4, DT3, DT2, DT1, DT0)$$

PB3 especifica el número de ciclos de línea (rango de 2 a 16) en un marco.

Número de ciclos de línea-1 = LF3 x 23 + LF2 x 22 + LF1 x 21 + LF0 x 20

FI decide el tipo de inversión de marco al final del ciclo de exploración común mientras el número de deber no es divisible por el número de ciclos de línea por marco. Por ejemplo, en la aplicación de 1/m deber y n ciclos de línea en un marco establecido, la diferencia de la elección en FI se muestra en la siguiente figura.

m = n x k + r, donde m, n, k, y r son todos números enteros, y r es el residuo de m dividido por n (r < n)."

“SLPOUT” o Modo de reposo desactivado (valor en hexadecimal 94) - No requiere un byte de parámetro ().

Este comando es para salir del MODO DE REPOSO (véase tabla 6).

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0
Comando	0	1	0	1	0	0	1	0	1	0	1

Tabla 6. *Comando para desactivar el modo de reposo*

“OSCON” u Oscilación interna activada (valor en hexadecimal D1) - No necesita byte de parámetro.

Este comando activa el circuito de oscilación interna. Es válido sólo cuando el circuito de oscilación interna CLS = ALTO (véase tabla 7).

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0
Comando	0	1	0	1	1	0	1	0	0	0	1

Tabla 7. *Comando para activar la oscilación interna*

“PWRCTRL” o Establecer control de potencia (20H) - Byte de Parámetro: 1

Este comando se utiliza para activar o desactivar el circuito de refuerzo (Booster), el regulador de voltaje, y el voltaje de referencia (véase tabla 8).

VR activa/desactiva el circuito de generación de voltaje de referencia. VR = "1": ACTIVADO, VR = "0": DESACTIVADO

VF activa/desactiva el seguidor de voltaje del circuito. VF = "1": ACTIVADO, VF = "0": DESACTIVADO

VB: Activa o desactiva el refuerzo (Booster). VB = "1": ACTIVADO, VB = "0": DESACTIVADO

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0	Función
Comando	0	1	1	0	0	1	0	0	0	0	0	
Byte de Parámetro 1 (PB1)	1	1	0	*	*	*	0	VB	0	VF	VR	Control de energía de la unidad LCD

Tabla 8. *Comando para establecer control de potencia*

“VOLCTRL” o Control de volumen electrónico (81H) - Byte de Parámetro: 2

Este comando se usa para programar el voltaje de suministro óptimo de la pantalla LCD V0 (véase tabla 9).

Con el comando VOLUP y VOLDOWN se puede ajustar el voltaje V0 y, por ende, el contraste de la pantalla LCD."

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0	Función
Comando	0	1	0	1	0	0	0	0	0	0		
Byte de Parámetro 1 (PB1)	1	1	0	*	*	VPR5	VPR4	VPR3	VPR2	VPR1	VPR0	[5:0]
Byte de Parámetro 2 (PB2)	1	1	0	*	*	*	*	*	VPR8	VPR7	VPR6	[8:6]

Tabla 9. *Comando de control de volumen electrónico*

"DISNOR" o Pantalla Normal (A6H) - No requiere un byte de parámetro

Este comando se utiliza para resaltar normalmente el área de visualización sin modificar los contenidos de la memoria RAM de datos de visualización (véase tabla 10).

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0
Comando	0	1	0	1	0	1	0	0	1	1	0

Tabla 10. *Comando de pantalla normal*

"**DISINV"** o Pantalla Invertida (A7) - No requiere un byte de parámetro

Destaca de forma inversa el área de visualización sin modificar el contenido de la RAM de datos de visualización. Este comando no invierte las áreas no visibles en caso de utilizar una visualización parcial (véase tabla 11).

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0
Comando	0	1	0	1	0	1	0	0	1	1	1

Tabla 11. *Comando de pantalla invertida*

"COMSCN" o Escaneo Común (BBH) - Byte de Parámetro: 1

Este comando se utiliza para especificar la dirección de barrido de la salida común y es útil para facilitar el cableado en el panel LCD (véase tabla 12).

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0	Función
Comando	0	1	0	1	0	1	1	1	0	1	1	
Byte de Parámetro 1 (PB1)	1	1	0	*	*	*	*	*	CD2	CD1	CD0	Dirección de barrido común

Tabla 12. *Comando de barrido común*

El Byte de Parámetro 1 (PB1) define la dirección del escaneo común (CD2, CD1, CD0). Los valores combinados de CD2, CD2 y CD0 (0, 1, 2, 3) mostrarán la pantalla de diferentes maneras, ya que la pantalla está dividida por la mitad (véase figuras 35, 36, 37 y 38).

- CD[2-0] = [0,0,0] (0→63, 64→127)

Figura 35. *Dirección del escaneo común 0*

- CD[2-0] = [0,0,1] (0→63, 127→64)

Figura 36. *Dirección del escaneo común 1*

- CD[2-0] = [0,1,0] (63→0, 64→127)

Figura 37. *Dirección del escaneo común 3*

- CD[2-0] = [0,1,1] (63→0, 127→64)

Figura 38. *Dirección del escaneo común 4*

Cuando se selecciona 1/128 (altura de la pantalla) para el deber de visualización, los pines y la salida común se escanean en el orden que se muestra a continuación en la tabla 13.

CD2	CD1	CD0	Dirección de barrido común					
			COM0 pin		COM63 pin	COM64 pin		COM127 pin
0	0	0	0	→	63	64	→	127
0	0	1	0	→	63	127	→	64
0	1	0	63	→	0	64	→	127
0	1	1	63	→	0	127	→	64

Tabla 13. *Pines y la salida de barrido común*

Tras ajustar la pantalla utilizando el comando "COMSCN", se muestra la información en ella. Sin embargo, puede suceder que las letras o las imágenes aparecen invertidas, como si se observan en un espejo. Para corregir esta situación y garantizar una correcta visualización, se emplea el comando "DATSDR". Este comando ajusta la orientación de la información, asegurándose de que se muestre correctamente en la pantalla.

“DATSDR” o Dirección de escaneo de datos (BCH) - Byte de Parámetro: 3

Este comando configura varios parámetros en las operaciones de datos de visualización almacenados en la RAM incorporada por la MPU (véase tabla 14).

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0	Función
Comando	0	1	0	1	0	1	1	1	1	0	0	
Byte de Parámetro 1 (PB1)	1	1	0	*	*	*	*	*	C/L	CI	LI	Visualización normal/inversa de la dirección y dirección de escaneo de dirección.
Byte de Parámetro 2 (PB2)	1	1	0	*	*	*	*	*	*	*	CLR	Disposición de P1, P2, P3
Byte de Parámetro 3 (PB3)	1	1	0	*	*	*	*	*	GS2	GS1	GS0	Configuración de la escala de grises

Tabla 14. *Comando de dirección de escaneo de datos*

PB1 se utiliza para especificar la visualización normal/inversa de la dirección de línea y columna y la dirección de escaneo de la dirección.

LI: Dirección normal/inversa de la dirección de línea. LI =0: Normal, LI =1: Inversa

CI: Dirección normal/inversa de la dirección de columna. CI =0: Normal, CI =1: Inversa

C/L: Dirección de escaneo de dirección. C/L =0: En la dirección de la columna, C/L =1: En la dirección de la línea (Si se desea ver los tipos de disposición de manera visual, en el Anexo, están las figuras 180 y 181).

PB2 sirve para cambiar la disposición de P1, P2, P3 del segmento de salida según la disposición de P1, P2, P3 en el panel LCD. Este comando establecerá si se cambia o no la posición de escritura de los datos (P1, P2, P3) en la memoria de visualización (véase tabla 15).

CLR	SEG0	SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7	...	SEG254
0	P1	P2	P3	P1	P2	P3	P1	P2	...	P3
1	P3	P2	P1	P3	P2	P1	P3	P2	...	P1

Tabla 15. *Disposición de P1, P2, P3*

PB3 sirve para seleccionar el modo de visualización de la escala de grises deseado, ya sea el modo 2B3P o el modo 3B3P (véase tabla 16).

GS2	GS1	GS0	Números de escala de grises
0	0	1	Escala de grises de 32, modo 2Byte 3Pixel
0	1	0	Escala de grises de 32, modo 3Byte 3Pixel

Tabla 16. *Modo de visualización de la escala de grises*

"LASET" o Establecer dirección de línea (75H) - Byte de Parámetro: 2

Este comando especifica el área de dirección de línea cuando la MPU accede a la RAM de datos de visualización. En esta operación, se incrementan las direcciones de línea desde la línea inicial hasta la final durante el escaneo, lo que a su vez aumenta la dirección de la columna en una unidad y hace que la dirección de la línea vuelva a la inicial (véase tabla 17).

Se debe tener en cuenta que siempre debe existir un par formado por la línea inicial y final. Además, se debe mantener la condición de que la línea de inicio sea siempre menor que la final.

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0	Función
Comando	0	1	0	0	1	1	1	0	1	0	1	
Byte de Parámetro 1 (PB1)	1	1	0	SL7	SL6	SL5	SL4	SL3	SL2	SL1	SL0	Línea de Inicio (SL)
Byte de Parámetro 2 (PB2)	1	1	0	EL7	EL6	EL5	EL4	EL3	EL2	EL1	EL0	Línea Final (EL)

Tabla 17. *Comando para establecer dirección de línea*

PB1 se usa para establecer la 'Línea de Inicio' (SL).

PB2 se usa para establecer la 'Línea Final' (EL).

"CASET" o Establecer dirección de columna (15H) - Byte de Parámetro: 2

Este comando especifica el área de dirección de columna cuando la MPU accede a la RAM de datos de visualización. En este proceso, las direcciones aumentan desde la columna de inicio hasta la columna final durante el escaneo en dirección de la columna, aumentando la dirección de la línea en una unidad y retornando la dirección de la columna a la columna inicial (véase tabla 18).

Es importante destacar que siempre debe existir un par formado por la columna inicial y la columna final. Además, se debe mantener la condición de que la columna de inicio siempre sea menor que la final.

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0	Función
Comando	0	1	0	0	0	0	1	0	1	0	1	
Byte de Parámetro 1 (PB1)	1	1	0	SC7	SC6	SC5	SC4	SC3	SC2	SC1	SC0	Línea de Columna (SC)
Byte de Parámetro 2 (PB2)	1	1	0	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0	Línea Columna (EC)

Tabla 18. *Comando para establecer dirección de columna*

PB1 se usa para establecer la 'Columna de Inicio' (SC).

PB2 se usa para establecer la 'Columna Final' (EC).

"ANASET" o Configuración del circuito analógico (32H) - Byte de Parámetro: 3

Este comando se utiliza para configurar ciertos aspectos del circuito analógico de la pantalla (véase tabla 19).

	A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0	Función
Comando	0	1	0	0	0	1	1	0	0	1	0	
Byte de Parámetro 1 (PB1)	1	1	0	*	*	*	*	*	OSF2	OSF1	OSF0	Frecuencia del oscilador (OSF)
Byte de Parámetro 2 (PB2)	1	1	0	*	*	*	*	*	*	BE1	BE2	Eficiencia del potenciador (BE)
Byte de Parámetro 3 (PB3)	1	1	0	*	*	*	*	*	BS2	BS1	BS0	proporción de sesgo del LCD (Configuración de Bias)

Tabla 19. *Comando para la configuración del circuito analógico*

PB1 se usa para ajustar la frecuencia del oscilador (OSF). Hay varias opciones de frecuencia, desde 12.7 KHz hasta 25.4 KHz. Para este TFG, la frecuencia se establece en 13.2 KHz (véase tabla 20).

OSF2	OSF1	OSF0	Frecuencia (KHz)
0	0	0	12.7 (Por defecto)
0	0	1	13.2
0	1	0	14.3
0	1	1	15.7
1	0	0	17.3
1	0	1	19.3
1	1	0	21.9
1	1	1	25.4

Tabla 20. *Frecuencia del oscilador*

PB2 se utiliza para ajustar la eficiencia del potenciador (BE). La frecuencia en los condensadores del potenciador puede ser de 3K, 6K (que es el valor predeterminado), 12 KHz o 24 KHz. Para este proyecto se selecciona la primera, la frecuencia en los condensadores del potenciador de 3 KHz (véase tabla 21).

BE1	BE0	Frecuencia en los condensadores del potenciador (Hz)
0	0	3K
0	1	6K (Por defecto)
1	0	12K
1	1	24K

Tabla 21. *Eficiencia del potenciador*

PB3 se utiliza para seleccionar la proporción de sesgo del LCD, es decir, el voltaje requerido para activar la pantalla LCD. En el TFG en desarrollo, se escoge 1/12, ya que este valor de *bias* viene especificado en el datasheet del LCD (véase tabla 22).

BS2	BS1	BS0	LCD bias
0	0	0	1/14
0	0	1	1/13
0	1	0	1/12
0	1	1	1/11
1	0	0	1/10
1	0	1	1/9
1	1	0	1/7
1	1	1	1/5

Tabla 22. *Proporción de sesgo del LCD (bias)*

En general, estos ajustes permiten personalizar la forma en que la pantalla LCD funciona y se muestra, permitiéndote optimizarla para tus necesidades específicas. Recuerda, sin embargo, que cada uno de estos ajustes puede afectar al rendimiento y la vida útil de la pantalla, por lo que deben ser seleccionados con cuidado.

7. Desarrollo gráfico de widget con IconEdit

Durante una reunión en Dismuntel, se llegó a un consenso acerca de la estrategia a seguir para el desarrollo de la interfaz gráfica. Este acuerdo refleja las necesidades e intereses del proyecto para la empresa, y no se basa en preferencias individuales. En otras palabras, el diseño y la presentación de la interfaz se han desarrollado de acuerdo con directrices específicas, en lugar de decisiones personales o subjetivas.

7.1. Fondo de la interfaz general

En este contexto, existen tres categorías de representaciones gráficas para visualizar cinco valores diferentes en la pantalla. Para los valores de voltaje, hercios y kilovatios se presentarán utilizando la misma metodología de representación. Estos valores se visualizarán mediante gráficos en forma de arcos, como se ilustra en la figura 39(son los tres arcos representados en la parte superior).

En segundo lugar, se decidió que la representación gráfica más adecuada para mostrar la temperatura es un gráfico tipo velocímetro. Este formato se seleccionó por su capacidad para mostrar de manera intuitiva las variaciones de temperatura (véase la figura 39, en la parte inferior izquierda).

Finalmente, para la visualización de la cantidad de combustible disponible, se eligió una barra horizontal de llenado. Este tipo de representación gráfica ofrece una forma clara y efectiva de visualizar este tipo de datos. Se ilustra en la figura 39, es la posición inferior derecha.

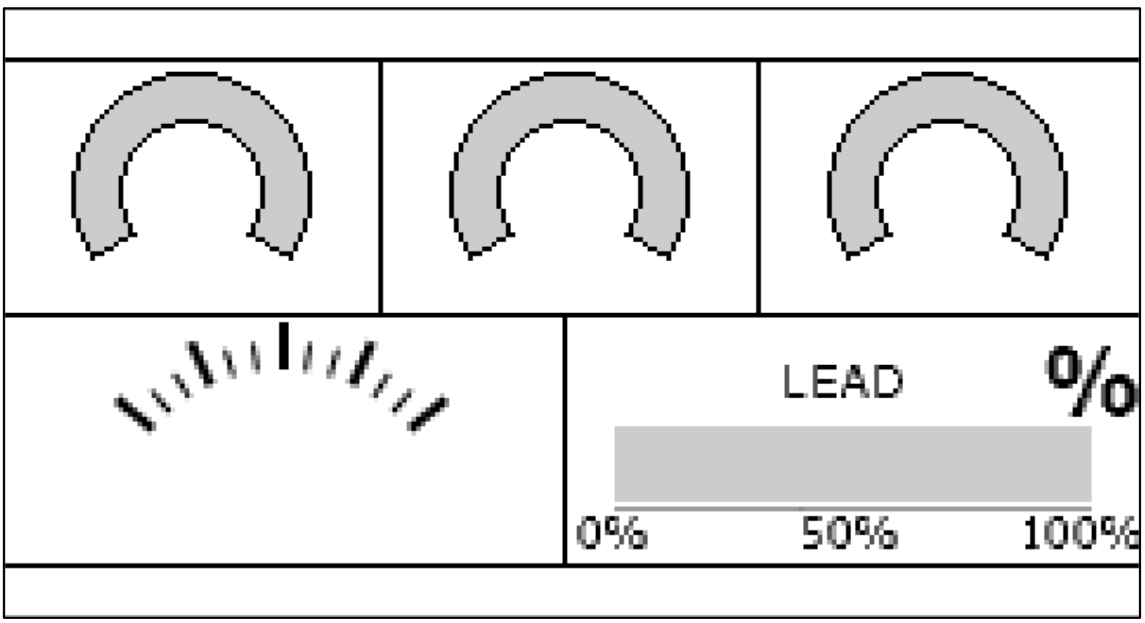


Figura 39. *Fondo sobre el que se mostrarán los valores*

A continuación, se presenta la metodología de creación de los símbolos que se utilizarán en este proyecto.

Se destaca que la empresa proporcionó una plantilla para el fondo, visible en la figura 40. Conforme a las instrucciones corporativas, se deben incorporar tres gráficos en forma de arco en la parte superior de la plantilla.

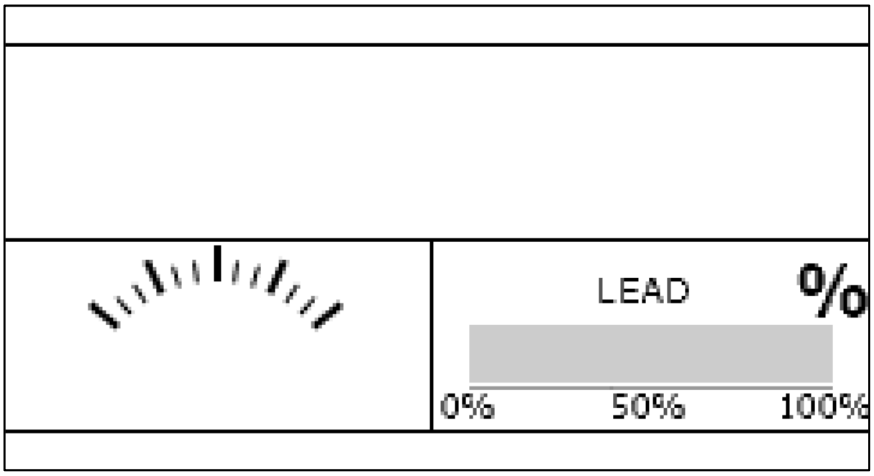


Figura 40. *Plantilla para el fondo*

La figura 41 representa tres gráficos en forma de arco que se pide implementar en la parte superior vacía de la interfaz. La diferencia es que, en lugar de visualizar los amperios (último gráfico de la figura 41), se desea controlar la potencia.

Figura 41. *Ejemplos de gráficos tipo arco*

Una vez conocido lo que se pide, se procede a desarrollar los tres gráficos superiores (ya que el resto del área gráfico está creada).

El primer paso es dividir el área disponible para los gráficos en tres segmentos iguales. Para hacer esto, se divide la anchura de la pantalla (240 píxeles) entre tres, restando dos píxeles para la separación. Esto resulta en dos segmentos de 79 píxeles y uno de 80 píxeles de ancho.

Al tener determinadas las dimensiones de cada área, se estipula que la circunferencia de cada gráfico ocupará un espacio de 51 x 51 píxeles. En consecuencia, se genera un nuevo símbolo con las dimensiones de 51 píxeles tanto de ancho como de alto.

Para ello, se toma como referencia la *tabla 1. Herramientas IconEdit* y se utiliza la herramienta denominada "Elipse hueca" para crear una elipse que ocupe la totalidad del área del nuevo símbolo. Seguidamente, se crea otra elipse un poco más pequeña dentro de la primera.

Mediante la herramienta "Dibujar línea", se trazan dos líneas a la altura deseada, rellenándose posteriormente el espacio entre las dos elipses y las dos líneas con un tono de gris claro. Para concluir el diseño del símbolo, se eliminan las partes restantes de la elipse y las líneas sobrantes. El resultado de este proceso puede observarse en la figura 42.

Cabe señalar que, para crear este símbolo, así como cualquier otro, resultan de gran utilidad las herramientas "Mostrar cuadrícula de píxeles" y "Cambiar el tamaño de píxel mostrado". Importante notar que, se ha creado un único símbolo de 51x51 píxeles que se implementará en las tres secciones que se han mencionado anteriormente (las tres áreas de la parte superior de la interfaz).

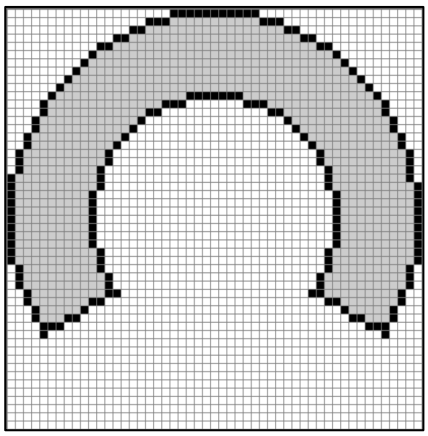


Figura 42. *Símbolo de fondo para los gráficos tipo arco*

Una vez creado este dibujo, se crea un nuevo símbolo que ocupe toda la pantalla (240x128), se añade la plantilla ofrecida por la empresa (véase figura 40) y se agrega el símbolo del nuevo arco anterior, tres veces en la parte superior, obteniendo el fondo de la interfaz. De este modo, se obtiene el fondo mostrado al comienzo de este apartado, en la figura 39.

7.2. Creación del widget tipo indicador

El siguiente paso es la creación de los widgets, componentes de la interfaz que permiten la interacción del usuario con la aplicación y proporcionan información en tiempo real.

El primer widget es la flecha del gráfico tipo velocímetro (véase figura 43). Para crear este símbolo, se calcula el área necesaria para que se vea bien y sin obstruir nada, resultando en un área de 9x26 píxeles. En este espacio, se dibuja una circunferencia de 9x9 píxeles en la parte inferior, un rectángulo de 3x15 píxeles para el cuerpo de la flecha, y una columna de 3 píxeles para la punta.

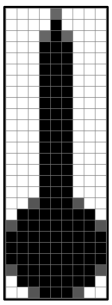


Figura 43. *Diseño del símbolo Flecha*

7.3. Creación del widget tipo arco

El segundo componente por diseñar corresponde a los símbolos que representan el nivel de llenado de los gráficos tipo arco. Para comenzar, se inicia la creación de un nuevo símbolo de dimensiones 51x51, copiando el símbolo del gráfico en arco presentado en la figura 42. Para copiar el símbolo se selecciona, con la herramienta “Editar en fotograma o mover fotograma” de la *tabla 1 Herramientas IconEdit*, el área del símbolo entero y se pega en el nuevo proyecto en blanco.

Posteriormente, en la pestaña de la barra de selección de modo de edición, se escoge la opción "Group Edit". Utilizando la herramienta "Insertar varios símbolos en el signo de intercalación", se selecciona el número de símbolos requeridos y se incorporan estos nuevos símbolos vacíos al proyecto.

En todos los nuevos símbolos, excepto en el último, se inserta el gráfico en arco copiado previamente. Para el último símbolo, se accede a la pestaña "Symbol Edit" y se crea un rectángulo negro de dimensiones 3x14, tal y como se ilustra en la figura 44.

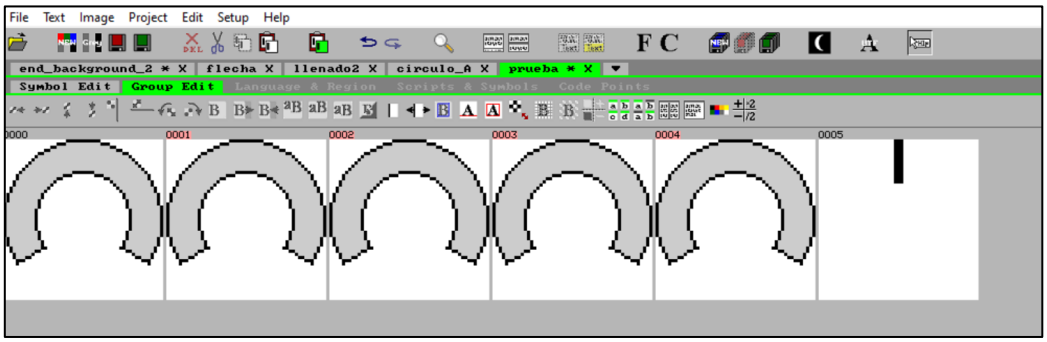


Figura 44. *Inicio en la creación del conjunto de símbolos tipo arco*

En el último símbolo, se selecciona la herramienta de “Rotar alrededor de un punto”, el centro del símbolo que es (25,25) y los grados que se quiere rotar la barra, es recomendable saber en el diseño de la figura 45, conocer el ángulo con el que se habían dibujado las líneas de los bordes.

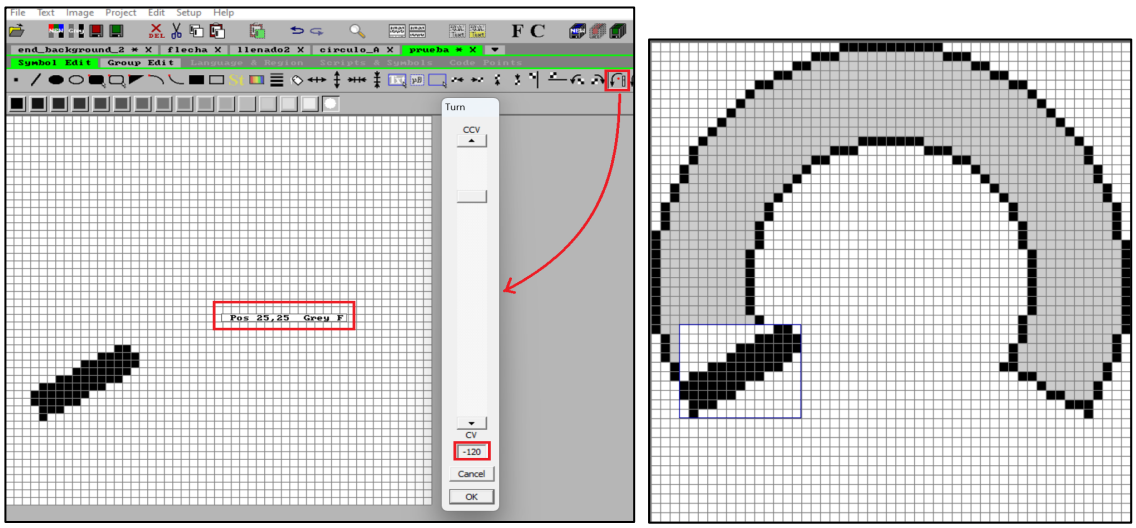


Figura 45. *Rotación de objetos en IconEdit*

Posteriormente, se copia el área que ocupa la barra usando nuevamente la herramienta “Editar en fotograma o mover fotograma”. Al pegar dicha área en el primer símbolo del proyecto (el primer gráfico tipo arco), en el resto de píxeles que ocupa la barra, se pega también el espacio en blanco que tiene alrededor y se pierde parte del contenido del simbolo. Por lo que, una vez pegada el área de la barra, se debe pintar nuevamente el fondo borrado de color gris y las elipses del contorno suprimidas.

Seguidamente, se gira la barra (el último símbolo del conjunto mostrado en la figura 46) para volver a su posición vertical y se retoca para que vuelva a tener el mismo aspecto que al inicio (3x14 píxeles). Obteniendo el resultado mostrado en la figura 46.

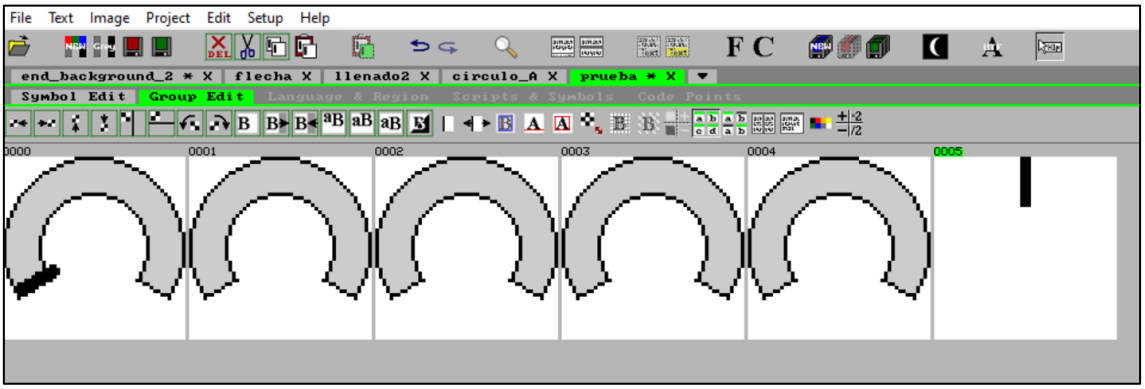


Figura 46. *Primeros pasos en la creación del conjunto de símbolos tipo arco*

El proceso se repite tantas veces como sea necesario para crear todos los símbolos de nivel de llenado deseados.

Se recomienda hacerse una tabla (véase *tabla 23*), con el número de símbolos que se van a crear, los grados de rotación en los que se les quiere hacer una representación de nivel de llenado, los nombres de cada una en hexadecimal y si ya ha sido creado el símbolo o todavía no.

Número símbolo	Grados rotados	Número hexadecima	Completado o no
0	-120	0	SI
1	-117	1	NO
2	-114	2	NO
3	-111	3	NO
4	-108	4	NO
5	-105	5	NO
6	-102	6	NO
7	-99	7	NO
8	-96	8	NO
9	-93	9	NO
10	-90	A	NO
11	-87	B	NO
12	-84	C	NO
13	-81	D	NO
14	-78	E	NO
15	-75	F	NO
16	-72	10	NO
17	-69	11	NO
18	-66	12	NO
19	-63	13	NO
20	-60	14	NO

Tabla 23. *Información sobre el conjunto de símbolos*

En el caso de la *tabla 23*, se crea un símbolo para indicar el nivel de llenado cada tres grados. El valor 40 tendrá 0º de rotación, es decir la barra en la posición inicial. Y para crear los 40 símbolos restantes del llenado en la parte derecha, se recomienda copiar estos símbolos y ponerlos en modo espejo.

Es decir, una vez se tienen los primeros cuarenta valores de llenados (véase figura 48), para hacer la otra mitad, se crean otros cuarenta nuevos símbolos vacíos con la herramienta “Insertar varios símbolos en el signo de intercalación”, se copian los ya completados y se pegan en los vacíos. Asimismo, es necesario cambiar el orden, el primero (que está completamente vacío) corresponde al último (que está completamente lleno).

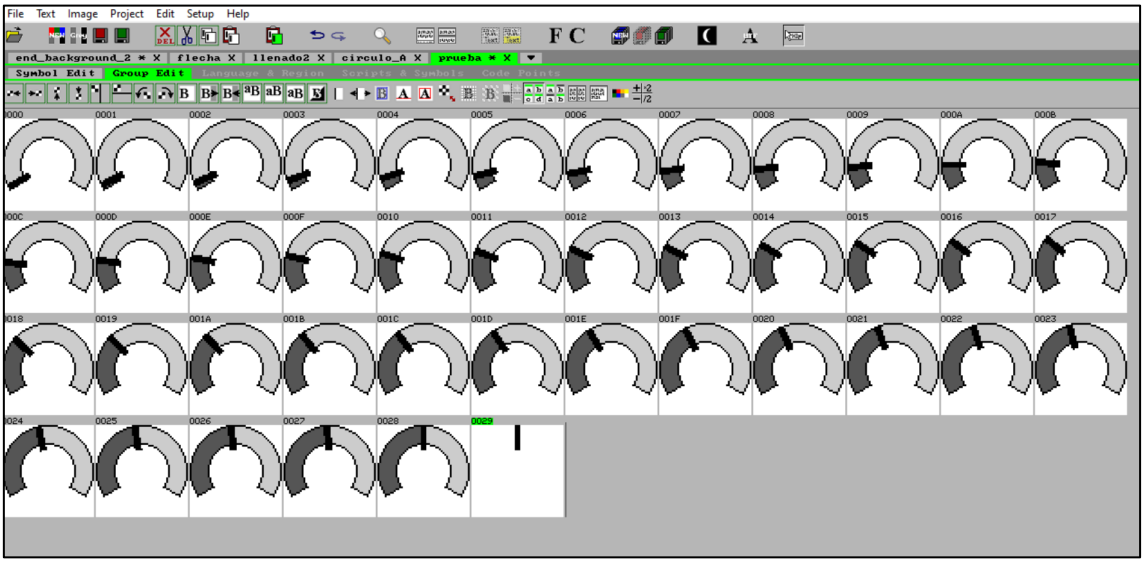


Figura 47. *Últimos pasos para la creación del conjunto de símbolos tipo arco*

A continuación, hay que usar la herramienta “Espejo” en los cuarenta últimos símbolos para que la barra esté del lado correspondiente. Por último, hay que cambiar los colores de lado, ya que por la parte izquierda se va llenando al aumentar los grados y tiene un tono gris más oscuro, y por la derecha está en gris claro que es la parte vacía.

De este modo, se obtiene el proyecto completo con ochenta niveles de llenado, como se puede ver en la figura 49.

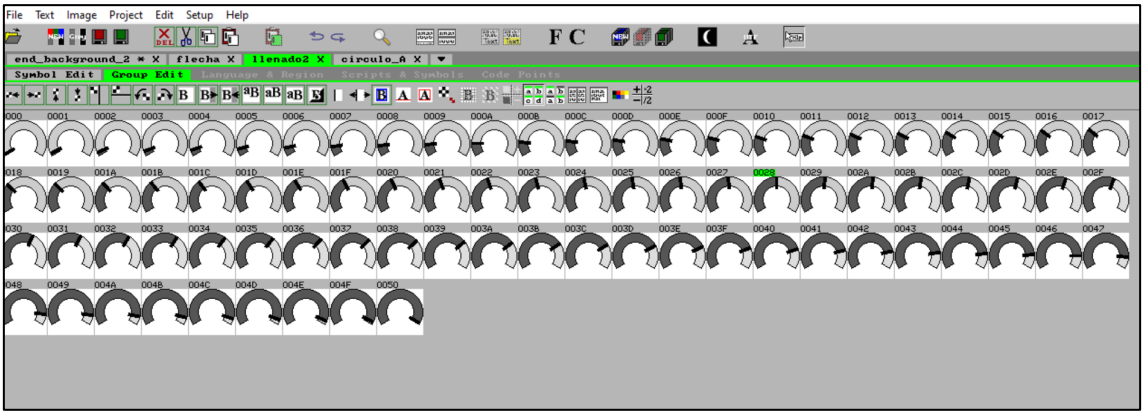


Figura 48. *Proyecto final del conjunto de símbolos tipo arco*

Con la creación de estos símbolos, se cuenta con los elementos necesarios para proceder con la siguiente fase del proyecto.

La finalidad del sistema de gestión de alarmas que se ha desarrollado en este proyecto es mostrar al usuario, de una forma fácil y organizada, todas las alarmas del sistema, representando de manera intuitiva cuales están activas y cuáles no. Para ello, se crea e incluyen en la plantilla la pantalla destinada a la gestión de alarmas, como se puede observar, por ejemplo, en la figura 107.

8. Programación e integración

Esta sección aborda un aspecto clave en la comprensión del proyecto: la programación software y la estructura seguida a lo largo del desarrollo práctico por parte de la alumna.

Este segmento se desglosa en dos apartados esenciales. En el primero se analizan los archivos de mayor relevancia que componen el software.

Este análisis permite una comprensión más profunda del proyecto, no solo desde una perspectiva técnica sino también desde el punto de vista del proceso de aprendizaje y desarrollo de habilidades en el ámbito de la programación software. Así, se facilita la comprensión de la relación entre la teoría y la práctica en el contexto de un proyecto real y concreto.

8.1. Archivos configurados durante el desarrollo del proyecto

En este apartado, se presenta una visión global del software del proyecto, poniendo especial atención a los componentes críticos que lo conforman. Los elementos clave del proyecto se clasifican principalmente en dos categorías: los archivos de cabecera, también conocidos como "header files" (archivos con la extensión “.h”), y los archivos de origen o "source files" (archivos con la extensión “.c”).

En el desarrollo de cualquier programa, estos archivos son esenciales. Los archivos de cabecera contienen definiciones y prototipos de funciones que ofrecen una interfaz para los archivos de origen, mientras que estos últimos albergan la lógica de programación principal y las implementaciones de funciones.

Los archivos mencionados, la gran mayoría, forman parte de la librería del proyecto, y dos de ellos, es el archivo del proyecto y su archivo de cabecera, siendo estos elementos de vital importancia en el desarrollo del software. En los siguientes subapartados, se profundizará en cada uno de estos componentes para entender mejor su función y su rol dentro del proyecto global.

Se seguirá un enfoque sistemático para la presentación y análisis de estos archivos. Primero, se examinarán los archivos de cabecera “sgtypes.h”, “gdispcfg.h” y “Hola_mundo_imagen.h” (corresponde al archivo de cabecer del proyecto principal). A continuación, se procederá a explorar los archivos de origen “ghwinitctrl.c” y “bussim.c”. Se muestra en la figura 49, la jerarquía de carpetas y archivos que componen el proyecto, en rojo los archivos sobre los que se hará mención en los puntos siguientes.

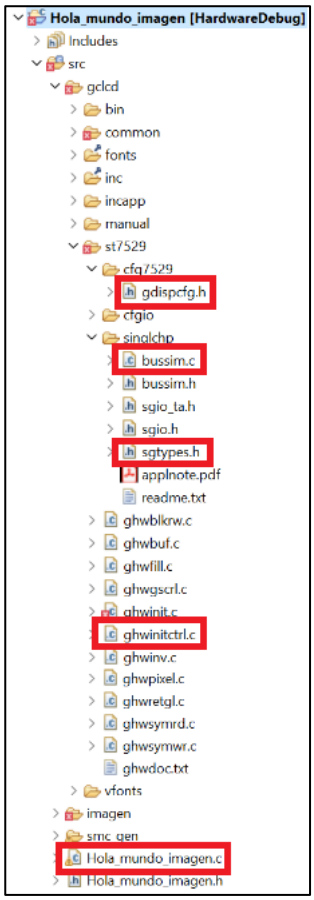


Figura 49. *Jerarquía de carpetas y archivos que componen el proyecto.*

Finalmente, se tratará el archivo del proyecto principal, el núcleo que coordina y gestiona todas las partes y funciones del software. Este archivo, que forma parte integral de la estructura del proyecto, es la base desde la cual se lanza y opera el programa en su totalidad.

8.1.1. Archivo de cabecera "sgtypes.h"

En este fichero se incluyen definiciones esenciales de ciertos tipos de datos, proporcionando una estructura sólida para el desarrollo del software. Estas definiciones son fundamentales para asegurar una gestión eficaz y precisa de los datos en el proyecto. A continuación, se detallan dichas definiciones:

"SGUCHAR" es el nombre que se le da a uint8_t, un tipo de dato entero sin signo de 8 bits. Este tipo de dato puede contener valores desde 0 hasta 255.

int8_t está definido como "SGCHAR", simboliza un tipo de dato entero con signo de 8 bits. Los valores que puede asumir este tipo de dato van desde -128 hasta 127.

"SGUINT", correspondiente a uint16_t, es un tipo de dato entero sin signo de 16 bits. Este tipo de dato puede almacenar valores desde 0 hasta 65,535.

"SGINT" es la denominación que se le da a int16_t, dato entero con signo de 16 bits, cuyo rango de valores posibles es de -32,768 hasta 32,767.

"SGULONG", es el nombre que asignado a uint32_t, dato entero sin signo de 32 bits, cuyo rango de valores va desde 0 hasta 4,294,967,295.

"SGLONG", correspondiente a int32_t, es un tipo de dato entero con signo de 32 bits. Este tipo de dato puede contener valores que van desde -2,147,483,648 hasta 2,147,483,647.

"SGBOOL", que corresponde a uint8_t, es generalmente utilizado para representar un valor booleano, donde 0 se toma como falso y cualquier valor no nulo se toma como verdadero.

La importancia de estas definiciones de tipos de datos reside en la necesidad de manejar datos de diferentes tamaños y signos en el proyecto. La elección del tipo de dato adecuado es crucial para asegurar el funcionamiento correcto del software, ya que cada tipo de dato tiene un rango específico de valores que puede representar. Utilizar el tipo de dato incorrecto puede dar lugar a errores o comportamientos no deseados. Además, el uso de estas definiciones de tipos de datos mejora la legibilidad del código, lo que facilita su mantenimiento y depuración.

8.1.2. Archivo de cabecera "gdispcfg.h"

El presente fichero, al igual que el anterior, se categoriza como un archivo de cabecera. Sin embargo, se distingue por no estar exclusivamente dedicado a las definiciones de tipos de datos. En lugar de eso, incorpora una abundancia de definiciones, que, de acuerdo con los requerimientos del proyecto particular, pueden precisar de modificación, habilitación o deshabilitación.

Centrándonos en el TFG en curso, es importante poner de manifiesto cinco definiciones que requieren ser modificadas.

La primera de ellas concierne al tamaño del LCD, el cual, por defecto en la librería, se establece en 255 de ancho y 160 de alto, parámetros comunes en la mayoría de las pantallas. Sin embargo, el proyecto actual opera con una pantalla de 240x128 píxeles, lo cual obliga a una reconfiguración de las dos primeras definiciones del archivo de la siguiente forma: "#define GDISPW 240U" y "#define GDISPH 128U".

La segunda involucra el desplazamiento de la posición de la pantalla hacia abajo y hacia la derecha. Este ajuste surge como resultado de la discordancia entre el controlador, optimizado para una pantalla de 255x160, y el tamaño real de la pantalla utilizada en este proyecto, que es menor, como se muestra en la figura 50. Por ende, la sección de la pantalla que corresponde al área blanca mostrada en esta figura (la pantalla de 240x128), no se visualizaría adecuadamente. A causa de la discrepancia de tamaño, las operaciones de visualización, como mostrar imágenes o escribir, comienzan desde el primer píxel en la esquina superior izquierda (que, al estar en zona gris, no se muestran por pantalla).

Por lo tanto, es necesario un desplazamiento de 15 píxeles en x y de 16 píxeles en y para una visualización correcta. No obstante, debido a que en el eje x cada 3 píxeles se interpreta como 1 (consulte la figura 179 del Anexo), se divide este valor entre tres. En consecuencia, se procede a modificar las dos definiciones correspondientes de la siguiente manera: "#define GHW_YOFFSET 16U" y "#define GHW_XOFFSET 5U". Con esta configuración, la visualización en la pantalla se realizará de manera correcta.

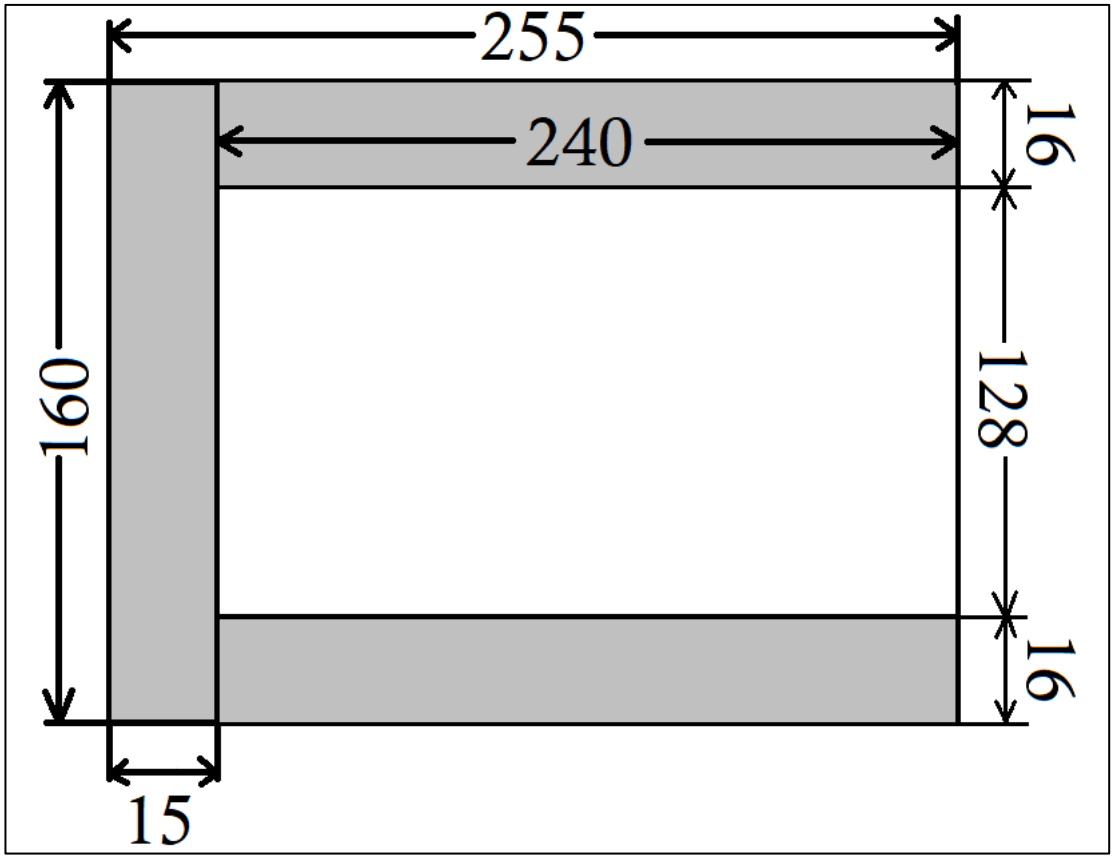


Figura 50. *Desplazamiento necesario para ver por la pantalla 240x128 correctamente.*

El siguiente elemento por modificar está relacionado con el modo en el que se visualiza la pantalla. Tal como se observó en el punto *6.2. Configuración del controlador st7529 en e2Studio*, sección *"COMSCN"*, para este tipo particular de pantalla con este controlador, es necesario activar el tercer modo, es decir, CD[2-0] = [0,1,0] (63→0, 64→127). Para efectuar esta transición, se requiere deshabilitar la configuración por defecto y habilitar la tercera opción (véase figura 51 para más detalles).

```
208     /*#define GHWCOLSCAN_LUHU*/     /* 0->63, 64-127 */
209     /*#define GHWCOLSCAN_LUHD*/       /* 0->63, 127-64 */
210     #define GHWCOLSCAN_HULD     /* 64-127, 63-0 */
211     /*#define GHWCOLSCAN_HDLD*/     /* 127-64, 63-0 */
```

Figura 51. *Modo de visualización con el que se trabaja en el archivo de cabecera "gdispcfg.h".*

Sin embargo, la última modificación (habilitar el modo de visualización "GHWCOLSCAN_HULD", descomentando la línea 210, visto en la figura 50) introduce una complicación: todos los valores se ven invertidos. En otras palabras, empiezan en la última fila y se ve en modo espejo. Para garantizar una visualización correcta, se hace necesario activar la definición del modo espejo vertical. Por lo tanto, se precisa habilitar la siguiente definición: "#define GHW_MIRROR_VER".

Por último, dado que las dimensiones de la pantalla son menores a las inicialmente establecidas, es preciso modificar el último valor, tal como se puede observar en la figura 51. Como se aprecia en dicha figura, cuando el modo de rotación no está activo (como en el caso

del presente proyecto), se activan dos variables: "GHW_Y_GABSTART" y "GHW_Y_GABEND". Sin embargo, para evitar errores de visualización, es recomendable desactivar ambas variables.

```
199     #ifdef GHW_ROTATED
200       #define GHW_Y_GABSTART GDISPW/2U        /* Set di
201       #define GHW_Y_GABEND   (160U-GDISPW/2U) /* Set di
202     #else
203       /*#define GHW_Y_GABSTART (160/2U)*/        /* Set
204       /*#define GHW_Y_GABEND  (160U/2U)*/  /* Set displ
205     #endif
```

Figura 52. Desactivación de las variables: "GHW_Y_GABSTART" y "GHW_Y_GABEND".

8.1.3. Archivo de cabecera "Hola_mundo_imagen.h"

En este archivo, que es el archivo de cabecera del proyecto principal, incluye la configuración de los puertos usados como leds, los valores de los leds (1 encendido y con 0 apagado), la configuración del interruptor, los valores de los botones y los puertos creados y configurados en el "Smart Configurator" del punto *6.1.2.1. Puertos*.

Para el correcto funcionamiento de los puertos configurados es necesario incluir "#include "r_smc_entry.h"" en el proyecto general "Hola_mundo_imagen.c" y en el fichero "r_smc_entry.h" incluir "#include "Config_PORT.h", como se muestra en la figura 53.

```
//-----------------------LEDS-----------------------
/* General Values */
#define LED_ON              (0)
#define LED_OFF             (1)

/* LED port settings */
#define LED0                (PORT3.PODR.BIT.B2)
#define LED1                (PORT3.PODR.BIT.B3)

//----------------------BUTTONS----------------------
/* General Values */
#define SW1_PUSH            (1)
#define SW1_RELEASE         (1)

/* Switches */
#define SW1                 (PORT3.PIDR.BIT.B4)

/* Conexiones manuales Display */
#define LCD_A_0             (PORTA.PODR.BIT.B0)     //CD
#define LCD_NWR             (PORT5.PODR.BIT.B0)     //NWR - /WR
#define LCD_D_0             (PORTD.PODR.BIT.B0)
#define LCD_D_1             (PORTD.PODR.BIT.B1)
#define LCD_D_2             (PORTD.PODR.BIT.B2)
#define LCD_D_3             (PORTD.PODR.BIT.B3)
#define LCD_D_4             (PORTD.PODR.BIT.B4)
#define LCD_D_5             (PORTD.PODR.BIT.B5)
#define LCD_D_6             (PORTD.PODR.BIT.B6)
#define LCD_D_7             (PORTD.PODR.BIT.B7)
#define LCD_Port_D          (PORTD.PODR.BYTE)
#define LCD_Port_D_Confi    (PORTD.PDR.BYTE)        //Para configurar
#define LCD_NRD             (PORT5.PODR.BIT.B2)     //NRD - /RD
#define LCD_NRST            (PORT5.PODR.BIT.B1)     //NRST - /RST
#define LCD_NRST_Confi      (PORT5.PDR.BIT.B1)      //Para configurar
#define LCD_NCE             (PORTC.PODR.BIT.B6)     //XCS - NCE - /CS
```

Figura 53. Definiciones del archivo de cabecera "Hola_mundo_imagen.h".

8.1.4. Archivo de origen "ghwinitctrl.c"

A partir de la línea 99 de este archivo, se encuentran las definiciones de los comandos vistos en el punto *6.2. Configuración del controlador st7529 en e2Studio*, el nombre de estos no coincide con los anteriores, pero es fácil guiarse por los valores en hexadecimal y las definiciones en verde (véase figura 54).

```
 99          /****************** Display controller commands and initialization *************/
100
101          /* ST7529 controller commands ( 1 byte ) */
102          #define GMODE_EXT0        0x30U
103          #define GMODE_EXT1        0x31U
104
105          #define GCTRL_ON          0xAFU /* Display on */
106          #define GCTRL_OFF         0xAEU /* Display off */
107          #define GCTRL_NORM_REV    0xA6U /* Normal/Reverse display | (0-1) */
108
109          #define GCTRL_SHL         0xBBU /* Norm/reverse SHL  */
110          #define GCTRL_DISPCTRL    0xCAU /* Display control + 3 bytes */
111
112          #define GCTRL_SLEEP       0x94U /* Power save (Sleep) mode | (0-1) Normal,sleep */
113
114          #define GCTRL_YADR        0x75U /* Set Y adr (line)  + 2 bytes */
115          #define GCTRL_XADR        0x15U /* Set X adr (3 pixel column) + 2 bytes */
116
117          #define GCTRL_DATSDR      0xBCU /* normal, reverse scan directions */
118
119          #define GCTRL_RAMWR       0x5CU /* Enable ram write */
120          #define GCTRL_RAMRD       0x5DU /* Enable ram read */
121          #define GCTRL_RMW_STRT    0xE0U /* Set read modify write start */
122          #define GCTRL_RMW_END     0xEEU /* Set read modify write end */
123
124          #define GCTRL_OSC_ON      0xD1U /* Turn oscillator on */
125          #define GCTRL_OSC_OFF     0xD2U /* Turn oscillator off */
126
127          #define GCTRL_POWER       0x20U /* Power-Ctrl VC,VR,VF + 1 byte */
128          #define GCTRL_CONTRAST    0x81U /* Set reference voltage + 2 bytes */
129
130          #define GCTRL_ANASET      0x32U /* Adjust booster and oscillator */
131          #define GCTRL_SWINIT      0x34U /* Software init */
```

Figura 54. *Definiciones de los comandos en el archivo de origen "ghwinitctrl.c".*

Por otro lado, a partir de la línea 180 aparecen la inicialización del controlador. Este es uno de los puntos más importantes para el buen funcionamiento de todo el proyecto, es necesario tomar el tiempo necesario en leer todas las especificaciones de los componentes hardware (véase punto *5.2. Hardware*) para que la visualización sea la óptima.

En las inicializaciones mencionadas, hay que modificar los datos que acompañan a los componentes, o a los Byte de Parámetro X vistos en el punto *6.2. Configuración del controlador st7529 en e2Studio*. A continuación, se activa el modo "Ext In" y se especifican una a una las modificaciones a realizar.

El primer comando es "GCTRL_DISPCTRL" que es equivalente a "DISCTRL" (para guiarse se puede visualizar la tabla 5). Como se especificó en el punto *6.2. Configuración del controlador st7529 en e2Studio,* es necesario que vaya acompañado con otros tres datos. El primero de ellos especifica la proporción de división de CL (clock), que tendrá valor 0, ya que no se quiere que el número de etapas de división del reloj se dividida. El siguiente parámetro especifica el ciclo de trabajo (proveniente del inglés duty cycle), como se comentó, en este proyecto se requiere trabajar con 1/144 Duty (especificado en el datasheet del LCD). El último, especifica el número de ciclos de línea y en este caso, el valor será 0, ya que cuanto mayor es el valor más claro se ve la pantalla. Se puede observar en la figura 55, en la línea 184 la manera en la que se llama al comando y las tres líneas siguientes la manera en la que se especifican los parámetros.

Los siguientes comandos no requieren de ningún parámetro adicional, por lo que iría el comando "GCTRL_SLEEP" y a continuación el "GCTRL_OSC_ON. Estos comandos se pueden observar en la figura 55, en las líneas 189 y 190.

El segundo comando a modificar es "GCTRL_POWER" que equivale a "PWRCTRL". Este comando es usado dos veces seguidas. El comando se repite y aparece en las líneas 192 y 194, de la figura 55, con el respectivo parámetro de cada comando en la línea siguiente (después de cada uno). El único parámetro que tiene este dato es información acerca del circuito de refuerzo (Booster o VB), del regulador de voltaje (VF) y del voltaje de referencia (VR), como se muestra

en la tabla 8. La primera vez que se llama se activa VB, para ello es necesario pasar el valor en hexadecimal ocho, que activa el bit correspondiente. La segunda vez que se llama, se activan el resto de bits (para que estén los tres activados) enviando así el valor B.

El siguiente es “GCTRL_CONTRAST” también conocido como “VOLCTRL”. Requiere dos parámetros, que regulan el contraste del generador (es ajustado en otro comando más adelante), el primero de ellos tendrá el valor “3D” y el segundo “04”. Se puede observar en la figura 55, en la línea 197 la manera en la que se llama al comando y las dos líneas siguientes la manera en la que se especifican los parámetros.

A continuación, está el comando “GCTRL_NORM_REV” que equivale a “DISNOR”, que como ya se ha visto anteriormente, no requiere ningún parámetro y se utiliza para resaltar normalmente el área de visualización (véase figura 55, línea 202).

```
168  /****************** initialization table *******************/
169
170  typedef struct
171     {
172     SGUCHAR index_dly;
173     SGUCHAR value;
174     } SETUP_REGS;
175
176  #define CMD    0x00U
177  #define DAT    0x80U
178  #define DLYMSK (~DAT)
179
180  static GCODE SETUP_REGS initdata[] =
181     {
182     {CMD,GMODE_EXT0},    /* ext =0 */
183
184     {CMD,GCTRL_DISPCTRL},
185     {DAT,0x00U},          /* CL=x1 */
186     {DAT,0x23U},          /* (144/4 -1) Duty = 144 */
187     {DAT,0x00U},          /* Fr inverse set */
188
189     {CMD,GCTRL_SLEEP},   /* sleep out */
190     {CMD,GCTRL_OSC_ON},  /* Start oscillator */
191
192     {CMD,GCTRL_POWER},   /* Turn booster on */
193     {DAT | 3U,0x08U},      /* VB = 1 first */
194     {CMD,GCTRL_POWER},   /* Turn ref voltage and voltage follower on */
195     {DAT,0x0BU},          /* VB,VF,VR */
196
197     {CMD,GCTRL_CONTRAST},/* Set contrast generator (adjusted later) */
198     {DAT,0x3dU},
199     {DAT,0x04U},
200
201     #ifdef GHW_INVERSE_DISP
202     {CMD,GCTRL_NORM_REV|0U}, /* Inverse display */
203     #else
204     {CMD,GCTRL_NORM_REV|1U}, /* Normal display */
```

Figura 55. *Inicializaciones del controlador en el archivo de origen “ghwinitctrl.c” (primero).*

El próximo comando es “GCTRL_SHL”, también conocido como “COMSCN”, que especifica la dirección de barrido de la salida común. En el anterior subapartado *8.1.2. Archivo de cabecera “gdispcfg.h”*, se ha mencionado que se ha activado la tercera opción (valor 2), al estar definido “GHWCOLSCAN_HULD” entrará en la opción correspondiente. Por ello, no es necesario modificar nada. Es el comando de la línea 208 de la figura 56.

Seguidamente, el comando “GCTRL_DATSDR” corresponde a “DATSDR”, que va acompañado de tres parámetros (véase figura 56, línea 219). El primero, se utiliza para especificar la visualización normal/inversa de la dirección de línea y columna y la dirección de escaneo de la dirección. En este caso, se utilizan direcciones invertidas de ambas (ya que está activado el modo “GHWCOLSCAN_HULD”). El siguiente dato, establece cambia o no en la posición de escritura de los datos (P1, P2, P3) en la memoria de visualización. En el proyecto en

desarrollo, tendrán valor 1. El último parámetro, es con el que se selecciona el modo de visualización de la escala de grises, que tendrá valor 1 ya que la pantalla utiliza el modo de escala de 2 byte, 3 píxeles. Estos valores son específicos para esta pantalla, controlador y modo “GHWCOLSCAN_HULD”, con estos, se intenta corregir la visualización de la pantalla.

El siguiente no requiere de datos que lo acompañen, y es el comando que activa el modo "Ext Out", se utiliza en la librería como “GMODE_EXT1”. Se puede ver en la línea 230 de la figura 56.

El último comando de la inicialización que tiene parámetros a modificar es “GCTRL_ANASET” que se corresponde con “ANASET” (véase figura 56, línea 231). El primer dato que lo acompaña se usa para ajustar la frecuencia del oscilador y debe tener el valor 1, ya que se corresponde a una frecuencia de 13.2 KHz. El segundo es el que ajusta la eficiencia del potenciador, se establece en cero para trabajar con una frecuencia en los condensadores del potenciador de 3 KHz. El último parámetro es el que selecciona la proporción de sesgo del LCD (bias), que en este caso, la pantalla trabaja con 1/12 bias.

Para finalizar la inicialización, se llama al comando “GCTRL_SWINIT”, que inicializa el software del dispositivo. Se activa el comando “GMODE_EXT0” y, por último, se llama al comando “DISON”, que activa la pantalla. Estos dos comandos corresponden a las líneas 236, 237 y 238 de la figura 56.

```
204             {CMD,GCTRL_NORM_REV|1U}, /* Normal display */
205             #endif
206
207             /* Column scan segment order (depends on display module layout) */
208             {CMD,GCTRL_SHL},
209             #if (defined GHWCOLSCAN_LUHU)
210             {DAT,0x00U},                    /* 0->63, 64-127 */
211             #elif (defined GHWCOLSCAN_LUHD)
212             {DAT,0x01U},                    /* 0->63, 127-64 */
213             #elif (defined GHWCOLSCAN_HULD)
214             {DAT,0x02U},                    /* 64-127, 63-0 */
215             #else        /*GHWCOLSCAN_HDLD*/
216             {DAT,0x03U},                    /* 127-64, 63-0 */
217             #endif
218
219             {CMD,GCTRL_DATSDR},   /* Data scan direction */
220             {DAT,MY_BIT|MX_BIT},  /* x,y scan order */
221
222             /* "R,G,B" - "B,G,R" mirroring (depends on both screen layout, x mirror and r
223             #ifdef GHW_COLOR_SWAP
224             {DAT,(MX_COLOR_SWAP & (~MX_SWAP))},
225             #else
226             {DAT,(MX_COLOR_SWAP & MX_SWAP)},
227             #endif
228             {DAT,1U},           /* 32 ey scale, 2 byte, 3 pixel mode (do not modify) */
229
230             {CMD,GMODE_EXT1},  /* ext =1 */
231             {CMD,GCTRL_ANASET},
232             {DAT,0x01U},
233             {DAT,0x00U},
234             {DAT,0x02U},
235
236             {CMD,GCTRL_SWINIT},
237             {CMD,GMODE_EXT0},   /* ext =0 */
238             {CMD,0xafU}
239             };
240
```

Figura 56. *Inicializaciones del controlador en el archivo de origen “ghwinitctrl.c” (segundo).*

8.1.5. Archivo de origen “bussim.c”

En este archivo, hay tres funciones que requieren ser implementadas. La primera de ellas es para escribir, la siguiente para leer y la última para reiniciar.

Las tres funciones, vienen en la librería y es necesaria crear las acciones una a una. Todas tienen unos comentarios en verde que guían al programador para saber paso a paso la secuencia que se debe seguir para la creación de la función. Cabe destacar que es necesario para este archivo haber configurado los puertos y los pines del microcontrolador con el modo "Smart Configurator" visto en el punto *6.1.2.1. Puertos*, y haber definido dichos pines en el subapartado *8.1.3. Archivo de cabecera "Hola_mundo_imagen.h"*.

Para la función de escribir (véase figura 57), es necesario que se le introduzca una variable, cuyos valores puedan ser 1 o 0: si la variable es 1 entiende que el valor que va a leer será un dato y si la variable es 0 entiende que el valor será un comando. También se introduce una variable llamada "dat", que posee el valor que se quiere escribir.

Si recibe un dato (variable "adr" es 1), establece la línea C/D[7] en alto, si recibe un comando (variable "adr" es 0) la establece en bajo. Seguidamente establece /CE[8] en bajo (desactivar), configura el puerto D como salida para permitir la escritura en el puerto. Se utiliza la variable dato introducida para escribir el puerto D con ese valor, se establece /WR[9] en bajo, es decir se activa la opción de escribir porque se activa a nivel bajo. Acto seguido, se establece /WR en alto para desactivarla y, por último, se establece /CE en alto pasivo para desactivarlo.

```
void simwrby(SGUCHAR adr, SGUCHAR dat)
{
   /* A: Set C/D line according to adr, Set /CE line active low */
    // Establecer C/D line según la dirección adr
        if (adr == 0) // Si adr es 0, establecer la línea C/D en bajo
        {
            LCD_A_0 = 0;            // A0 , C/D
        }
        else // Si adr es diferente de 0, establecer la línea C/D en alto
        {
            LCD_A_0 = 1;            // A0 , C/D
        }
        // Establecer /CE line en bajo
        LCD_NCE = 0;                // NCE , /CE , XCS
        R_BSP_SoftwareDelay (1, BSP_DELAY_MICROSECS);

   /* B1: Make data port an output (if required by port architecture) */
    LCD_Port_D_Confi = 0xFF;        // Configurar el puerto D como salida
    R_BSP_SoftwareDelay (1, BSP_DELAY_MICROSECS);

   /* B2: Write data to data port */
    LCD_Port_D = dat;               // Escribir el dato en el puerto D
    R_BSP_SoftwareDelay (1, BSP_DELAY_MICROSECS);

   /* C: Set /WR active low, (Delay min 80 ns), */
    LCD_NWR = 0;                    // Establecer /WR en bajo
    R_BSP_SoftwareDelay (1, BSP_DELAY_MICROSECS);

    // Timer
    R_BSP_SoftwareDelay (1, BSP DELAY MICROSECS);

   /* D: Set /WR passive high */
    LCD_NWR = 1;                    // Establecer /WR en alto
    R_BSP_SoftwareDelay (1, BSP DELAY MICROSECS);

   /* E: Set /CE passive high */
    LCD_NCE = 1;                    // Establecer XCS Ó LCD_ASSR (/CE) en alto pasivo
    R_BSP_SoftwareDelay (1, BSP DELAY MICROSECS);
}
```

Figura 57. *Función de escribir creada en el de origen "bussim.c".*

Para la función leer se siguen pasos similares (véase figura 58). En la función se introduce un único valor que es el dato que se quiere leer, se limpia una variable con el nombre "dat" y se hacen los mismos primeros pasos que escribir. Pero se configura el puerto D como entrada (en

[7] Sigla que en inglés significa *Command/Data, es decir, indica si está hablando de un dato (con un 1) o de un comando (0).*

[8] Sigla que en inglés significa *Chip Enable*, en español Habilitación de Chip. Al llevar la barra delante significa que es negado, es decir, se activa cuando está a nivel bajo (con un 0) y se desactiva cuando está a nivel alto (con un 1).

[9] Siglas que en inglés significa write, en castellano escribir, es el chip que habilita o deshabilita escribir en el puerto D.

lugar de salida) y, para permitir la lectura, se establece /RD[10] en bajo para activar la opción de leer. Se lee el puerto D usando la variable “dat”, se establece /RD en alto para desactivarlo, se establece /CE en alto pasivo (activar, al igual que en escribir) y por último, devuelve el dato leído.

```
SGUCHAR simrdby(SGUCHAR adr)
{
    SGUCHAR dat = 0U;
    /* a: Set C/D line according to adr. Set /CE line active low */
    // Establecer C/D line según la dirección adr
        if (adr == 0) // Si adr es 0, establecer la línea C/D en bajo
        {
            LCD_A_0 = 0;            // A0 , C/D
        }
        else // Si adr es diferente de 0, establecer la línea C/D en alto
        {
            LCD_A_0 = 1;            // A0 , C/D
        }
        // Establecer /CE line en bajo
        LCD_NCE = 0;                // LCD_ASSR , /CE , XCS

    /* b: Make data port an input (if required by port architecture) */
    LCD_Port_D_Confi = 0x11;        // Configurar el puerto D como entrada

    /* c: Set /RD active low, (Delay min 150ns), */
    LCD_NRD = 0;                    // Establecer /RD en bajo

    // Timer
    R_BSP_SoftwareDelay (1, BSP_DELAY_MICROSECS);

    /* d: Read data from data port */
    dat = LCD_Port_D;               // Leer el puerto D

    /* e1:Set /RD passive high, */
    LCD_NRD = 1;                    // Establecer /RD en alto

    /* e2:Set /CE passive high (could be ignored) */
    LCD_NCE = 1;                    // Establecer XCS Ó LCD_ASSR (/CE) en alto pasivo

    return dat;
}
```

Figura 58. *Función de leer creada en el de origen “bussim.c”.*

La función más diferente es la de reinicio, como se puede ver en la figura 59. Primero se escribe en el puerto D, el comando “LCD_EPINT”, que se utiliza para generar una señal de reconocimiento. Seguidamente se configura el /RS a nivel bajo, es decir, se activa el reinicio. Se efectúa un retardo y se desactiva /RS, es decir, se configura como alto.

```
void sim_reset( void )
{
    /* 1. Init data port setup (if required by port architecture) */
     LCD_Port_D = LCD_EPINT;          // Escribir en el puerto D el codigo inicial

    /* 2. Make C/D, /RD, /WR, /CE to outputs (if required by port architecture) */

    /* 3. Set LCD reset line /RST active low   (if /RST is connected to a port bit) */
     LCD_NRST = 0;
     R_BSP_SoftwareDelay (1, BSP_DELAY_MILLISECS);

    /* 4. Set LCD reset line /RST passive high (if /RST is connected to a port bit) */
     LCD_NRST = 1;
}
```

Figura 59. *Función de reinicio creada en el de origen “bussim.c”.*

8.1.6. Archivo principal del proyecto “Hola_mundo_imagen.c”

En este subapartado se analiza detalladamente el archivo "Hola_mundo_imagen.c". Este archivo, considerado como el núcleo del proyecto, coordina y ejecuta los demás archivos mencionados anteriormente. Su objetivo principal es generar y actualizar una interfaz de usuario en la pantalla.

Dada la extensión y complejidad del contenido de este archivo, se ha decidido fragmentar la exposición de su estructura y funciones en tres secciones. En la primera se abordarán los "#includes" del programa y se explicará la importancia de las variables externas.

[10] Siglas que en inglés significa read, en castellano leer, es el chip que habilita o deshabilita leer el puerto D.

La segunda sección se dedicará a detallar las funciones que han sido desarrolladas en el archivo, proporcionando una explicación de cada una y su propósito en el conjunto del programa.

Finalmente, en la tercera sección, se presentará el código completo para brindar una visión general y a cómo todas las partes se interrelacionan para lograr el objetivo final. Se asegurará que a lo largo de estas secciones se aclararán todos los términos específicos y conceptos clave para garantizar una comprensión completa de este archivo esencial del proyecto.

8.1.6.1. Declaraciones iniciales y variables externas.

Para comenzar con la explicación del código principal de proyecto, es indispensable iniciar mencionando las declaraciones que debe disponer para que no den errores de compilación. Las declaraciones de procesador, típicamente se usan para incluir archivos de cabecera, que permiten acceder a otros archivos del programa. Las declaraciones del archivo, se muestran en la figura 60.

```
#include "r_smc_entry.h"
#include "Hola_mundo_imagen.h"
#include <bussim.h>
#include <gdisp.h> /* Prototype LCD driver functions */
#include <stdio.h> /* Sprintf prototype */
#include <gdispcfg.h>
#include <gdisphw.h>
```

Figura 60. *Declaraciones del archivo principal del proyecto "Hola_mundo_imagen.c".*

Por otro lado, en esta sección se han añadido las variables externas (véase figura 61). Estas, son las variables declaradas fuera de cualquier función, es decir, en el ámbito global del programa, pero se utilizan o referencian en más de un archivo de código fuente. Se declaran con la palabra clave "extern". El propósito principal de las variables externas es permitir la comunicación y el intercambio de datos entre diferentes archivos de código fuente dentro de un mismo programa.

En este proyecto, se han utilizado las variables externas para hacer referencia a los símbolos creados en el apartado *7. Desarrollo gráfico de widget con IconEdit*. Esto es debido, a que está guardada la información de los símbolos en forma de matriz en un archivo con terminación ".sym" y se hace referencia a ella desde un archivo ".c". Por lo que primero se tiene que llamar a la variable del archivo ".c" (que será la variable externa), que a su vez estará llamando al archivo ".sym".

En la figura 61, se muestran las variables externas que se utilizan para llamar a los símbolos creados para hacer la interfaz de usuario.

```
extern GCODE GFONT FCODE end_background_2;
extern GCODE GFONT FCODE flecha;
extern GCODE GFONT FCODE llenado2;
extern GCODE GFONT FCODE circulo_A;
```

Figura 61. *Variables externas del archivo principal del proyecto "Hola_mundo_imagen.c".*

8.1.6.2. Funciones

En esta sección se incluyen las funciones que se utilizan en el código principal.[19]

- Función "test_symbol_fondo"

La función es de tipo estática, por lo que no devuelve ningún valor (void). Está relacionada con la visualización de símbolos gráficos en la interfaz de usuario (véase figura 62).

Primero, con "PGSYMBOL", que es un puntero[11] que apunta a un símbolo creado con IconEdit.

La siguiente línea, llama a la función "ggetfsym" que pertenece a la librería de gráficos, devuelve un puntero. El primer argumento pasado a esta función (0U), indica el índice del símbolo que se quiere visualizar (en este caso el primero, en el único caso que hay más de uno es el del proyecto creado en el punto 7.3. Creación del widget tipo arco) y el segundo parámetro es "&end_background_2", que es la dirección de memoria del que contiene los datos del símbolo. El puntero obtenido se almacena en la variable ps.

La última línea llama a la función "gputsym", que muestra el símbolo por pantalla. Los dos primeros argumentos (0U, 0U) son las coordenadas (x,y) en las que se mostrara el símbolo (en este caso, la esquina superior izquierda de la pantalla) y ps es el puntero al símbolo que se va a representar.

```
// end_background.c must be included in the compilation project
static void test_symbol_fondo(void)
    {
    PGSYMBOL ps;
    ps = ggetfsym(0U, &end_background_2);  //Get a pointer to first symbol in font
    gputsym(0U,0U,ps); // Output symbol in upper left corner
    }
```

Figura 62. *Función "test_symbol_fondo" localizada en el archivo principal del proyecto "Hola_mundo_imagen.c".*

- Función "test_symbol_flecha"

La siguiente función es exactamente igual que la anterior, pero, en vez de mostrar el fondo (como la figura 62), esta mostrará una flecha, que indicará la temperatura (véase figura 63).

```
static void test_symbol_flecha(void)
    {
    PGSYMBOL ps;
    ps = ggetfsym(0U, &flecha);
    gputsym(54U,78U,ps);
    }
```

Figura 63. *Función "test_symbol_flecha" localizada en el archivo principal del proyecto "Hola_mundo_imagen.c".*

- Función "escribirtemp"

Se utiliza para convertir un valor de grados (pasado como parámetro grados de ángulo) en un valor de temperatura y para visualizarlo en la interfaz gráfica (véase figura 64).

Primero, realiza el cálculo que transforma el valor en grados (ángulo) a una escala de temperatura (de 0ºC a 200ºC). En este caso, se multiplican los grados (ángulo) por 200.0 / 136.0 para obtener variable "temp". Este cálculo relaciona los grados de inclinación de una barra, con temperatura en grados centígrados.

[11] Un puntero es una variable que almacena la dirección de memoria de otra variable, es decir, no guarda un valor en sí mismo, sino que apunta a la ubicación en la que se encuentra el valor real.

Luego, se establece la letra, con fuente “times13” para visualizar el valor de la temperatura.

En función del número de dígitos que componen el valor, la función establece una posición diferente para que la visualización del número esté centrada en la pantalla.

Finalmente, se convierte la temperatura temp a una cadena de caracteres usando la función sprintf. Esta cadena de caracteres se almacena en el array llamado num y se visualiza en la pantalla en la posición establecida anteriormente.

```
void escribirtemp(SGINT grados)
{
    float temp;
    char num[5];

    temp =  grados * (200.0 / 136.0);

    gselfont(&times13);

    if (temp < 10)
    {
       gsetpos(5+41,117); // 85
    }
    if (temp >= 10 && temp < 100)
    {
       gsetpos(10+33,117); // 77
    }
    if (temp >= 100)
    {
       gsetpos(15+25,117); // 69
    }
    sprintf(num, "%.1f", temp); // CON UN DECIMAL
    gputs(num);
}
```

Figura 64. Función “escribirtemp” localizada en el archivo principal del proyecto “Hola_mundo_imagen.c”.

- Función “test_symbol_circulo”

A diferencia de las dos funciones anteriores (figuras 62, 63 y 64), esta no solo muestra un símbolo, sino que, dependiendo del parámetro que se dé inicialmente, aparecerá en una posición u otra, la unidad de medida respectiva de cada una y el valor numérico (véase figura 65 y 66).

Esta función recibe dos parámetros, “type” y “valor_llenado”. “type” determina qué tipo de símbolo se va a dibujar y cómo se va a manejar el valor que se muestra. En cambio, “valor_llenado” representa un valor que se quiere representar en la interfaz gráfica.

La función trabaja de la siguiente manera: se obtiene un puntero a un símbolo específico usando la función “ggetfsym”. Basándose en el parámetro “type”, dibuja el símbolo y muestra una unidad de medida y un valor numérico en una posición específica. La forma de imprimir el valor por pantalla varía dependiendo de “type”. Si “type” es 0, muestra símbolo en una posición (arriba a la izquierda), un carácter "V" y un número entero (el valor). Si “type” es 1, muestra el símbolo en otra posición (arriba, en medio), la unidad de medidas “Hz” y un número con un único decimal. Por último, si “type” es 2, se mostrará el símbolo en arriba a la izquierda, con la unidad “kW” y un valor entero.

Los valores numéricos son calculados a partir de “valor_llenado” (que será un número entre 0 y 80) y según la cantidad de dígitos que tiene, se pondrá en una posición u otra. Para indicar la posición, se utiliza la función “gsetpos” y para mostrar por pantalla “gputs”.

```
79  static void test_symbol_circulo(SGCHAR type, SGUCHAR valor_llenado)
80  {
81      PGSYMBOL ps;
82      ps = ggetfsym(valor_llenado, &llenado2);
83      SGINT valor = 999 * valor_llenado / 80;
84      float valor2 = 99.9 * valor_llenado / 80;
85      char num[5];
86      if (type == 0)
87      {
88          gputsym(14U,13U,ps);
89          gselfont(&times9);
90          gsetpos(36,43);
91          gputs("V");
92
93          gselfont(&times13);
94          if (valor >= 100 && valor <= 999)
95          {
96              gsetpos(28,62);
97          }
98          if (valor >= 10 && valor <= 99)
99          {
100             gsetpos(32,62);
101         }
102         if (valor >= 0 && valor <= 9)
103         {
104             gsetpos(36,62);
105         }
106         sprintf(num, "%d", valor);
107         gputs(num);
108     }
109
110     if (type == 1)
111     {
112         gputsym(94U,13U,ps);
113         gselfont(&times9);
114         gsetpos(115,43);
115         gputs("Hz");
116
```

Figura 65. *Función “test_symbol_circulo” en el archivo principal del proyecto “Hola_mundo_imagen.c” (primero).*

```
117         gselfont(&times13);
118         if (valor2 >= 10)
119         {
120             gsetpos(106,62);
121         }
122         if (valor2 < 10)
123         {
124             gsetpos(110,62);
125         }
126         sprintf(num, "%.1f", valor2); // CON UN DECIMAL
127         gputs(num);
128     }
129
130     if (type == 2)
131     {
132         gputsym(174U,13U,ps);
133         gselfont(&times9);
134         gsetpos(194,43);
135         gputs("kW");
136
137         gselfont(&times13);
138         if (valor >= 100 && valor <= 999)
139         {
140             gsetpos(188,62);
141         }
142         if (valor >= 10 && valor <= 99)
143         {
144             gsetpos(192,62);
145         }
146         if (valor >= 0 && valor <= 9)
147         {
148             gsetpos(196,62);
149         }
150         sprintf(num, "%d", valor);
151         gputs(num);
152     }
153 }
```

Figura 66. *Función “test_symbol_circulo” en el archivo principal del proyecto “Hola_mundo_imagen.c” (segundo).*

- Función “drawColumn”

Su finalidad es dibujar una barra que representa un porcentaje en la interfaz gráfica (véase figura 67 y 68).

A esta función se le introduce inicialmente una variable llamada “percentage”, que indica el porcentaje de llenado.

Antes de nada, se pintará una barra de gris claro (“G_LLIGHTGREY”) que ocupe el 100% de llenado, para que borre el valor anterior, si existía, y si no servirá como indicador del llenado completo.

Seguidamente, se utiliza un bucle *for* que va desde 0 hasta el valor del porcentaje, dibujando segmentos de columna. Hay dos tonos de grises, gris oscuro (“G_DDARKGREY”) y gris (“G_GREY”), dependiendo de si el valor del segmento es múltiplo de 5. Ofreciendo de manera intuitiva algo con lo que guiarse al ver el gráfico, ya que cada 4 valores el quinto tendrá una columna oscura para visualizar el valor con más precisión.

Por último, imprime el valor del porcentaje en la interfaz gráfica dependiendo de la cantidad de dígitos que tenga. Para que el valor esté siempre en una posición centrada.

```
156  void drawColumn(uint8_t percentage)
157  {
158      gsetcolorf(G_LLIGHTGREY);
159      gfillvp(129U, 88U,229U, 103U, 0xFFFFU);
160      //gfillvp(129U, 88U, 229U, 103U, 0x0000U);
161      uint8_t column;
162      GCOLOR c;
163      char str[3];
164
165      // Dibujar la columna
166      for (column = 0; column <= percentage; column++)
167      {
168          if (column % 5 == 0)
169          {
170              c = G_DDARKGREY;
171
172          } else
173          {
174              c = G_GREY;
175              //c = 14U;
176          }
177
178          gsetcolorf(c);
179          gfillvp(column + 129U, 88U, column + 129U, 103U, 0xFFFFU);
180      }
181
182      if (percentage == 100)
183      {
184          gsetcolorf(G_BLACK);
185          gselfont(&ariel18);
186          gsetpos(192,87);
187          gputs("100");
188      } else
```

Figura 67. Función “drawColumn” en el archivo principal del proyecto “Hola_mundo_imagen.c” (primero).

```
189         {
190             if (percentage <= 9 )
191             {
192                 gsetcolorf(G_BLACK);
193                 gselfont(&ariel18);
194                 gsetpos(210,87);
195                 sprintf(str,"%d",percentage);
196                 gputs(str);
197             }
198             else
199             {
200                 gsetcolorf(G_BLACK);
201                 gselfont(&ariel18);
202                 gsetpos(201,87);
203                 sprintf(str,"%d",percentage);
204                 gputs(str);
205             }
206 
207         }
208     }
```

Figura 68. *Función "drawColumn" en el archivo principal del proyecto "Hola_mundo_imagen.c" (segundo).*

- Función "nuevoAleatorio"

Esta última función del archivo principal se usa para generar un número aleatorio dentro de un rango específico que está cerca del valor anterior (para que no haya tanta diferencia entre un valor y otro, por mucho que sean aleatorios). El rango se define como diez unidades por debajo y diez por encima del valor dado, pero también se asegura que el rango (10 por abajo y por arriba) no se exceda los límites máximos y mínimos establecidos. (véase figura 69).

Primero se define la variable "rangoMin" como 10 unidades menos que el valor dado y "rangoMax" como 10 unidades más.

Se comprueba si "rangoMin" es menor que el mínimo permitido ("min" que viene dada cuando se llama la función). Si es así, lo ajusta al mínimo permitido (es decir el valor de "rangoMin" pasará a ser el mismo de "min", porque no puede ser más pequeño que le menor posible).

Del mismo modo, se comprueba si "rangoMax" es mayor que el máximo permitido ("max"). Si es así, lo ajusta al máximo permitido.

Se genera un número aleatorio entre el rango ["rangoMin", "rangoMax"]. Para hacer esto, se usa la función "rand()" que genera un número aleatorio y luego se aplica el operador de módulo % para limitarlo dentro del rango. Después, se suma "rangoMin" para asegurar que el número aleatorio generado se encuentre dentro del rango ["rangoMin", "rangoMax"].

Finalmente, se retorna el valor aleatorio generado.

```
SGINT nuevoAleatorio (SGINT valor, SGINT min, SGINT max)
{
    SGINT rangoMin, rangoMax;

    rangoMin = valor - 10;
    rangoMax = valor + 10;

    if (rangoMin < min)
        rangoMin = min;

    if (rangoMax > max)
        rangoMax = max;

    valor = rangoMin + rand() % (rangoMax - rangoMin + 1);

    return valor;
}
```

Figura 69. *Función "nuevoAleatorio" en el archivo principal del proyecto "Hola_mundo_imagen.c".*

8.1.6.3. Código principal

En esta última sección, se presentará el código completo del proyecto. El propósito general de este código es mostrar y actualizar varios gráficos en la pantalla de forma repetitiva, con algunos de los valores de los gráficos que cambian aleatoriamente en cada iteración.

La función “main()” es la función principal del programa y realiza varias tareas:

1. La primera es inicializar el sistema gráfico con la función “ginit()”. Esta función se incluye dentro de otros archivos y funciones (como es entre otros, el archivo de origen “ghwinitctrl.c”).
2. La segunda, es declarar varias variables e inicializar algunas, que son “i”, “j”, “valor_flech”, “valor_barr”, “valor_cir_0”, “valor_cir_1” y “valor_cir_2”.
3. Por último, se desactivan las actualizaciones de la pantalla con “gsetupdate(GUPDATE_OFF)” para evitar el parpadeo durante la actualización de los gráficos y para poder guardar el estado anterior de la pantalla únicamente con el fondo.

Una vez finalizadas las inicializaciones, declaraciones y desactivar la actualización de la pantalla, se entra al código importante. Todas las funciones mencionadas a continuación, están definidas en el punto *8.1.5.2. Funciones.*

Primero, se llama la función “test_symbol_fondo()”, se encarga de configurar el fondo de la pantalla.

A continuación, se realiza un bucle desde 0 hasta 100, durante el cual, si es la primera iteración del bucle (es decir, j == 0), se asigna a las variables “valor_flech”, “valor_barr”, “valor_cir_0”, “valor_cir_1” y “valor_cir_2” un número aleatorio dentro de un rango específico (véase líneas 418, 419, 420, 421 y 422, de la figura 70).

Si no es la primera iteración, se llama a la función “nuevoAleatorio()”, para cada una de estas variables recientemente mencionadas. Se genera un número aleatorio dentro de un rango específico que está cerca del valor actual de cada variable (es decir, el valor anterior y el nuevo, no pueden llevarse más del 10%).

Se llama a las funciones: “escribirtemp(valor_flech)”, para mostrar la temperatura por pantalla; “test_symbol_circulo()” tres veces con diferentes parámetros, ya que se va a mostrar en cada uno un valor numérico diferente, en una posición distinta y con la unidad de medida respectiva de cada uno; “drawColumn(valor_barr)”, para dibujar el gráfico en forma de barra horizontal; y “gputsymrot()”, que rota el símbolo de la flecha medidora de temperatura y lo muestra por pantalla (véase figuras 70 y 71).

Seguidamente, se activa y desactiva las actualizaciones de pantalla con “gsetupdate(GUPDATE_ON)” y “gsetupdate(GUPDATE_OFF)”. Se hace una pausa durante un periodo de tiempo específico con la función “R_BSP_SoftwareDelay()”.

Al final del bucle, se activan las actualizaciones de pantalla con “gsetupdate(GUPDATE_ON)”. Finalmente, el programa entra en un bucle infinito con “while(1)”, lo que significa que permanecerá en esta línea de código de forma indefinida.

```
402  void main(void)
403  {
404      ginit();
405
406      int16_t j, valor_flech, valor_barr, valor_cir_0, valor_cir_1, valor_cir_2;
407
408      gsetupdate(GUPDATE_OFF);
409
410      test_symbol_fondo();
411
412      // ALEATORIO A LA VEZ
413      for (j = 0; j <= 100; j++)
414      {
415          //valor = 1 + rand() % 100;
416          if (j == 0)
417          {
418              valor_flech = 0 + rand() % 137;
419              valor_barr = 0 + rand() % 101;
420              valor_cir_0 = 0 + rand() % 81;
421              valor_cir_1 = 0 + rand() % 81;
422              valor_cir_2 = 0 + rand() % 81;
423          }
424          else
425          {
426              valor_flech = nuevoAleatorio(valor_flech, 0, 137);
427              valor_barr = nuevoAleatorio(valor_barr, 0, 100);
428              valor_cir_0 = nuevoAleatorio(valor_cir_0, 0, 80);
429              valor_cir_1 = nuevoAleatorio(valor_cir_1, 0, 80);
430              valor_cir_2 = nuevoAleatorio(valor_cir_2, 0, 80);
431          }
432
433          escribirtemp(valor_flech);
434              gselfont(&times13);
435              gsetpos(75,117);
436              gputs("°C");
437          test_symbol_circulo(0,valor_cir_0);
438          test_symbol_circulo(1,valor_cir_1);
439          test_symbol_circulo(2,valor_cir_2);
```

Figura 70. *Código principal del archivo principal del proyecto "Hola_mundo_imagen.c" (primero).*

```
440          drawColumn(valor_barr);
441          gputsymrot(58, 100, G_DEGREE_TO_RADIAN(68-valor_flech), ggetfsym(0U, &flecha_corta), 0,
442          gsetupdate(GUPDATE_ON);
443          gsetupdate(GUPDATE_OFF);
444          GTRANSPARENT;
445          drawColumn(valor_barr);
446          R_BSP_SoftwareDelay (1000000, BSP_DELAY_MICROSECS);
447      }
448      gsetupdate(GUPDATE_ON);
449
450      while(1);
451  }
```

Figura 71. *Código principal del archivo principal del proyecto "Hola_mundo_imagen.c" (segundo).*

9. Pruebas

En el siguiente apartado, se abordará una revisión exhaustiva de las pruebas llevadas a cabo durante el período de prácticas de la estudiante en Dismuntel. Este análisis seguirá un orden cronológico, permitiendo observar de manera detallada el progreso alcanzado, abarcando desde los primeros intentos y los problemas encontrados hasta los logros y avances logrados. Esta revisión constituirá un recorrido claro por el trabajo realizado, presentando no solo los aciertos sino también las dificultades y desafíos encontrados. Este enfoque refleja la intención de mostrar un panorama completo y realista del proceso de aprendizaje que tuvo lugar durante las prácticas.

9.1. Diseños de conexiones

Al comienzo de las prácticas, se utilizaron los diseños de las conexiones entre el controlador y la placa, creadas por un trabajador de la empresa. De este modo, aprender a configurar los pines de los puertos con el entorno de desarrollo e2Studio utilizando el modo de trabajo “Smart Configurator”, visto en el apartado *6.1.2.1. Puertos*.

Sin embargo, con estas conexiones se trató de modificar los elementos mostrados por pantalla sin emplear librerías, pero el LCD no cambiaba su estado (se mantenía en blanco). Por ello, se utilizó un multímetro para comprobar si las conexiones del diseño dispuesto por la empresa eran correctas. Pero resultaron los errores de unión mostrados en la figura 70.

NÚMERO DE PIN	ACTUAL	CORRECTO O INCORRECTO	AL QUE DEBERÍA IR
PIN 41	25	INCORRECTO	-
PIN 42	26	CORRECTO	
PIN 43	-	INCORRECTO	25
PIN 44	35	CORRECTO	
PIN 46	19	CORRECTO	
PIN 60	17,23,24	CORRECTO	
PIN 62	18,22	CORRECTO	
PIN 70	36	CORRECTO	
PIN 71	33	INCORRECTO	-
PIN 72	34	CORRECTO	
PIN 73	31	INCORRECTO	33
PIN 74	32	CORRECTO	
PIN 75	29	INCORRECTO	31
PIN 76	30	CORRECTO	
PIN 77	27	INCORRECTO	29
PIN 78	28	CORRECTO	
PIN 79	-	INCORRECTO	27

Figura 72. *Errores en las conexiones del primer diseño.*

Como se puede observar en la figura 72, los valores impares de los pines de la placa más elevados (del 71 al 79) son incorrectos. Esto es debido a que una línea entera de enlaces estaba soldada desplazada un lugar al correspondiente (donde debería estar conectado el pin del controlador 33 al 73 de la placa, está erróneamente conectado el pin 31 del controlador), esto se puede ver de una forma más intuitiva en la figura 2. Por otro lado, también había otro fallo en los pines de valores más bajos, también por desplazamiento del lugar.

A causa de esto, se tuvieron que volver a realizar las conexiones entre la placa base y el controlador st7529. Pero, este no fue el único error encontrado, se reestableció el diseño proporcionado por la empresa debido a que algunas conexiones eran erróneas.

Una vez solucionados los problemas, con el nuevo diseño y uniones, sí se pudieron ejecutar cambios en pantalla, pero sutiles, tanto que casi eran inapreciables. Por lo que, fue necesario comprobar y reajustar los parámetros de inicialización (véase en el *punto 8.1.4. Archivo de origen "ghwinitctrl.c"*). A continuación, se muestran dos figuras de la evolución de reajustar los parámetros iniciales sin poseer los óptimos.

En la figura 73, se puede observar que, al no usar librería y realizar las configuraciones a mano, la mitad de la pantalla no permitía mostrar nada y había poco contraste.

Figura 73. *Primera visualización lograda por pantalla.*

Para solucionar dicho fallo, era necesario cambiar el parámetro del modo de visualización por pantalla y ajustar las dimensiones de la pantalla. Para el contraste se encontró una solución que no era la óptima, puesto que, causaba parpadeo en la pantalla. Esta solución consistía en la configuración de "DISCTRL" sustituir el primer parámetro (PB1 que especifica la proporción de división de CL) por el valor 4 en vez del 0, es decir, cambiar la proporción de división de CL a dos divisiones.

Como se puede observar en la figura 74, sí se llegaba a solucionar aparentemente el problema del contraste (parpadeando). Aun así, se tuvo que reajustar también un desplazamiento tanto en x como en y, por el problema de visualización por pantalla (al ser la pantalla más pequeña que la más común, problema visto en el apartado *8.1.2. Archivo de cabecera "gdispcfg.h"*).

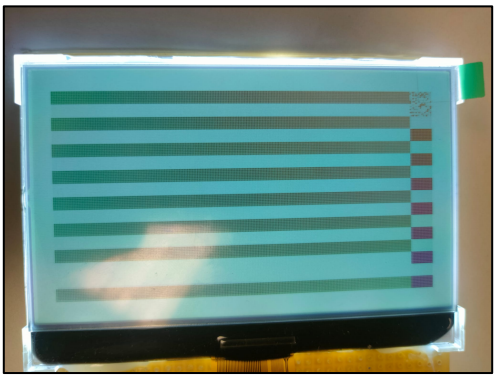


Figura 74. *Solución temporal para el poco contraste.*

Una vez se pudo imprimir y visualizar por pantalla lo mejor posible patrones sencillos (en ese momento), se empleó una imagen que tenía la empresa de 1 byte por píxel. Para mostrar la imagen por pantalla, facilitaron a la alumna el código mostrado en la figura 77.

9.2. Funcionamiento de byte por pixel

Antes de explicar el código de la figura 77, es necesario explicar cómo funciona el mostrar imágenes de 1 byte por píxel sin utilizar librerías.

Como se puede ver en el punto *8.1.4. Archivo de origen "ghwinitctrl.c"*, en el tercer parámetro del comando "DATSDR", se establece que se trabaja en el modo 2B3P (2 bytes, 3 píxeles). Esto quiere decir que, en el primer byte los 5 primeros valores pertenecen al primer píxel, del primer byte los tres últimos valores pertenecen al segundo píxel, del segundo byte los dos primeros valores pertenecen al segundo píxel y del segundo byte los últimos 5 valores pertenecen al tercer píxel (véase figura 75).

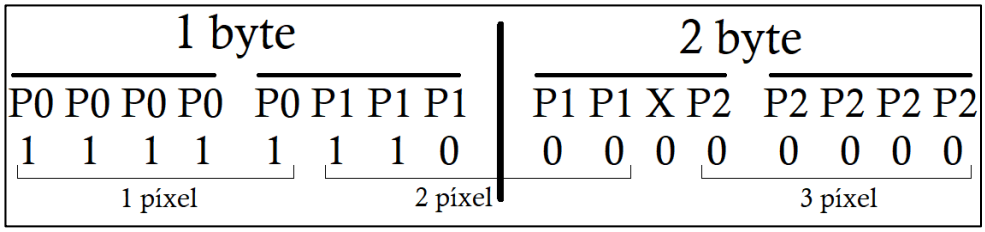


Figura 75. Funcionamiento del modo 2B3P.

Para simplificar la comprensión del cálculo en el modo de trabajo 2B3P de una imagen de 1 byte por píxel, se va a efectuar el siguiente ejemplo. Con los tres valores de una imagen de 1 byte por píxel, que son 0x0F, 0x0C y 0x00 (ejemplo de la figura 75), se va a mostrar cómo se codifica para mostrar 3 píxeles en la pantalla escribiendo por comando 2 bytes.

Una vez creada una imagen con IconEdit, se pasa a código en c, se puede observar que el código creado, es una matriz que cada byte corresponde a un píxel. El problema, es que el valor máximo de dicho byte es 0x0F que es 16 en decimal, esto quiere decir, que trabaja con 16 tonos de gris. En cambio, la pantalla utilizada en este proyecto trabaja con 32 tonos de gris. Por lo tanto, se debe multiplica cada valor de la matriz (en este caso los valores de la imagen) por dos, para que el valor esté sobre 32 tonos.

Una vez multiplicado, se pasa a binario el numero en hexadecimal para que sea más sencillo de entender. Como se muestra en la anterior figura (figura 75), el primer píxel se compone de los cinco primeros valores del byte. Tras realizar la multiplicación los bits con valor son los últimos cinco, para que estén al principio hay que desplazar los bits tres posiciones a la izquierda (véase figura 76).

Para el segundo píxel, son los tres últimos del primer byte y los dos primeros del segundo byte. Por ello, una vez multiplicados, al igual que antes, son los últimos cinco valores.

Para que estén en las tres últimas posiciones hay que desplazar los bits dos posiciones a la derecha. Y para que en el siguiente byte aparezcan los últimos dos valores al principio, hay que desplazar el byte de la imagen multiplicado seis posiciones a la izquierda (véase figura 76).

Por último, para el tercer píxel, una vez se multiplica el byte de la imagen por dos para que esté sobre 32 tonos de grises, no hace falta desplazar, ya que los cuatro primeros valores siempre serán 0 y los importantes son los últimos cinco que están en la posición correcta.

Todos estos cálculos se muestran en la figura 76.

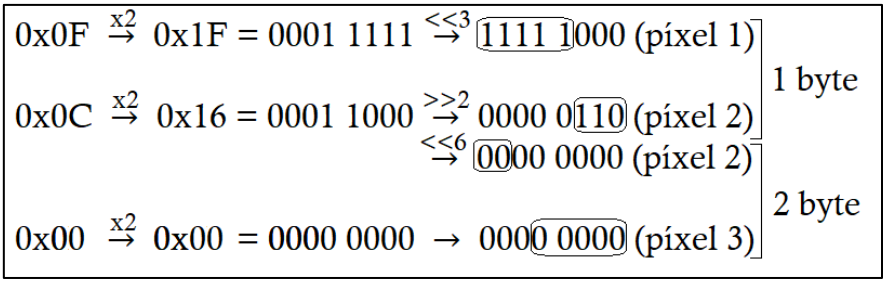


Figura 76. *Cálculos para mostrar una imagen en formato un byte por pixel en modo 2B3P.*

9.3. Código para imprimir por pantalla

Una vez explicado cómo se realizan los cálculos para mostrar una imagen de 1 byte por píxel en modo 2B3P. Se va a explicar el código de la figura 77, proporcionado por la empresa a la alumna.

La estructura general de este bloque de código es un bucle que recorre la imagen píxel por píxel. El bucle “for” exterior itera sobre las líneas de una imagen, asumiendo que la imagen tiene 128 líneas. “línea” es el nombre de la variable que se utiliza para recorrer estas líneas. Por otro lado, el bucle “for” interior itera sobre las columnas de la imagen, asumiendo que la imagen tiene 240 columnas divididas entre 3 (se divide 240 entre 3, ya que en cada iteración se van a completar 3 píxeles). “columna” es el índice que se utiliza para recorrer estas columnas.

Se toma el primer valor del píxel de la imagen en la posición i (inicializado en 0 y se incrementa después de cada lectura de la imagen “IMAGEN2”), se multiplica el valor del byte (correspondiente al píxel) por 2 y se asigna a la variable “gris2”. Este valor se desplaza a la izquierda tres posiciones de bits y se asigna a la variable “grado2”.

Se toma el segundo valor de píxel de la imagen, se ajusta de la misma manera y se asigna de nuevo a “gris2”. Este nuevo “gris2” se desplaza a la derecha dos posiciones de bits y luego se combina con grado2 usando la operación OR bitwise[12] (|=). Esto combina la información de ambos valores, píxel anterior y los últimos tres valores del píxel actual, en un byte. Utilizando la función “ControlLCD(gris2,1)” se muestra el primer byte completado indicado con en el primer píxel de la pantalla y parte de información en el segundo.

Por ello se continúa el procesamiento del segundo byte. Se ajusta “grado2” de nuevo con otro bit-shift[13] a la izquierda de 6 (6 desplazamientos a la izquierda) y una operación OR bitwise. Se toma el tercer valor de píxel de la imagen, se ajusta multiplicando por dos y se asigna a “gris2”. Este “gris2” se combina con el “grado2” con una operación OR bitwise y luego se envía al LCD otro byte con información del final del segundo píxel y el tercer píxel entero utilizando nuevamente la función “ControlLCD(gris2,1)”.

[12] Bitwise, es un término en inglés que se traduce como "a nivel de bit" en español. Se refiere a operaciones que se realizan directamente sobre los bits de los datos en una computadora. Estas operaciones incluyen AND, OR, XOR (exclusivo OR) y NOT.

[13] bit-shift, es una operación en la que los bits en un valor de datos binarios son desplazados hacia la izquierda o hacia la derecha. Es similar a multiplicar o dividir un número por dos.

```
for (linea = 0, i = 0; linea < 128; linea++)
{
    for ( columna = 0; columna < (240/3); columna++)
    {
        // BYTE 1                          // PRIMER BYTE
        gris2 = (IMAGEN2[i++]*2);          // BYTE PARA PIXEL 1
        grado2 = gris2 << 3;

        gris2 = (IMAGEN2[i++]*2);          // BYTE PARA PIXEL 2
        grado2 |= (gris2 >> 2);

        ControlLCD(gris2,1);

        // BYTE 2                          // SEGUNDO BYTE
        grado2 = gris2 << 6;

        gris2 = (IMAGEN2[i++]*2);          // BYTE PARA PIXEL 3
        grado2 |= gris2;
        ControlLCD(gris2,1);
    }
}
```

Figura 77. El código para imprimir por pantalla una imagen en formato un byte por pixel en modo 2B3P.

9.4. Cración de imagen con IconEdit (1Byte x 2 píxels)

Se crea una imagen con el entorno de desarrollo IconEdit (véase figura 78), y se quiere mostrar dicha imagen por LCD.

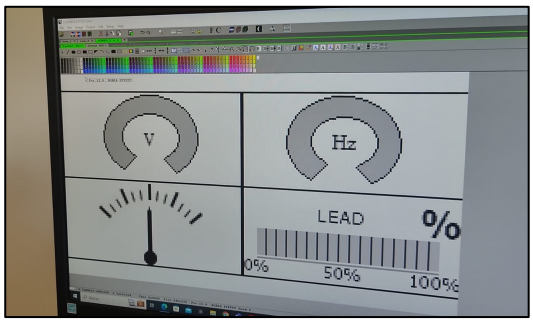


Figura 78. Creación de una imagen con IconEdit (1Byte x 2 píxels) para su posterior visualización.

El programa IconEdit permite descargar la imagen como código en c, pero en formato 1 byte por 2 píxeles (véase figura 177 del Anexo) y se guarda la imagen en forma de matriz. Una vez creada la imagen y guardada como código, se copia la matriz de píxeles (la imagen creada con IconEdit) al inicio del código principal de e2Studio (ya que todavía no se estaba empleando librerías).

Para visualizar por pantalla una imagen guardada en formato 1 byte por dos píxeles, se facilita una imagen, la figura 79, que muestra cómo se escribe la información de seis píxeles en 4 bytes en el modo 2B3P.

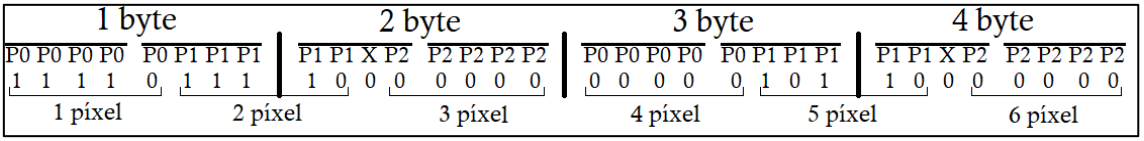


Figura 79. Funcionamiento del modo 2B3P para una imagen guardada en formato 1Byte x 2 píxels.

A continuación, se muestra el cálculo de los 3 primeros bytes de una imagen, para mostrar seis píxeles y la forma en la que se escriben en 4 bytes (véase figura 80).

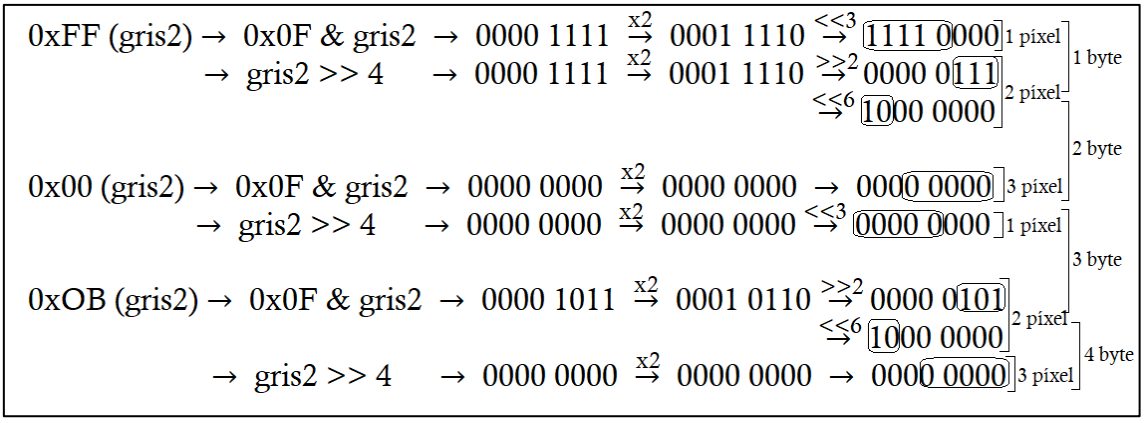


Figura 80. *Cálculos para mostrar una imagen en formato un byte por dos píxeles en modo 2B3P.*

En las figuras 81 y 82, se muestra el código resultante para realizar la visualización de la imagen en formato 1 byte por dos píxeles.

```
        for (linea = 0, i = 0, mascara = 0x01; linea < 128; linea++)
        {
            for (columna = 0; columna < (240 / 6); columna++)
            {
                // BYTE 1                          // PRIMER BYTE
                gris = (IMAGE[i++]);                   // BYTE PARA PIXEL 1 Y 2

                pixel1 = (gris >> 4) * 2;    // Primer pixel
                grado = pixel1 << 3;

                pixel2 = (gris & 0x0f) * 2;      // Segundo pixel 1/2
                grado |= (pixel2 >> 2) & 0x07;

                simwrby(1, grado);           // PRIMER BYTE FINALIZADO


                // BYTE 2                          // SEGUNDO BYTE
                grado = pixel2 << 6;            // Segundo pixel 2/2

                gris = (IMAGE[i++]);                   // BYTE PARA PIXEL 3 Y 1

                pixel3 = (gris >> 4) * 2;    // Tercer pixel
                grado |= pixel3 & 0x1F;

                simwrby(1, grado);           // SEGUNDO BYTE FINALIZADO


                // BYTE 3                          // TERCER BYTE
                pixel1 = (gris & 0x0f) * 2;      // Primer pixel
                grado = pixel1 << 3;

                gris = (IMAGE[i++]);                   // BYTE PARA PIXEL 2 Y 3

                pixel2 = (gris >> 4) * 2;    // Segundo pixel 1/2
                grado |= (pixel2 >> 2) & 0x07;

                simwrby(1, grado);           // TERCER BYTE FINALIZADO
```

Figura 81. *El código para imprimir por pantalla una imagen en formato un byte por dos píxeles (primero).*

```
                // BYTE 4
                grado = pixel2 << 6;              // Segundo pixel 2/2

                pixel3 = (gris & 0x0f) * 2;       // Tercer pixel
                grado |= pixel3 & 0x1F;

                simwrby(1, grado);              // CUARTO BYTE FINALIZADO
            }
        }
```

Figura 82. El código para imprimir por pantalla una imagen en formato un byte por dos píxeles (segundo).

Seguidamente se muestran las pruebas generadas por la alumna hasta dar con el código correcto (el mostrado en la figura 81 y 82). El principal error de la alumna se debió a no escribir correctamente la estructura de los primeros 6 píxeles. También erró a causa de una mala interpretación del tutor de prácticas, puesto qué comentó que los bits más importantes eran los finales. Por ello, si por ejemplo se tiene el byte 0x2B, el profesor comentó de hacer primero el 0x0B (la parte de detrás) y a continuación el 0x02 (la parte de delante). La forma correcta es lo inverso.

9.4.1. Primera prueba

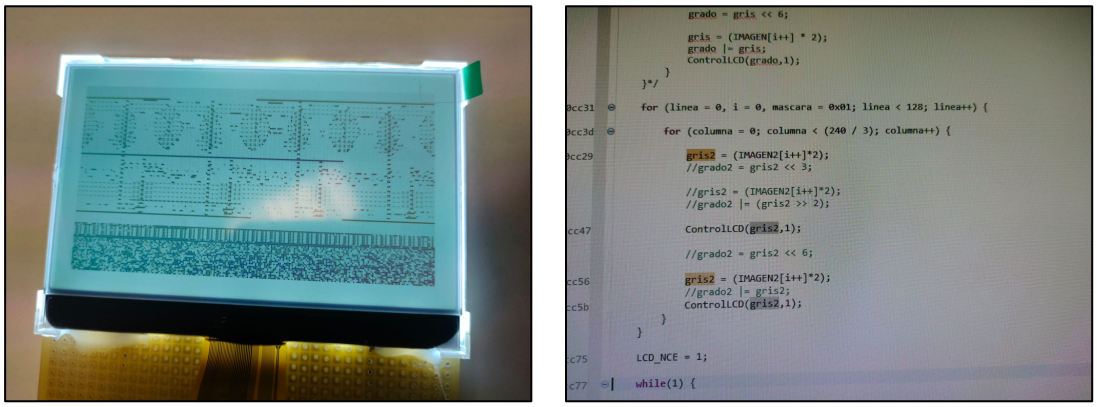


Figura 83. Primera prueba para imprimir una imagen por pantalla.

9.4.2. Segunda prueba

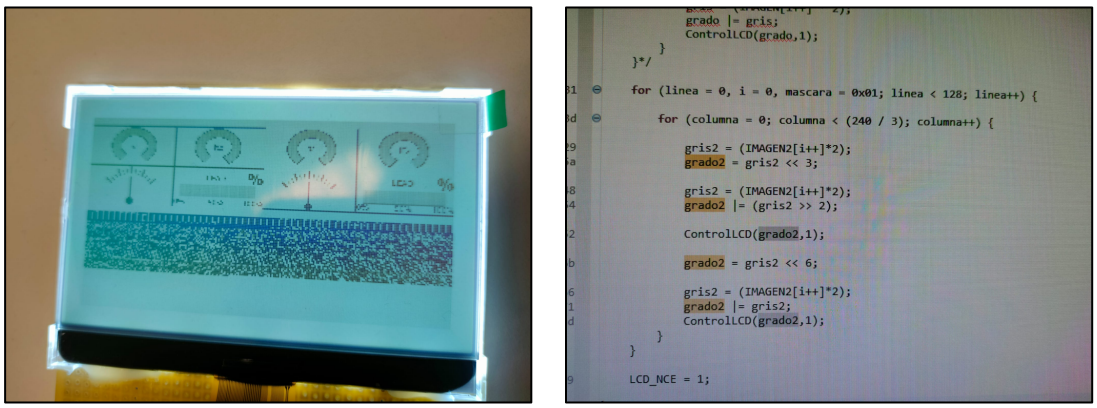


Figura 84. Segunda prueba para imprimir una imagen por pantalla.

9.4.3. Tercera prueba

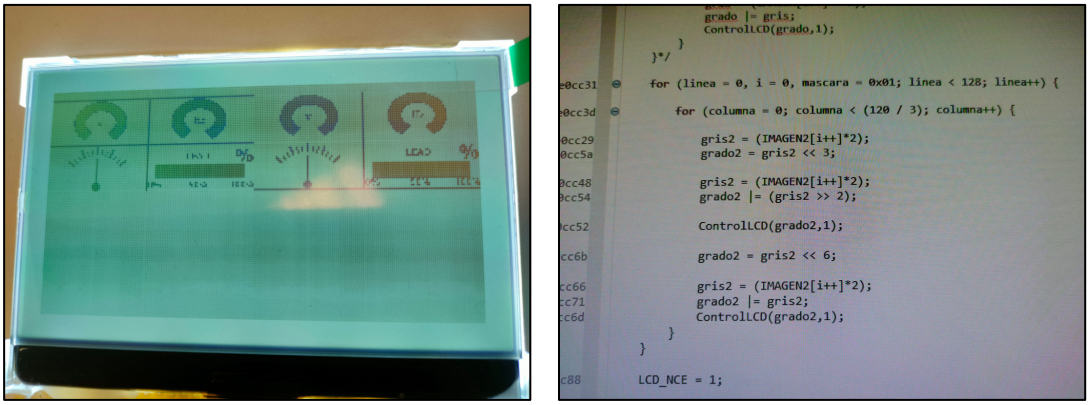


Figura 85. Tercera prueba para imprimir una imagen por pantalla.

9.4.4. Cuarta prueba

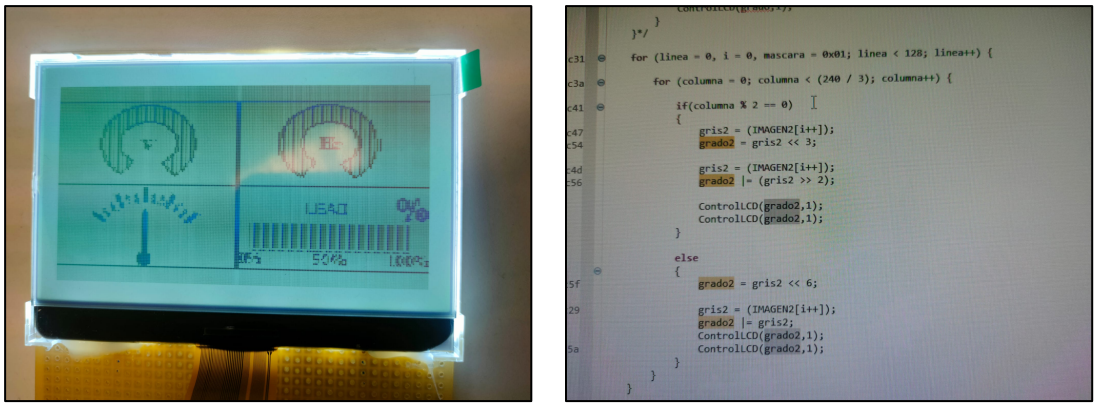


Figura 86. Cuarta prueba para imprimir una imagen por pantalla.

9.4.5. Quinta prueba

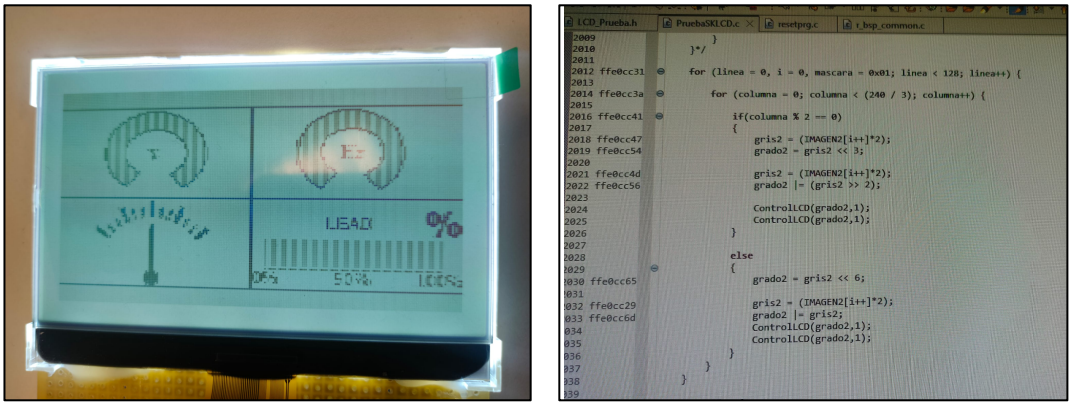


Figura 87. Quinta prueba para imprimir una imagen por pantalla.

9.4.6. Sexta prueba

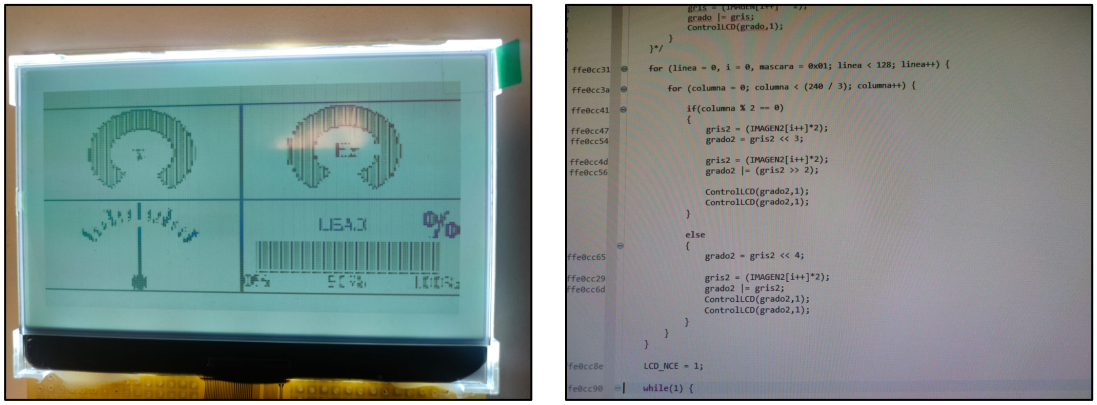


Figura 88. Sexta prueba para imprimir una imagen por pantalla.

9.4.7. *Séptima prueba*

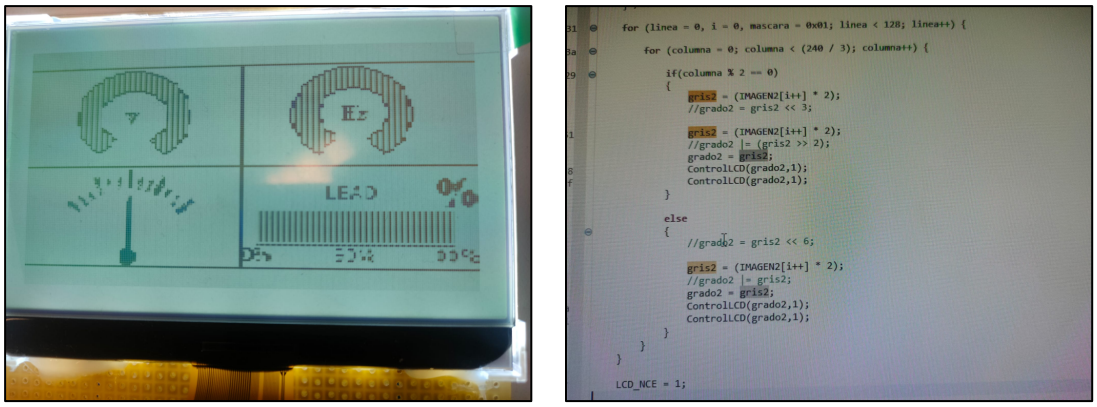


Figura 89. Séptima prueba para imprimir una imagen por pantalla.

9.4.8. *Prueba final*

Para este último, se ve claramente que a causa de seguir lo que mencionó el tutor, los píxeles están intercambiados de sitio.

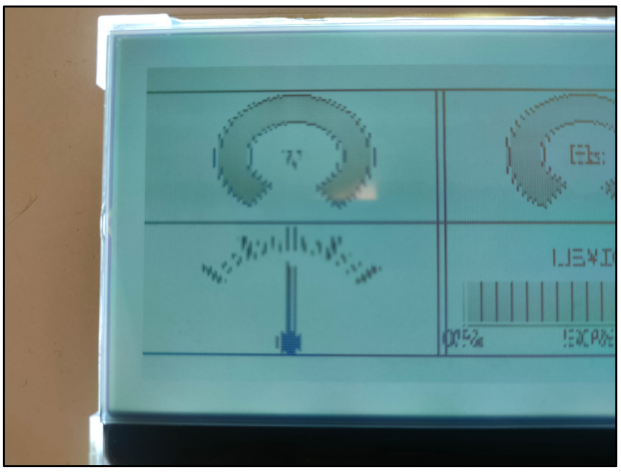


Figura 90. Octava prueba para imprimir una imagen por pantalla.

Una vez finalizado el proceso de mostrar por pantalla una imagen creada con IconEdit (véase figura 90), realizando las configuraciones en e2Stuidio a mano, se decide incorporar las librerías. Por ello, se crea un nuevo proyecto y se añaden las librerías.

Debido a los numerosos fallos obtenidos a causa de las configuraciones, como puede ser el parpadeo de la pantalla o tras incorporar grises, la pantalla se aclara y no se distinguen los grises, ni siquiera se distingue el negro (prácticamente parece gris claro), la empresa le facilitó a la estudiante el contacto del fabricante.

9.5. Correos con el fabricante

En este apartado, se va a mencionar las dudas presentadas al fabricante a través de correos electrónicos y las soluciones aportadas.

9.5.1. Primer correo

Se le introdujo el proyecto al fabricante, los componentes hardware y las características esenciales. Se le adjuntaron imágenes de las inicializaciones de "ghwinitctrl.c" que vienen por defecto con la librería y los cambios realizados. Señalando las inicializaciones por defecto mostradas en la figura 91:

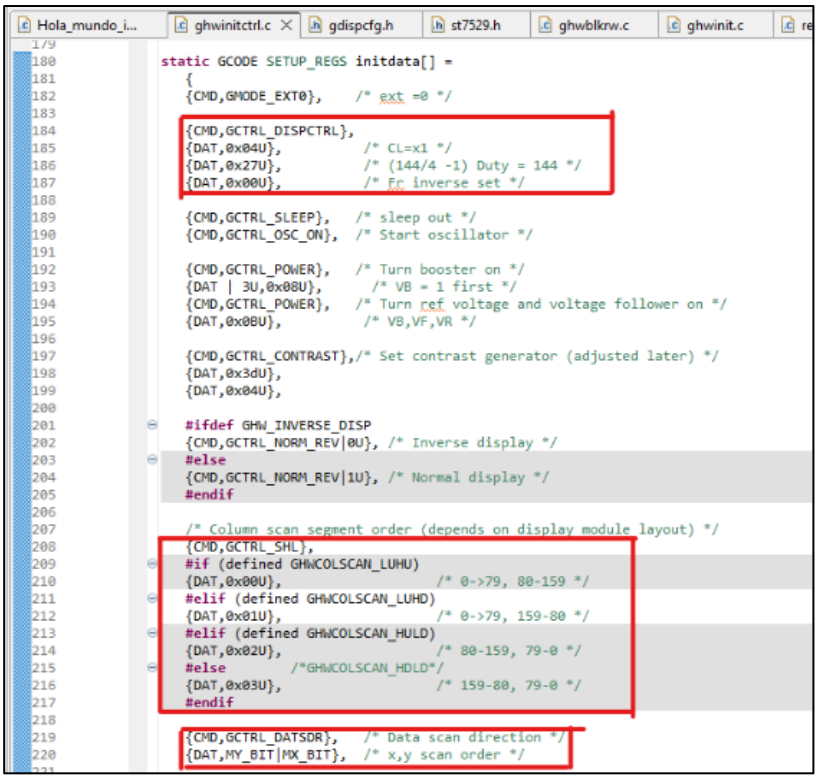


Figura 91. Inicializaciones de "ghwinitctrl.c" por defecto.

Y a continuación los por defecto en el archivo "gdispcfg.h", mostrado en la figura 92:

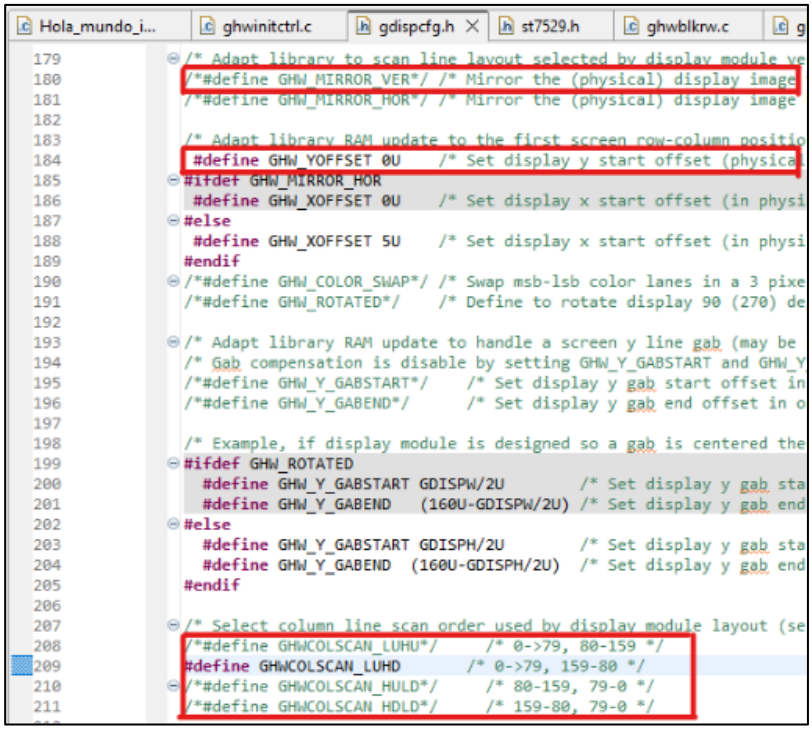


Figura 92. Archivo "gdispcfg.h" por defecto.

Se obtenía por pantalla lo mostrado en la figura 93:

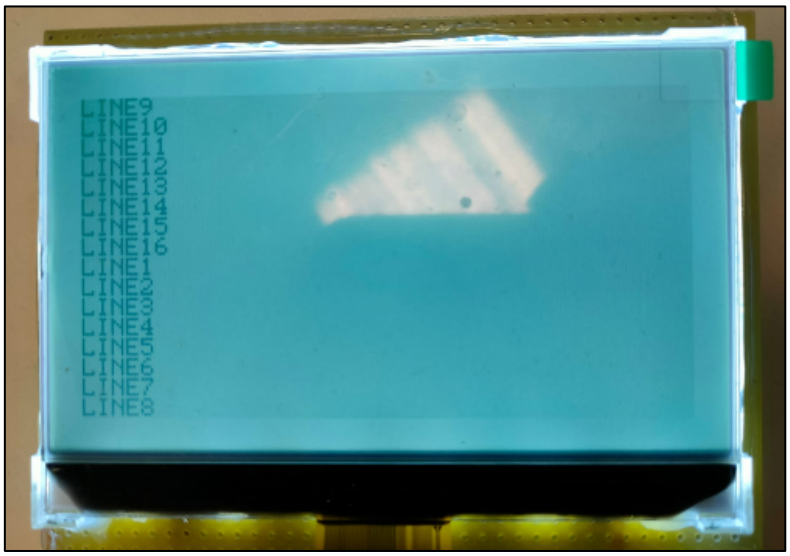


Figura 93. Visualización en la pantalla con configuraciones por defecto.

En el primer dato del comando "GCTRL_DISPCTRL" aparece el valor "0x04" la imagen parpadea, pero si se establece el valor "0x00" la imagen se vuelve demasiado clara y cuando hay tonos de grises casi no se distingue (véase figura 94).

Figura 94. *Visualización por pantalla error.*

Modificando el segundo dato del comando "GCTRL_DISPCTRL" de "0x27" a "0x23", ya que las especificaciones del LCD indican que este dispositivo trabaja con un duty de 1/144 (0x23), no de 1/160 (0x27), desaparecen dos filas de "LINE", pero la imagen parpadea menos (véase figura 95).

Figura 95. *Visualización por pantalla error.*

Ahora se quiere mostrar por pantalla que las líneas estén ordenadas, comenzando por la "LINEA 1". Para que esto sea posible, se modifica el valor "GCTRL_SHL" de "0x01" a "0x02". Para cambiar este valor en "gdispcfg.h" debe deshabilitarse "#define GHWCOLSCAN_LUHD" y habilitar "#define GHWCOLSCAN_HULD". Con este comando las líneas 1 y 2 no aparecen, ya que empieza antes de la última línea (es decir, están por debajo y por eso comienza con la línea 3), se escribe en espejo y aparece el primer error encontrado, el salto de línea entre "LINE 8" y "LINE 9" (véase figura 96).

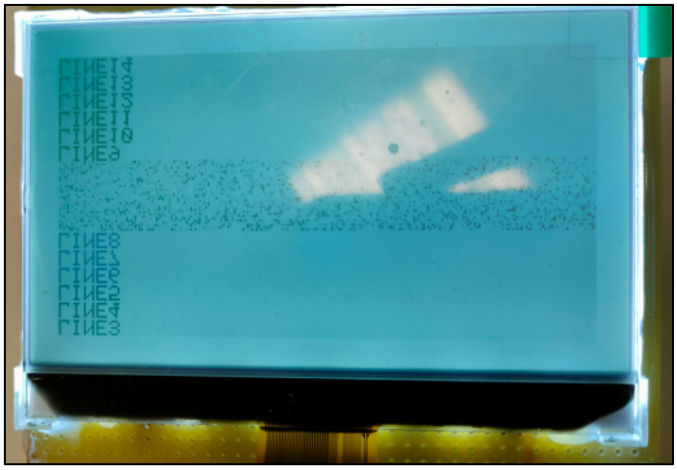


Figura 96. Visualización por pantalla error.

En "gdispcfg.h" se cambia "GHW_YOFFSET" a 16U para que empiece abajo con "LINE 1", (véase figura 97).

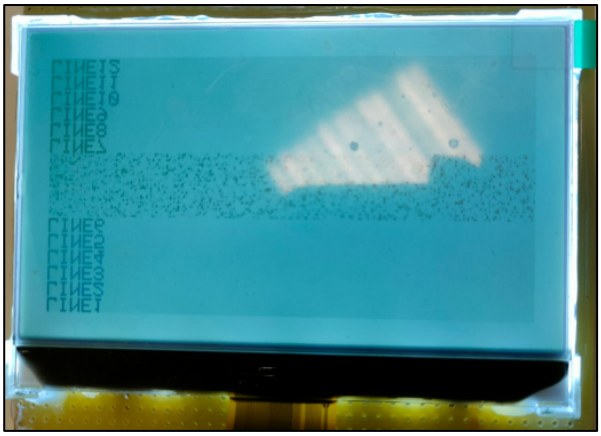


Figura 97. Visualización por pantalla error.

Por último, para que se lea correctamente se activa "#define GHW_MIRROR_VER" (véase figura 98).

Figura 98. Visualización por pantalla error.

Por un lado, el cambio de #define GHWCOLSCAN_LUHD a #define GHWCOLSCAN_HULD, provoca la aparición del error de unas líneas de pantalla con interferencias.

Por otro lado, el error de la visualización de los tonos de grises es causado por la mala configuración, ya que no se diferencian los tonos de grises sin que deje de parpadear la pantalla. La única solución fue sustituir en el primer dato de "GCTRL_DISPCTRL" por "0x04", pero la imagen parpadea.

9.5.2. Segundo correo

Al modificar partes del código de la siguiente manera “#define GHW_Y_GABSTART GDISPH/2U” y “#define GHW_Y_GABEND (160U/2U)”, se obtiene la figura 99.

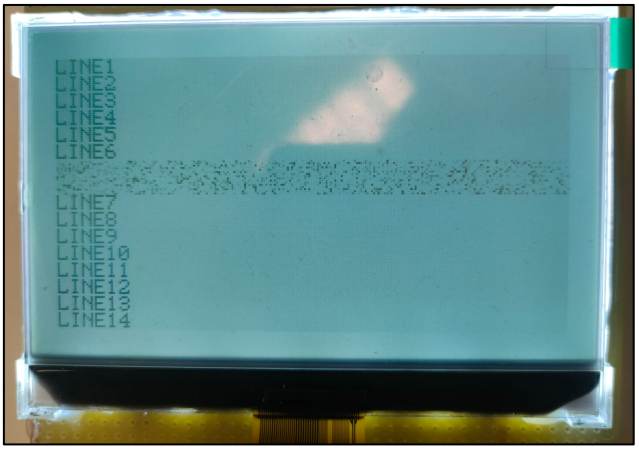


Figura 99. *Visualización por pantalla error.*

Al cambiar “#define GHW_YOFFSET (160U-GDISPH)”, se obtiene la figura 100:

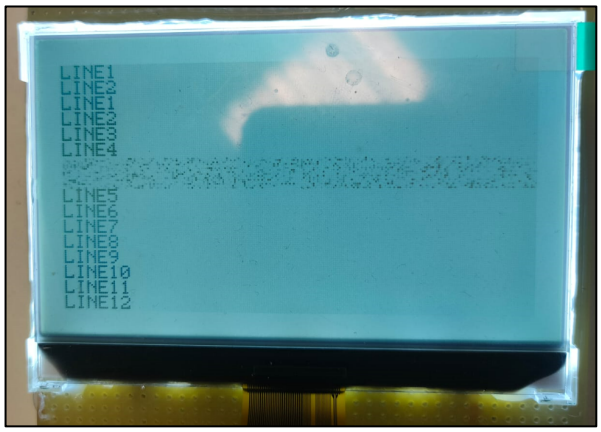


Figura 100. *Visualización por pantalla error.*

En donde "LINE 1" y "LINE 2" se repiten.

9.5.3. Tercer correo

Se procede a modificar el archivo "gdispcfg.h" de la siguiente manera (véase figura 101):

Hola_mundo_imagen.c | ghwinitctrl.c | gdispcfg.h | resetprg.c

```
177  /****************** LOW-LEVEL DRIVER CONFIGURATIONS ******************/
178
179  /* Adapt library to scan line layout selected by display module vendor */
180  #define GHW_MIRROR_VER /* Mirror the (physical) display image vertically */
181  /*#define GHW_MIRROR_HOR*/ /* Mirror the (physical) display image horizontally */
182
183  /* Adapt library RAM update to the first screen row-column positions */
184   #define GHW_YOFFSET 16U    /* Set display y start offset (physical Y coordinate) */
185  #ifdef GHW_MIRROR_HOR
186   #define GHW_XOFFSET 0U    /* Set display x start offset (in physical X storage units (each have 3 pixels))
187  #else
188   #define GHW_XOFFSET 5U    /* Set display x start offset (in physical X storage units (each have 3 pixels))
189  #endif
190  /*#define GHW_COLOR_SWAP*/ /* Swap msb-lsb color lanes in a 3 pixel storage unit (normally not needed) */
191  /*#define GHW_ROTATED*/    /* Define to rotate display 90 (270) degrees (remember to swap values used in GD
192
193  /* Adapt library RAM update to handle a screen y line gab (may be needed when screen height is smaller thar
194  /* Gab compensation is disable by setting GHW_Y_GABSTART and GHW_Y_GABSTART to the same value */
195  /*#define GHW_Y_GABSTART*/    /* Set display y gab start offset in on-chip video ram (physical Y coordinate
196  /*#define GHW_Y_GABEND*/      /* Set display y gab end offset in on-chip video ram (physical Y coordinate)
197
198  /* Example, if display module is designed so a gab is centered then these setting can be used */
199  #ifdef GHW_ROTATED
200    #define GHW_Y_GABSTART GDISPW/2U        /* Set display y gab start offset at half screen (physical Y coor
201    #define GHW_Y_GABEND   (160U-GDISPW/2U) /* Set display y gab end at other half of video RAM (physical Y c
202  #else
203    #define GHW_Y_GABSTART GDISPH/2U        /* Set display y gab start offset at half screen (physical Y coor
204    #define GHW_Y_GABEND  (160U/2U)  /* Set display y gab end at other half of video RAM (physical Y coordina
205  #endif
206
207  /* Select column line scan order used by display module layout (select only one) */
208  /*#define GHWCOLSCAN_LUHU*/     /* 0->79, 80-159 */
209  /*#define GHWCOLSCAN_LUHD*/       /* 0->79, 159-80 */
210  #define GHWCOLSCAN_HULD     /* 80-159, 79-0 */
211  /*#define GHWCOLSCAN_HDLD*/     /* 159-80, 79-0 */
```

Figura 101. Archivo "gdispcfg.h".

El resultado obtenido es un desplazamiento de la pantalla hacia la izquierda (véase figura 102).

Figura 102. Visualización por pantalla error.

Modificando "gdispcfg.h" de la manera mostrada en la figura 103, se obtiene el resultado visualizado en la figura 104:

```
Hola_mundo_imagen.c | ghwinitctrl.c | gdispcfg.h | resetprg.c
177  /****************** LOW-LEVEL DRIVER CONFIGURATIONS ******************/
178
179  /* Adapt library to scan line layout selected by display module vendor */
180  /*#define GHW_MIRROR_VER*/ /* Mirror the (physical) display image vertically */
181  /*#define GHW_MIRROR_HOR*/ /* Mirror the (physical) display image horizontally */
182
183  /* Adapt library RAM update to the first screen row-column positions */
184   #define GHW_YOFFSET 16U    /* Set display y start offset (physical Y coordinate) */
185  #ifdef GHW_MIRROR_HOR
186   #define GHW_XOFFSET 0U    /* Set display x start offset (in physical X storage units (each have 3 pix
187  #else
188   #define GHW_XOFFSET 5U    /* Set display x start offset (in physical X storage units (each have 3 pix
189  #endif
190  /*#define GHW_COLOR_SWAP*/ /* Swap msb-lsb color lanes in a 3 pixel storage unit (normally not needed)
191  /*#define GHW_ROTATED*/    /* Define to rotate display 90 (270) degrees (remember to swap values used
192
193  /* Adapt library RAM update to handle a screen y line gab (may be needed when screen height is smaller
194  /* Gab compensation is disable by setting GHW_Y_GABSTART and GHW_Y_GABSTART to the same value */
195  /*#define GHW_Y_GABSTART*/    /* Set display y gab start offset in on-chip video ram (physical Y coord
196  /*#define GHW_Y_GABEND*/      /* Set display y gab end offset in on-chip video ram (physical Y coordin
197
198  /* Example, if display module is designed so a gab is centered then these setting can be used */
199  #ifdef GHW_ROTATED
200    #define GHW_Y_GABSTART GDISPW/2U        /* Set display y gab start offset at half screen (physical Y
201    #define GHW_Y_GABEND   (160U-GDISPW/2U) /* Set display y gab end at other half of video RAM (physica
202  #else
203    #define GHW_Y_GABSTART GDISPH/2U        /* Set display y gab start offset at half screen (physical Y
204    #define GHW_Y_GABEND  (160U/2U)  /* Set display y gab end at other half of video RAM (physical Y coo
205  #endif
206
207  /* Select column line scan order used by display module layout (select only one) */
208  /*#define GHWCOLSCAN_LUHU*/     /* 0->79, 80-159 */
209  /*#define GHWCOLSCAN_LUHD*/       /* 0->79, 159-80 */
210  #define GHWCOLSCAN_HULD     /* 80-159, 79-0 */
211  /*#define GHWCOLSCAN_HDLD*/     /* 159-80, 79-0 */
```

Figura 103. *Archivo "gdispcfg.h".*

Figura 104. *Visualización por pantalla error.*

Como se puede observar, la "LÍNEA 7" y la "LÍNEA 8" aparecen dos veces, la primera vez, están situadas encima de la "LÍNEA 6", el problema es que están invertidas con respecto a los primeros números. La segunda vez que aparecen la "LINEA 7" y la "LINEA 8" en la mitad superior debajo de la "LINEA 9" sí están correctas.

Modificando "gdispcfg.h" de la manera mostrado en la figura 105, se obtiene el resultado indicado en la figura 106.

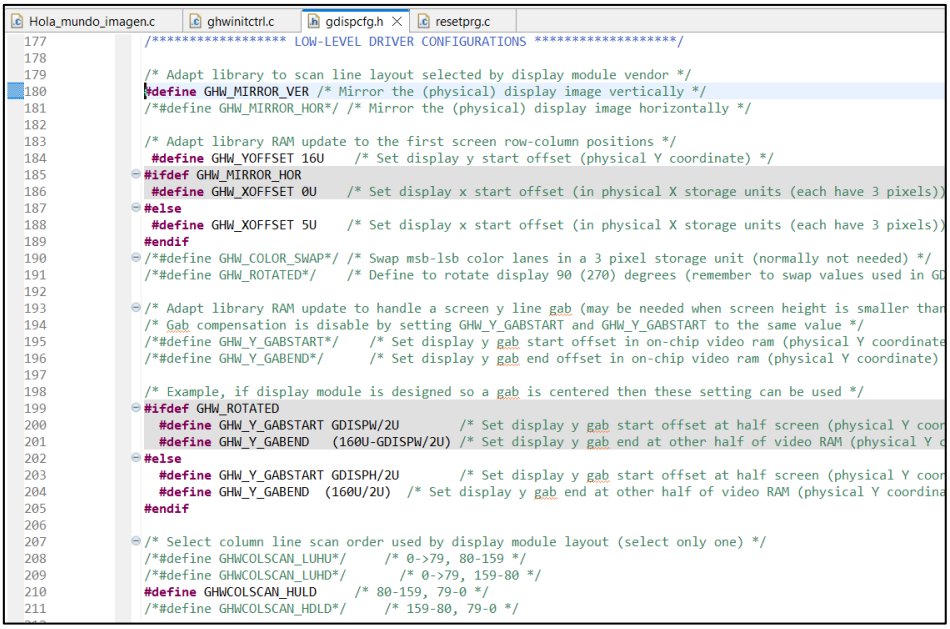


Figura 105. Archivo "gdispcfg.h".

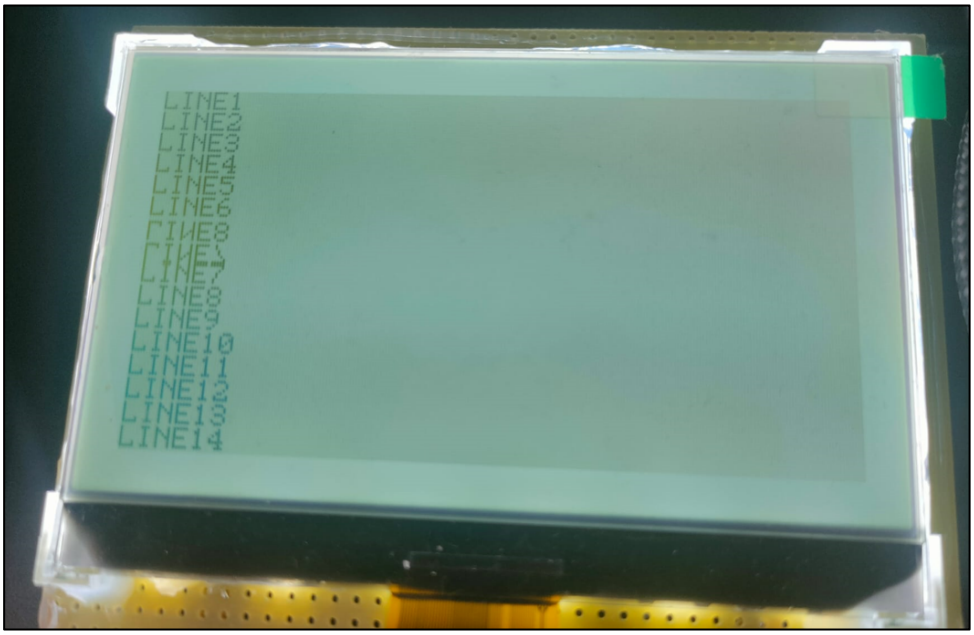


Figura 106. Visualización por pantalla error.

Nuevamente, las líneas "LINEA 7" y "LINEA 8" aparecen duplicadas. Y dan el mismo error que en la figura 104.

9.5.4. Solución a los correos

En este correo se pregunta acerca de la repercusión de eliminar los parámetros “GHW_Y_GABSTART” y “GHW_Y_GABEND”. Probado varias configuraciones se obtienen los siguientes resultados.

1. Con la configuración mostrada en la figura 107, se obtiene el resultado mostrado en la figura 108:

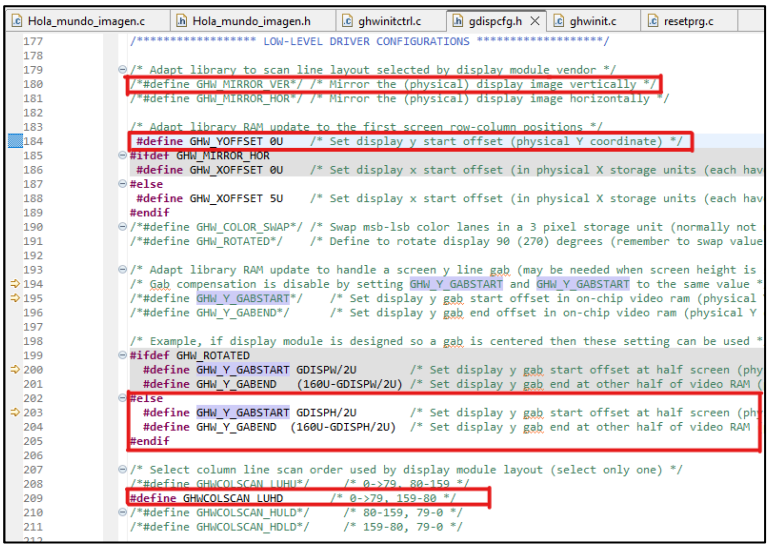


Figura 107. Inicializaciones de "ghwinitctrl.c" error.

Figura 108. Visualización por pantalla error.

2. Con la configuración mostrada en la figura 109, se obtiene el resultado mostrado en la figura 110:

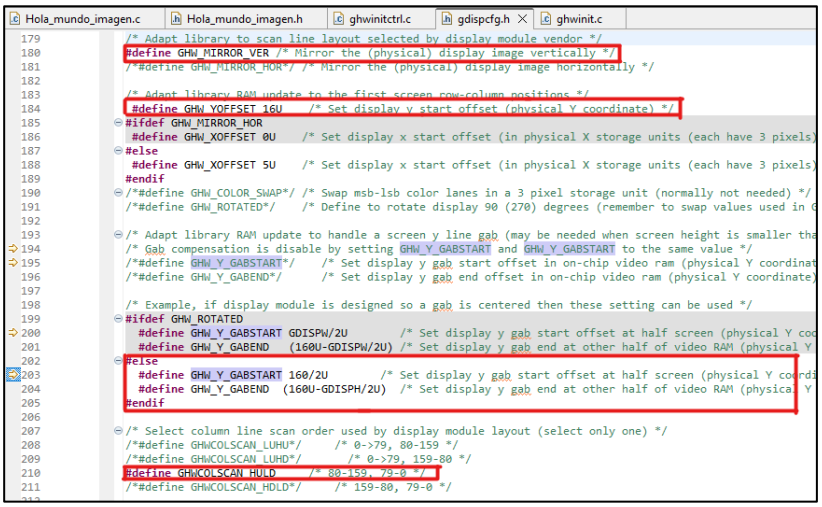


Figura 109. Inicializaciones de "ghwinitctrl.c" error.

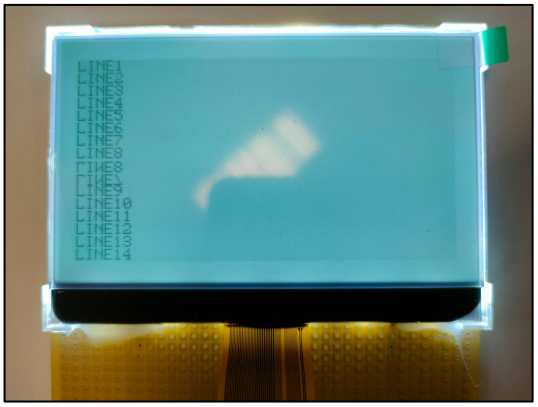


Figura 110. Visualización por pantalla error.

3. Con la configuración mostrada en la figura 111, se obtiene el resultado mostrado en la figura 112:

Figura 111. Archivo "gdispcfg.h".

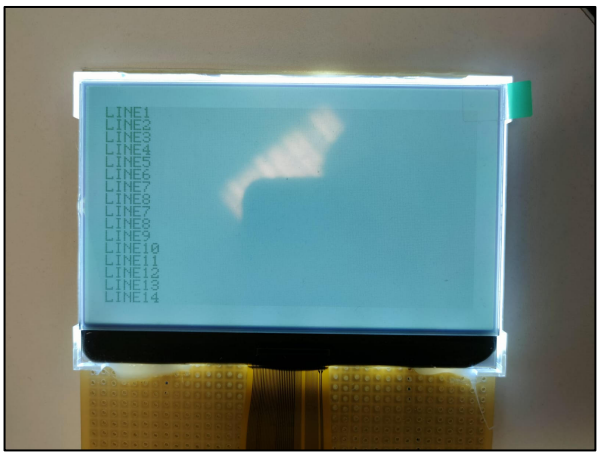


Figura 112. Visualización por pantalla error.

4. Finalmente, con cualquiera de las configuraciones mostradas en las figuras 113, 114 y 115, se obtiene el resultado mostrado en la figura 116:

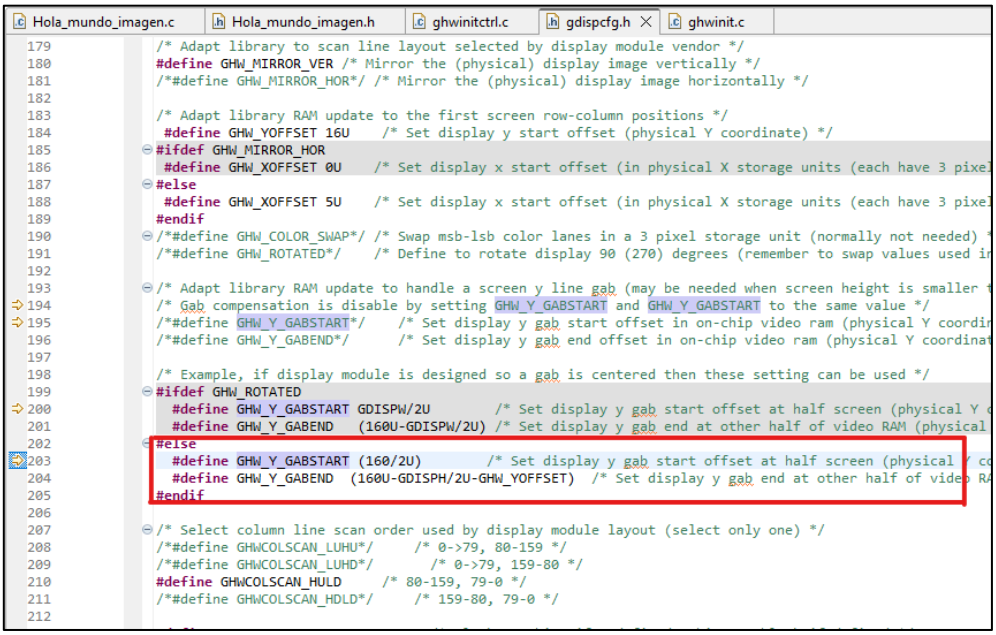


Figura 113. Archivo "gdispcfg.h".

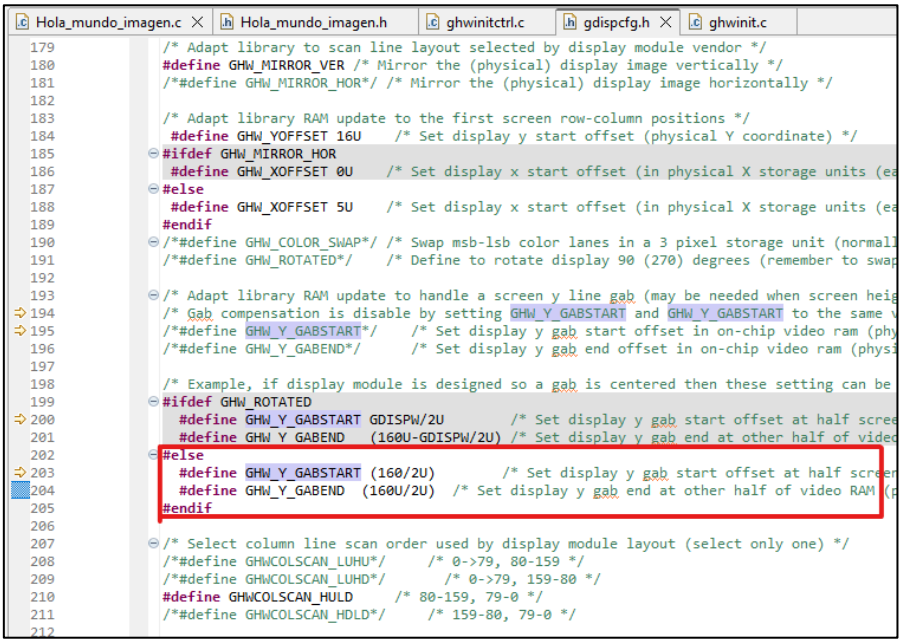


Figura 114. Archivo "gdispcfg.h".

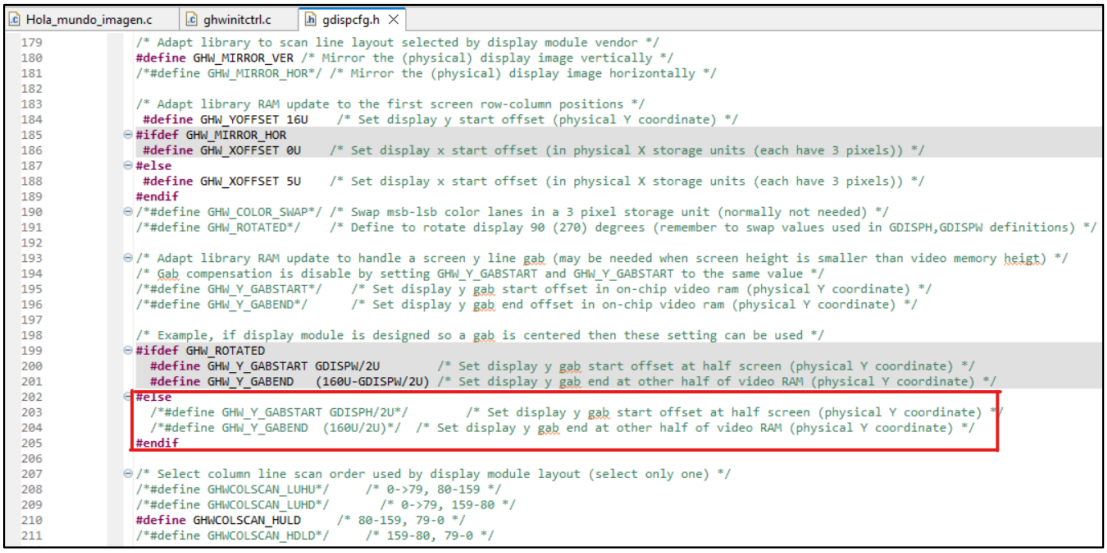


Figura 115. Archivo "gdispcfg.h" correcto.

Figura 116. Visualización por pantalla correcta.

La respuesta para solucionar el último problema mencionado, por parte del fabricante, fue que se deshabiliten las definiciones "/*#define GHW_Y_GABSTART*/" y "/*#define GHW_Y_GABEND*/". Esto es porque, sólo son necesarios cuando debe haber un hueco. Es decir, con "GHWCOLSCAN_HULD" y "#define GHW_YOFFSET 16U" ya se mueve el hueco para que esté antes de la primera línea y después de la última línea de visualización, la declaración de espacio es innecesaria.

9.5.5. Otro correo

En este correo se modifican algunos valores de la configuración de contraste y de inicialización. Con estos valores de inicialización (véase figura 117):

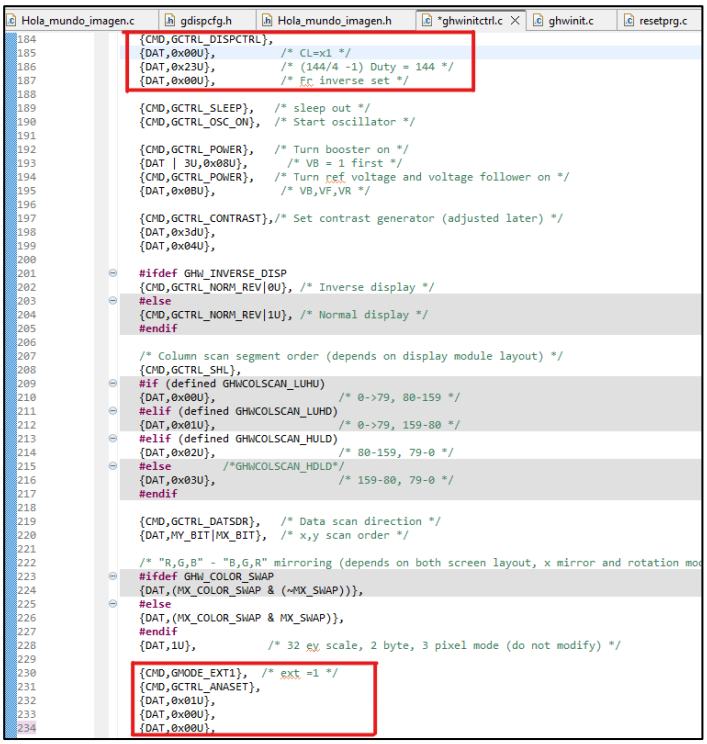


Figura 117. Inicializaciones de "ghwinitctrl.c" error.

Y con cualquiera de los valores de contraste mostrados en la figura 118 y 119, se obtiene de resultado la figura 120:

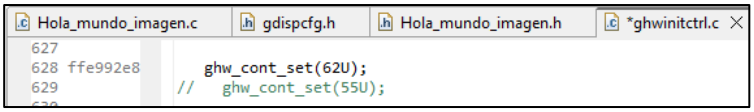


Figura 118. Modificación del valor de contraste.

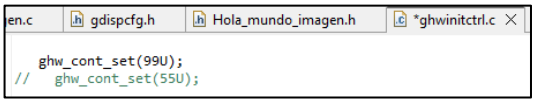


Figura 119. Modificación del valor de contraste.

Figura 120. Visualización por pantalla error.

El problema de estas configuraciones es que no se distinguen los tonos de gris.

Ajustando el primer valor de "GCTRL_DISPCTRL" a 4 (véase figura 121), obtenemos de resultado la figura 122:

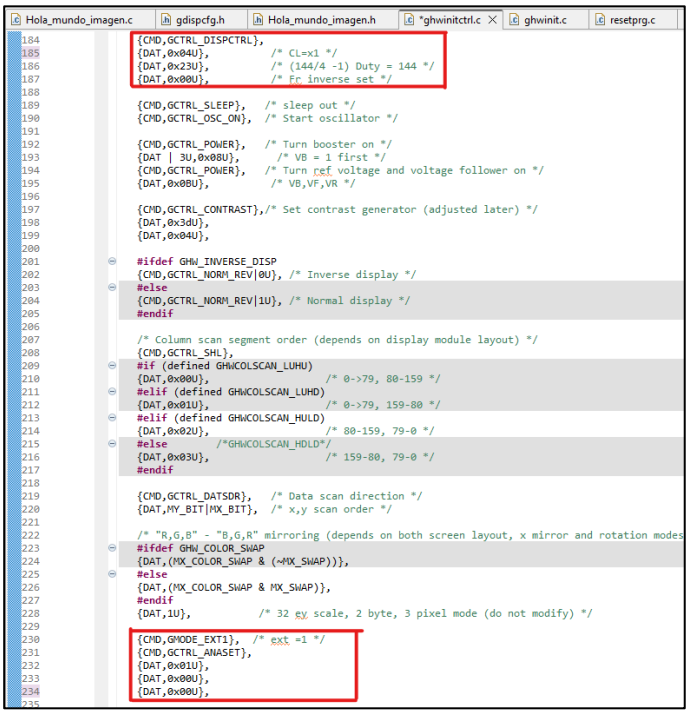


Figura 121. Inicializaciones de "ghwinitctrl.c" error.

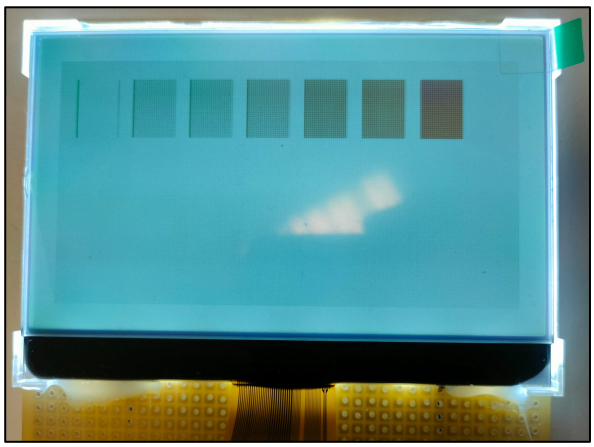


Figura 122. Visualización por pantalla error.

El problema es que parpadea demasiado.

9.5.6. Solución a los parpadeos

Para poder solucionar el fallo de los parpadeos ha sido requerido ajustar los comandos de inicialización de la manera mencionada en el punto *8.1.4. Archivo de origen "ghwinitctrl.c"*.

Haciendo modificaciones en la inicialización. Si se escribe un 2 en el tercer valor de "GCTRL_ANASET" y con un 0 en el primer valor de "GCTRL_DISPCTRL" (véase figura 123), se obtiene el resultado mostrado en la figura 124.

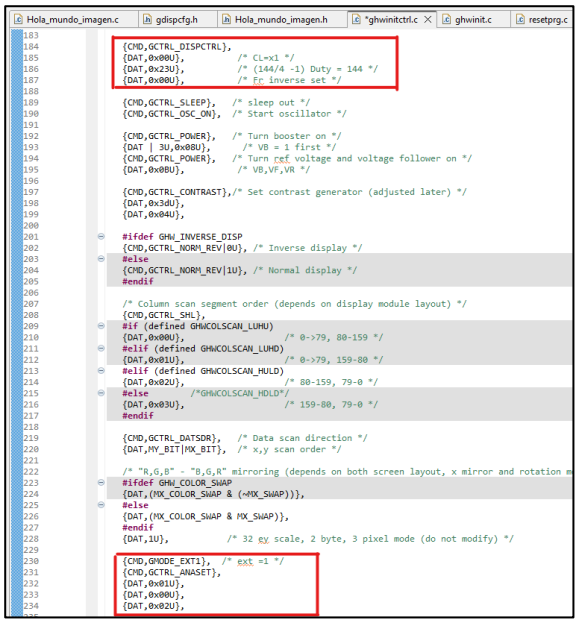


Figura 123. Inicializaciones de "ghwinitctrl.c" correcta.

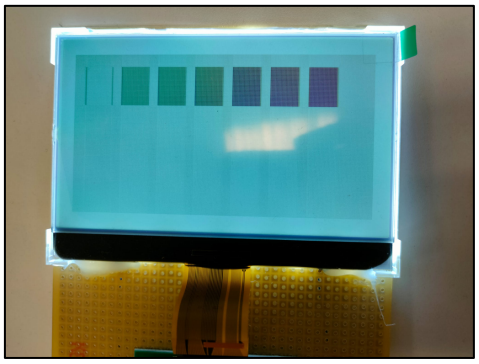


Figura 124. Visualización por pantalla correcta.

9.5.7. Otras dudas

Para poder mostrar por pantalla una imagen creada con IconEdit mientras se emplean las librerías, se pidió ayuda al fabricante. La solución aportada era escribir al inicio del código principal “extern GCODE GFONT FCODE nombre_symbolo_a_mostrar;” y colocar una función como la anterior pero sustituyendo el nombre de dicha función (en este caso “nombre_symbolo_a_mostrar”) por el nombre de la imagen que se desea mostrar (en este caso es “end_background_2” y cambiar los puntos en los que se quiere colocar “gputsym();”. También se puede modificar el valor que hay delante del nombre de la imagen a mostrar, el primer valor de “ggetfsym();”, en este caso 0U, ya que sólo hay una imagen. Pero puede haber más de un símbolo en el mismo archivo .c guardado (véase figura 62).

```
static void test_symbol_fondo(void)
    {
    PGSYMBOL ps;
    ps = ggetfsym(0U, &end_background_2);
    gputsym(0U,0U,ps); // Output symbol in
    }
```

Figura 125. Función “test_symbol_fondo” en el archivo principal del proyecto “Hola_mundo_imagen.c” para solucionar fallos.

También el fabricante modificó el modo de guardar la imagen en dicho programa, para que sea guardada como 4 bit grey-level symbol (debe guardarse como un símbolo de nivel de gris de 4 bits). Este es el modo erróneo como se guardaba anteriormente, por lo que daba fallo (véase figura 126):

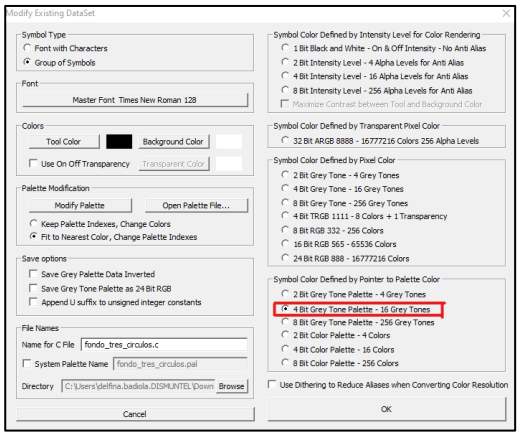


Figura 126. Error al guardar la imagen en IconEdit.

Se debe guardar como “4 Bit Grey Tone – 16 Grey Tones”, para asegurar que los píxeles del símbolo contienen toda la información del color gris.

9.6. Otros fallos

A la hora de utilizar la función de rotación, se creaban ciertas interferencias. Copiando un código que aparece en la explicación de las funciones de la librería (véase figura 127), se obtiene el resultado mostrado en la figura 128:

Example:

```
static struct
   {
   GSYMHEAD sh;
   SGUCHAR b[48];
   }
GCODE PCODE testsym2 =     /* Long box figure for rotation example */
   {{39, 9},{
   0xFFU,0xFFU,0xFFU,0xFFU,0xFFU,0x80U,0x00U,0x00U,0x00U,0x02U,
   0x80U,0x00U,0x00U,0x00U,0x02U,0x80U,0x00U,0x00U,0x00U,0x02U,
   0x80U,0x00U,0x00U,0x00U,0x02U,0x80U,0x00U,0x00U,0x00U,0x02U,
   0x80U,0x00U,0x00U,0x00U,0x02U,0x80U,0x00U,0x00U,0x00U,0x02U,
   0xFFU,0xFFU,0xFFU,0xFFU }};

void main(void)
   {
   SGINT x,y;
   float angle;
   int degree;
   SGUCHAR modeselect;
   const GMODE mode[9] = { /* All possible connection points */
      GALIGN_TOPS    GALIGN_LEFT, GALIGN_TOP|    GALIGN_HCENTER,  GALIGN_TOP|
                                                                  GALIGN_RIGHT,
      GALIGN_VCENTER|GALIGN_LEFT, GALIGN_VCENTER|GALIGN_HCENTER,
                                                  GALIGN_VCENTER|GALIGN_RIGHT,
      GALIGN_BOTTOM |GALIGN_LEFT, GALIGN_BOTTOM| GALIGN_HCENTER, GALIGN_BOTTOM|
                                                                  GALIGN_RIGHT};

   ginit();
   x = GDISPW/2U;
   y = GDISPH/2U;
   gsetupdate(GUPDATE_OFF);

   /* Show rotation with all connection point options */
   for (modeselect = 0U; modeselect < 9U; modeselect++)
      {
      /* Rotate symbol around connection point */
      for (degree = 0; degree <= 360; degree += 15)
         {
         angle = ((float)degree * PI)/((float)180);
         /* Draw symbol */
         gputsymrot(x,y, angle, (PGSYMBOL) &testsym2, 0, 0, mode[modeselect]);

         /* Mark rotation center */
         gcircle(x,y,3,0U);
         gsetupdate(GUPDATE_ON);
         /* Drawing completed */

         if ((modeselect == 8) && (degree == 360))
            {
            return; /* Exit her to keep last image */
            }

         /* Add any delay here if there is a need to increase the time
            each animation step is shown */

         /* Clear symbol (using exact same coordinates) */
         gsetupdate(GUPDATE_OFF);
         gputsymrot(x,y, angle, (PGSYMBOL) &testsym2, 0, 0,
                                              GSYMCLR|mode[modeselect]);
         }
      gsetupdate(GUPDATE_ON);
      }
   }
```

The 9 anchor connection points:

145

Figura 127. Código sacado del manual.

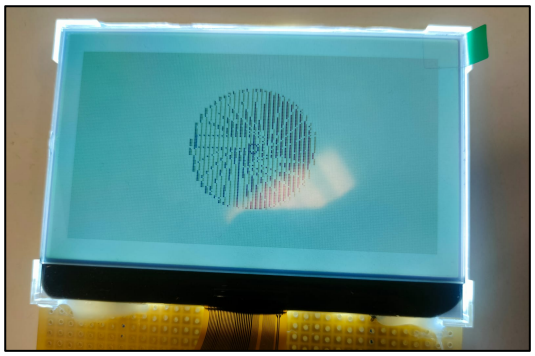
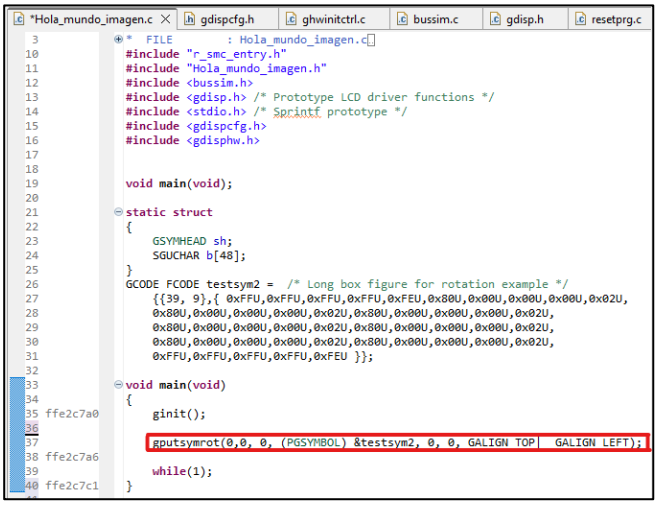


Figura 128. *Error de visualización por pantalla.*

Como se puede observar, se crean dentro de la barra (que debería estar vacía) líneas verticales de distintos tonos de gris (véase figura 129). Esto sólo sucede cuando se rota un objeto, es decir, cuando la barra se imprime directamente en la pantalla.

Figura 129. *Correcta visualización por pantalla.*

Pero rotando el símbolo en el mismo sitio (véase figura 130), aparecen interferencias en la barra.



```
*Hola_mundo_imagen.c | gdispcfg.h | ghwinitctrl.c | bussim.c | gdisp.h | resetprg.c
 3          *  FILE        : Hola_mundo_imagen.c
10          #include "r_smc_entry.h"
11          #include "Hola_mundo_imagen.h"
12          #include <bussim.h>
13          #include <gdisp.h> /* Prototype LCD driver functions */
14          #include <stdio.h> /* Sprintf prototype */
15          #include <gdispcfg.h>
16          #include <gdisphw.h>
17
18
19          void main(void);
20
21          static struct
22          {
23              GSYMHEAD sh;
24              SGUCHAR b[48];
25          }
26          GCODE FCODE testsym2 =  /* Long box figure for rotation example */
27              {{39, 9},{ 0xFFU,0xFFU,0xFFU,0xFFU,0xFEU,0x80U,0x00U,0x00U,0x00U,0x02U,
28              0x80U,0x00U,0x00U,0x00U,0x00U,0x02U,0x80U,0x00U,0x00U,0x00U,0x00U,0x02U,
29              0x80U,0x00U,0x00U,0x00U,0x00U,0x02U,0x80U,0x00U,0x00U,0x00U,0x00U,0x02U,
30              0x80U,0x00U,0x00U,0x00U,0x00U,0x02U,0x80U,0x00U,0x00U,0x00U,0x00U,0x02U,
31              0xFFU,0xFFU,0xFFU,0xFFU,0xFEU }};
32
33          void main(void)
34          {
35  ffe2c7a0        ginit();
36
37              gputsymrot(0,0, 0, (PGSYMBOL) &testsym2, 0, 0, GALIGN_TOP | GALIGN_LEFT);
38  ffe2c7a6
39              while(1);
40  ffe2c7c1    }
```

Figura 130. *Única función que aparece en el código.*

Es exactamente el mismo símbolo, pero con la función de rotación cero grados (véase figura 131).

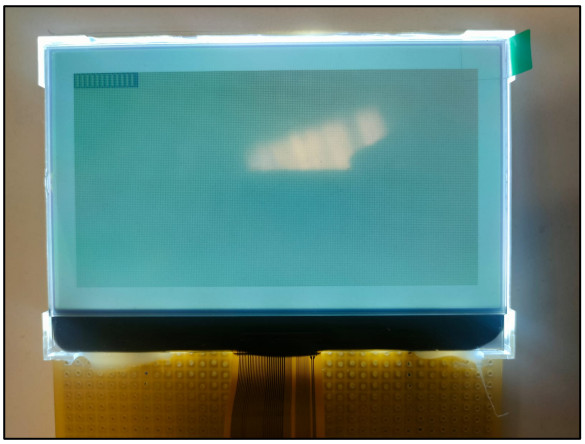


Figura 131. Error de visualización por pantalla.

Para corregir el fallo, es necesario habilitar la siguiente definición “GBUFFER”. De este modo, se ve correctamente la imagen girada.

9.7. Letras

De las últimas pruebas realizadas, consisten en imprimir distintos tipos de letra por pantalla, para ver todas las fuentes que compró la empresa. El problema que tienen los archivos que contienen los tipos de letra es que ocupan demasiada memoria, por ello, para poder imprimir varios tipos diferentes de ellos se deben guardar en el proyecto unos cuatro tipos de letra diferentes como mucho (ya que hay algunas fuentes que ya vienen por defecto).

Como se puede ver en la tabla 24, los tipos de letras ocupan demasiada memoria.

Nombre de la fuente	Tamaño
ariel18	1570
ariel9	436
cp8859_14	1e00
cp8859_9	b00
footnote	a00
mono5_8	a00
mono8_8	500
ms58p	a00
msfont58	a00
msfont78	b00
narrow10	558
narrow10_w	1938
narrow13_e	2544
narrow15	e40
narrow15_w	4340
narrow18_e	5e68
narrow20	12b4
narrow20_w	5844
narrow24_e	7c38
times13	5be
times16	d04
times9	436
uni_16x16	4510
B_12x20_b_g2	7ec0
B_12x20_g2	7ec0
B_12x24_b_g2	9684
B_12x24_g2	9684

Tabla 24. Espacio que ocupan algunos tipos de letras.

Para obtener el tamaño de estas fuentes, es necesario entrar dentro de la carpeta “HardwareDebug”, en el archivo con el nombre del archivo principal y con terminación “.map” (es

decir “Hola_mundo_imagen.map” en este proyecto, véase figura 132). Una vez se muestran los tipos de letra por pantalla, (es decir, al ejecutar con “Debug”), aparecerá en este archivo el espacio que ocupan las fuentes que se han ejecutado.

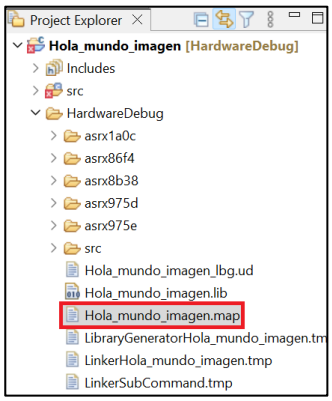


Figura 132. *Ubicación del archivo para encontrar el tamaño que ocupan los tipos de letras.*

A continuación, se muestra un fallo que generaba la visualización de las fuentes con grises (véase figura 133).

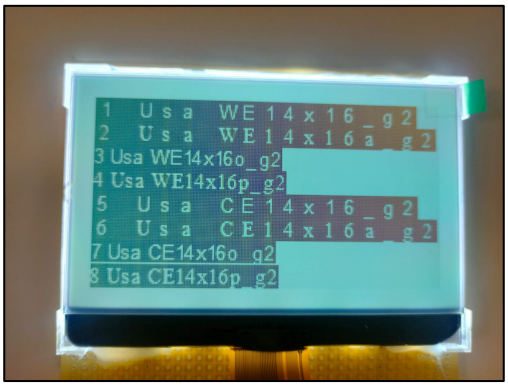


Figura 133. *Error de visualización del tipo de letra por pantalla.*

Como se puede ver en la figura 133, el fondo es negro cuando las letras deberían ser las que estén en negro (con tonos de grises) y el fondo en blanco.

Para solucionar este fallo, era necesario activar "GHW_INVERTGRAPHIC_SYM".

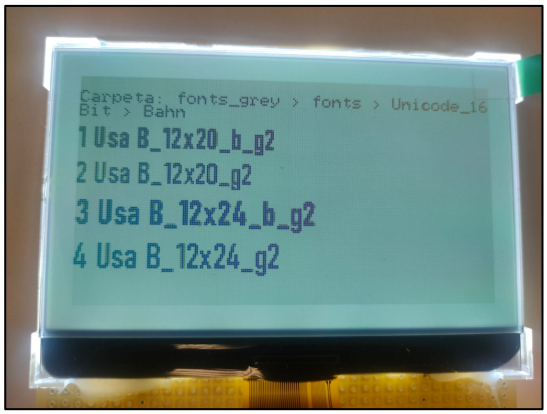


Figura 134. *Correcta visualización del tipo de letra por pantalla.*

Tal como se muestra en la figura 134, se ve una vez solución al error. Pero finalmente se decidió utilizar letras en negro, sin tono de grises, ya que se ven más nítidas. A continuación, se comparten dos ejemplos de varias fuentes que se probaron.

Estas son algunos de los tipos de letras que vienen por defecto (véase figura 135).

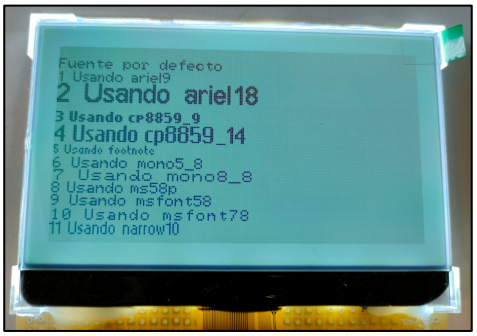


Figura 135. Correcta visualización del tipo de letra por pantalla.

Y estos son la misma fuente con diferentes tamaños y espacios. Estas fuentes son algunas de las compradas por la empresa (véase figura 136).

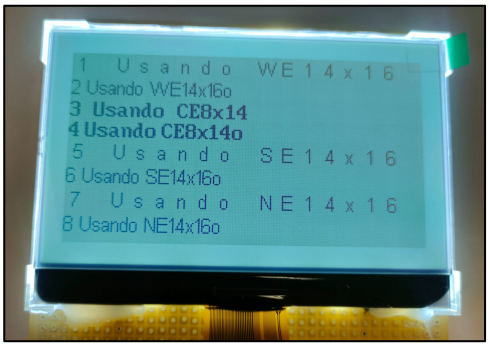


Figura 136. Correcta visualización del tipo de letra por pantalla.

9.8. Últimas pruebas

Para que se modifiquen cada uno de los gráficos, lo recomendable es ir probando el funcionamiento de uno a uno. En el caso de la figura 137, primero se creó la función de "drawColumn" para comprobar el funcionamiento de esta, y por último probar según la cantidad de dígitos del valor del porcentaje, la posición más adecuada en la pantalla.

Figura 137. Pruebas con la función "drawColumn".

En las tres siguientes imágenes se muestra cómo se modificaba la posición de los valores del gráfico en forma de arco de la derecha.

- Con un dígito (véase figura 138):

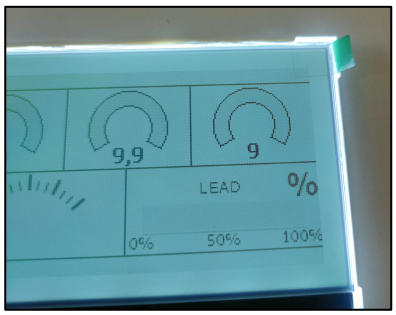


Figura 138. *Pruebas para encontrar la posición centrada de los valores en el gráfico en forma de arco.*

- Con dos dígitos (véase figura 139):

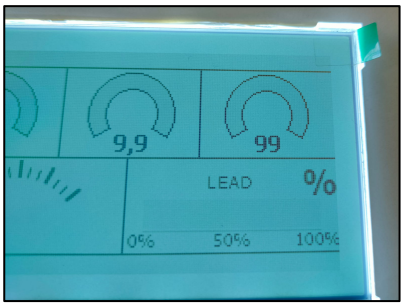


Figura 139. *Pruebas para encontrar la posición centrada de los valores en el gráfico en forma de arco.*

- Con el máximo de dígitos (véase figura 140):

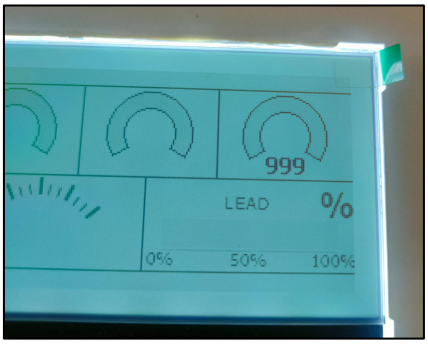


Figura 140. *Pruebas para encontrar la posición centrada de los valores en el gráfico en forma de arco.*

Lo mismo se debe ejecutar con las posiciones de cada uno de los gráficos en tipo arco y cada uno de los tres tipos de dígitos que pueden llegar a tener.

También se realizada de la misma forma para colocar la unidad de medidas en el centro del círculo, la diferencia es que este será siempre igual e irá en la misma posición.

En el último gráfico, se realiza algo similar. Se pide añadir el valor de los grados centígrados que marca el gráfico tipo velocímetro.

Al comienzo se optó por colocar los grados en la parte derecha, pero se solicitó la modificación de este gráfico (véase figura 141).

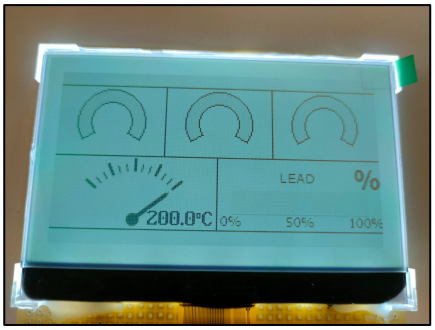


Figura 141. Pruebas con el gráfico tipo velocímetro.

Finalmente, se modificó de la siguiente manera para que la flecha marcadora fuese más corta y estuviese la medida de la temperatura debajo (véase figura 142).

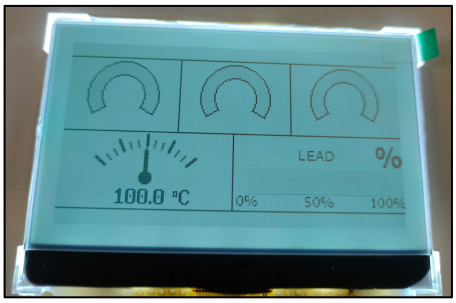


Figura 142. Pruebas con el gráfico tipo velocímetro.

Se realizó una prueba para verificar que, en cada dígito todos los valores estuviesen centrados (véase figura 143).

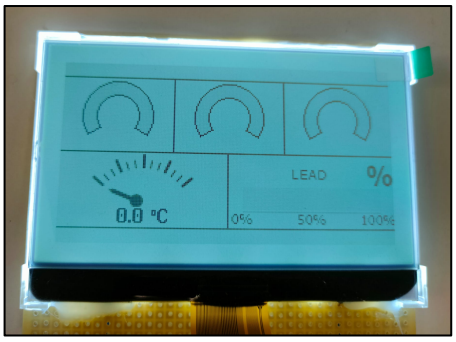


Figura 143. Pruebas con el gráfico tipo velocímetro.

Lo más conveniente después de haber realizado todas estas pruebas, es hablar con el fabricante si aparece algún fallo que no se conoce su solución. El fabricante contesta rápido y es de gran ayuda. Además, también se debe pedir al fabricante que proporcione las especificaciones necesarias para el LCD que se quiere usar con las mediadas y el controlador que se utiliza.

10. Conclusión

En este punto del trabajo, es pertinente efectuar una vinculación entre los resultados obtenidos y los objetivos establecidos inicialmente para el proyecto. Es grato destacar que los propósitos planteados han sido efectivamente alcanzados.

En lo que respecta al objetivo principal del proyecto, que radica en el desarrollo de una interfaz gráfica de usuario para el control de grupos electrógenos, se ha logrado cumplir satisfactoriamente. Los avances realizados en este sentido han permitido crear una herramienta visualmente intuitiva y funcionalmente eficaz para la supervisión y el control de estos sistemas de energía.

Pasando a los subobjetivos específicos, el primero de ellos, relacionado con la caracterización de la pantalla, ha sido cubierto en gran medida en la sección *8.1.4. Archivo de origen "ghwinitctrl.c"*. Este subobjetivo apunta a comprender a fondo la funcionalidad y especificaciones de la pantalla LCD en la que se implementa la aplicación. Este primer subobjetivo abarca y cumple con los siguientes dos propósitos:

- Por un lado, la determinación de las opciones de reloj se trata en el apartado *6.1.1. Modificando "Clocks"*. Esta sección se refiere a la configuración de la frecuencia de reloj, que determina la velocidad a la que se ejecutan las instrucciones.

- Por otro lado, la programación de los registros se refiere al proceso de configurar y controlar los registros del microprocesador. Se trata en el apartado *6. Integración del hardware*.

El segundo subobjetivo, se centra en el diseño de objetos gráficos, es decir, de los elementos visuales que se utilizarán en una interfaz gráfica de usuario. Como pueden ser los gráficos, imágenes y cualquier otro componente visual que mejore la interacción del usuario con la aplicación o el sistema. Este subobjetivo se ha desarrollado con éxito en la sección *7. Desarrollo gráfico de widget con IconEdit*.

Finalmente, el tercer subobjetivo, correspondiente al desarrollo de la librería, se ha cumplido en su totalidad en la sección *8. Programación e integración*. A lo largo de este apartado, se ha elaborado un conjunto de funciones y procedimientos que han permitido la integración y programación exitosa de la interfaz gráfica en el sistema empotrado. Así mismo, se ha utilizado una librería que contiene código para realizar ciertas funciones ya hechas que el programador pueda reutilizar y configurar para que se facilite y agilice el proceso de desarrollo software.

Por todo ello, se puede concluir que los resultados obtenidos en este proyecto reflejan el logro de los objetivos planteados inicialmente, proporcionando una interfaz gráfica eficiente para el control de grupos electrógenos y demostrando el valor de la integración de hardware y software en sistemas empotrados.

10.1. Relación con los estudios cursados

Este proyecto está relacionado con los estudios cursados en el Grado en Ingeniería Electrónica Industrial y Automática. De hecho, se han aplicado algunos de sus contenidos formativos, puesto que consta de:

- Programación: La programación que se emplea en este proyecto utiliza lenguaje en c, que ha sido adquirido durante el Grado, a través de la asignatura de “Informática” de primero, y de “Informática Industrial I” e “Informática Industrial II” de tercero.

- Competencias adquiridas: En las asignaturas de “Informática Industrial I” e “Informática Industrial II” de tercero, además de proporcionar conocimientos de programación básicos, se ha adquirido las competencias necesarias para implementar proyectos informáticos modulares y para diseñar y desarrollar sistemas informáticos basados en microcontroladores. Se podría decir pues, que no ha sido complicado el aprender el funcionamiento de este nuevo entorno de desarrollo gracias a lo estudiado en el Grado, pero ha sido necesario dedicar mucho tiempo para lograr moverse por él de forma rápida y eficiente.

- Gestión de interfaz de usuario: Todo lo relacionado con la interfaz de usuarios, no ha sido demasiado complejo el desarrollarlo, puesto que, gracias a la asignatura “Informática Industrial I”, se poseían conocimientos para implementar proyectos informáticos modulares. Esto implica diseñar sistemas que incorporen técnicas para gestionar interfaces de usuario y de proceso, manejar la imagen del sistema y planificar tareas.

- Desarrollo gráfico: Por último, gracias a las asignaturas de “Expresión gráfica” ha sido sencillo comprender el entorno de desarrollo de IconEdit. Esto es debido a que, durante las sesiones prácticas de la asignatura, los estudiantes aprendieron a utilizar herramientas de diseño asistido por ordenador (CAD), permitiendo crear proyectos gráficos propios utilizando la tecnología que actualmente se utiliza en la industria.

10.2. Conocimientos adquiridos

Con la realización de esta memoria se han adquirido los siguientes conocimientos:

- Con el entorno de desarrollo e2Studio: Como se explica a lo largo de este TFG, el alumno obtiene habilidades esenciales para programar, depurar y codificar eficazmente aplicaciones de microcontroladores. Asimismo, se aprende a trabajar con una variedad de funciones de diagnóstico y optimización que pueden ayudar a mejorar la eficiencia y la eficacia del software (como puede ser el modo de trabajo “Smart Configurator”).

- Con el entorno de desarrollo IconEdit: Se adquieren habilidades importantes para la creación de gráficos para aplicaciones de software. Se aprende a trabajar con paletas de colores, a importar y exportar imágenes en una variedad de formatos, y a optimizar gráficos para su uso en aplicaciones específicas. Además, se aprende a integrar gráficos directamente en el software.

También se han adquirido conocimientos relacionados con la redacción de la memoria:

- Elaborar la memoria.

- Incluir referencias en Word.

- Añadir, eliminar y modificar estilos.

- Crear índices de figuras.

- Emplear las figuras y referenciarlas.
- Utilizar la herramienta de Zotero.

Las principales competencias transversales desarrolladas en este TFG son las siguientes:

- Comprensión e integración.
- Aplicación y pensamiento práctico.
- Análisis y resolución de problemas.
- Innovación, creatividad y emprendimiento.
- Diseño y proyecto.
- Comunicación efectiva.
- Aprendizaje permanente.
- Planificación y gestión del tiempo.

10.3. Trabajos futuros

En trabajos futuros, se tiene pensado modificar la interfaz gráfica de usuario vista en el capítulo *7. Desarrollo gráfico de widget con IconEdit*. Por un lado, se ha planteado la idea de sustituir los tres gráficos tipo arco superiores (debido a que ocupan demasiado espacio de memoria del microcontrolador) por controladores de batería y de alarmas. También se va a modificar el gráfico en forma de velocímetro para controlar la cantidad de combustible.

Por otro lado, también se ha planteado añadir en la interfaz la propiedad de paginación, para permitir mostrar más controles en la pantalla de manera lo más intuitiva posible.

Una vez se configuren todos los controles que se desean implementar se pretende incorporar estas interfaces para el control de nuevos grupos electrógenos con los que va a trabajar la empresa. Y se modificarán los ya existentes que utilizan pantallas de retroiluminación de 4 líneas por 20 dígitos por estas nuevas pantallas más sofisticadas.

11. Referencias

[1] «Dismuntel_Info», 13 de abril de 2022. https://www.dismuntel.com/empresa/ (accedido 3 de julio de 2023).
[2] «Dismuntel_Ingeniería», 14 de febrero de 2022. https://www.dismuntel.com/servicios-dismuntel/ingenieria/ (accedido 3 de julio de 2023).
[3] «Dismuntel_Software», 7 de abril de 2022. https://www.dismuntel.com/servicios-dismuntel/software-dismuntel/ (accedido 3 de julio de 2023).
[4] «Interfaz de línea de comandos o CLI», *ComputerWeekly.es*. https://www.computerweekly.com/es/definicion/Interfaz-de-linea-de-comandos-o-CLI (accedido 3 de julio de 2023).
[5] «Interfaz de usuario, ejemplos y tipos.» http://global.tiffin.edu/noticias/interfaz-de-usuario-ui-ejemplos-y-tipos (accedido 3 de julio de 2023).
[6] «Interfaz táctil».
[7] C. Fernández, «Interfaz de Usuario de Voz o VUI», *ABAMobile*, 21 de abril de 2021. https://abamobile.com/web/que-es-vui-voice-user-interface-o-interfaz-de-usuario-de-voz/ (accedido 3 de julio de 2023).
[8] «Interfaz de usuario de lenguaje natural». https://es.dbpedia.org/page/Interfaz_de_usuario_de_lenguaje_natural (accedido 3 de julio de 2023).
[9] R. M. C. González, D. M. García, D. R. Ramey, y J. V. Llop, «Interfaces Usuario-Máquina.».
[10] «Interfaz de Realidad aumentada», *Rock Content - ES*, 15 de diciembre de 2019. https://rockcontent.com/es/blog/realidad-aumentada/ (accedido 3 de julio de 2023).
[11] «IconEditManual.pdf».
[12] «IconEditQuickGuide.pdf».
[13] «microcontrolador_RX671.pdf».
[14] «ST7529_Sitronix.pdf».
[15] «Target_board_RX671.pdf».
[16] «Adaptador para LCD». https://www.amazon.es/sourcing-map-Convertidor-Impresora-Port%C3%A1til/dp/B09XMF4NLZ/ref=sr_1_3?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&amp&crid=1ELO9VYF4RJSK&amp&keywords=adaptador+ffc+36&amp&qid=1679304771&amp&sprefix=adaptador+ffc+36%2Caps%2C109&amp&sr=8-3 (accedido 3 de julio de 2023).
[17] «Displaytech_128240D.pdf».
[18] «manual_usuario_serie_RX600_de_familia_RX.pdf».
[19] «gclcd_reference_manual_1811247.pdf».

12. Anexos

A continuación, se aportará la documentación adicional utilizada para la elaboración del proyecto. Entre ellos se encuentra el manual de usuario, por si se quiere realizar exactamente el mismo proyecto al desarrollado y, la visualización de los píxeles en el controlador ST7529.

12.1. Manual de usuario

12.1.1. Guía rápida de instalación del entorno de desarrollo e2Studio

Para descargar el programa de e2Studio, es necesario acceder a la página de RENESAS y escribir en el buscador “e2Studio - information for RX Family” y, una vez en dicha página, se da al botón de “Download Upgrade IDE”. En la parte inferior de esta página hay una apartado en el que se muestran vídeos que también explican paso a paso la instalación del programa y la creación de proyectos.

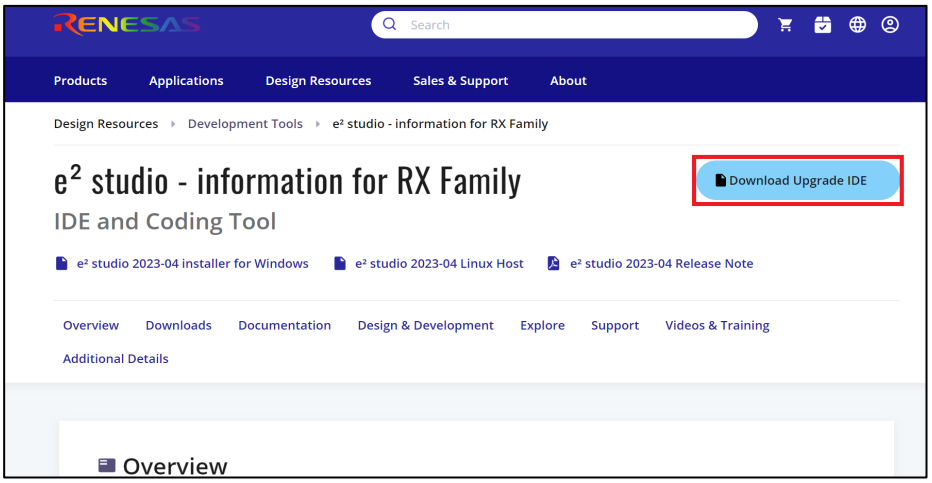


Figura 144. *Página con instrucciones de instalación, archivos y documentos del programa e2Studio*

Después de hacer clic sobre el botón, se abre otra página con los términos, una vez leídos se presiona el botón de “Accept and download” y se descarga un archivo tipo zip.

Figura 145. *Términos de instalación del entorno de desarrollo e2Studio*

A continuación, se accede a la carpeta de "Descargas" del ordenador en el que se desea realizar la instalación, se descomprime el archivo zip y se pulsa sobre el nuevo archivo generado. Saldrá una ventana en la que aparezca un cartel de "Windows protegió su PC", se hace clic sobre "Más información" y sobre "Ejecutar de todas formas". Se abre una nueva ventana que muestra el porcentaje de extracción.

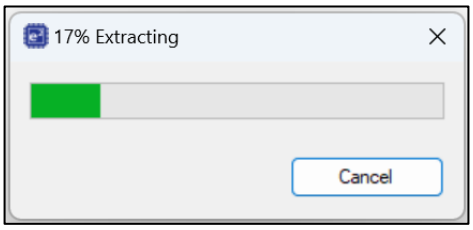


Figura 146. *Porcentaje de extracción*

Una vez se llega a la extracción total, aparece otra ventana y se selecciona la opción de instalar el programa para todos los usuarios "All Users".

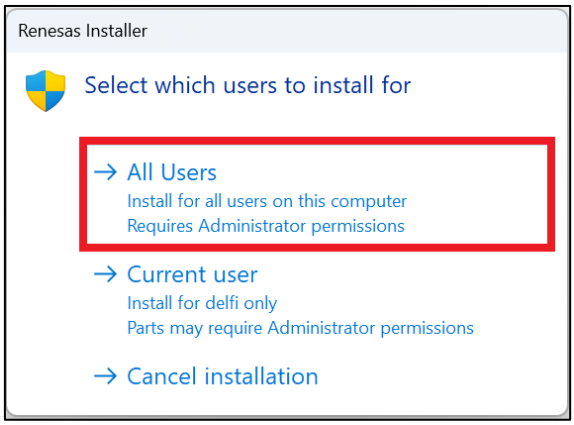


Figura 147. *Selección de instalación para los usuarios*

Aparece una advertencia en el que aparece escrito “¿Quieres permitir que esta aplicación haga cambios en el dispositivo?” y se selecciona “Sí”.

Ahora comienzan los últimos pasos para completar la instalación del programa. Aparece una ventana de Renesas con la última actualización del entorno de desarrollo de e2Studio. Una vez aparecen los cuatro círculos en verde con un tick en el centro, se presiona el botón de “Next”.

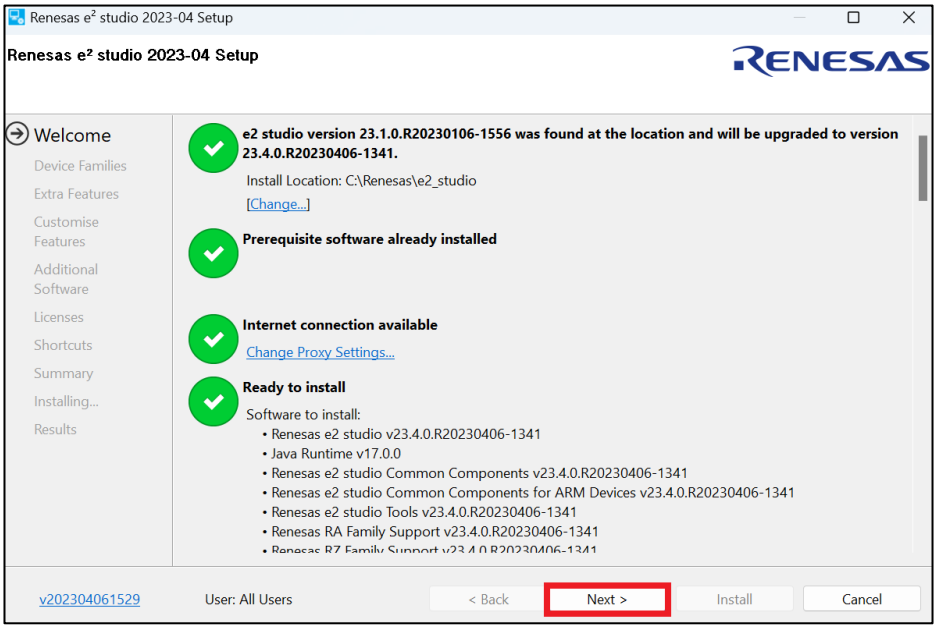


Figura 148. Comprobación de que todo sea correcto para la instalación del programa

En el siguiente apartado, se selecciona únicamente el tipo de familia con el que se va a trabajar, en este caso la “Familia RX”, y se pasa a la siguiente opción.

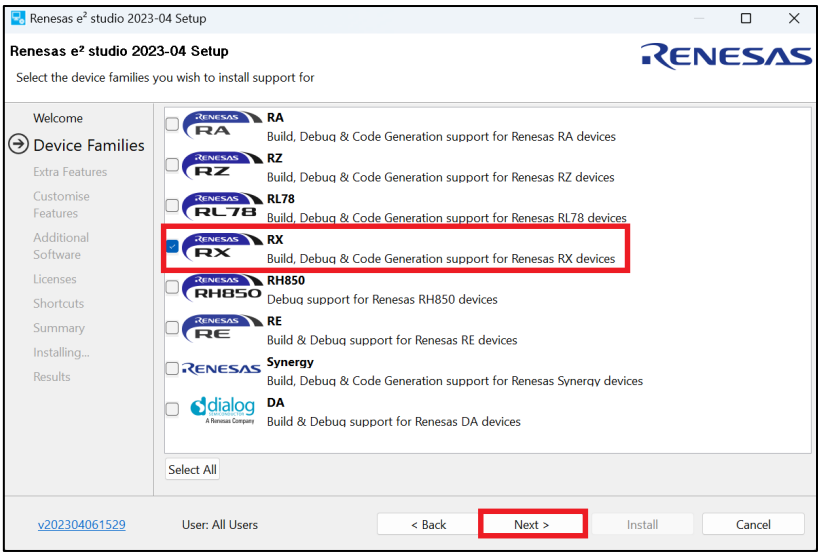


Figura 149. Comprobación de que todo sea correcto para la instalación del programa

En la siguiente opción, se seleccionan los componentes que se van a usar. En el caso de este proyecto, solo son necesarios los que vienen por defecto. Por lo que se pulsa el botón de “Next”.

En la siguiente página, aparece la selección de software. Primero, en el apartado “Renesas QE (1)” se selecciona únicamente la opción de “QE for Capacitive Touch”.

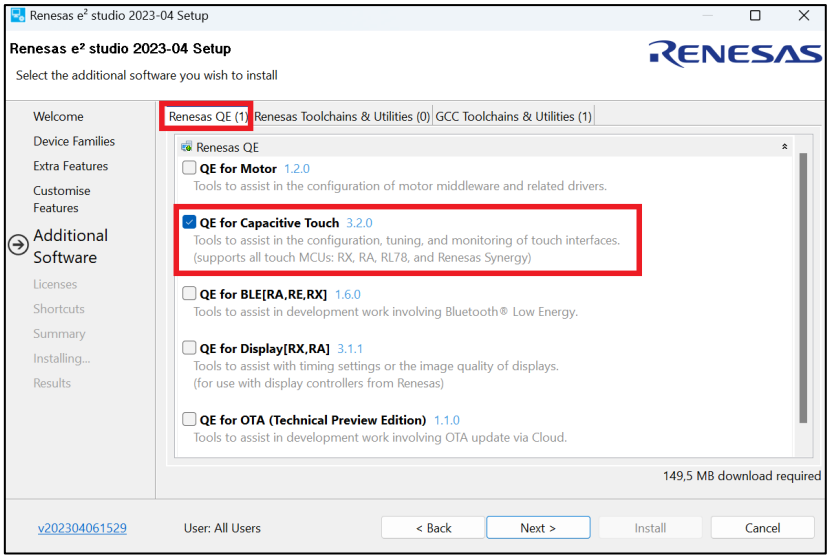


Figura 150. *Selección de software, Renesas QE*

En la misma página, se selecciona el apartado de “Renesas Toolchains & Utilities (1)”, se escoge la última versión del compilador CCRx, en este caso “Renesas CC-RX v2.08.01” y se presiona el botón de “Next”.

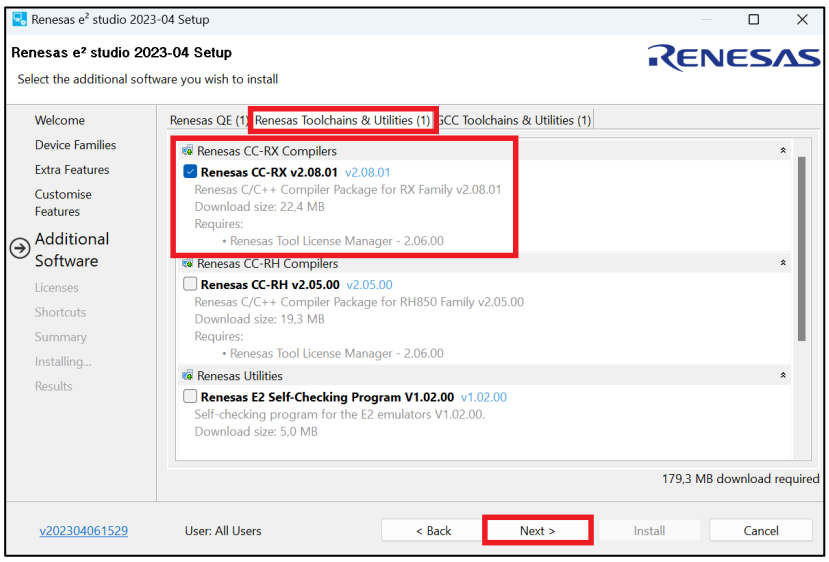


Figura 151. *Selección de software, compilador CC-RX de Renesas*

Seguidamente, aparecen los términos de la licencia, una vez leídos, se debe hacer clic sobre la opción de “I accept the terms of Software Agreements” para aceptar los términos y se pulsa “Next”.

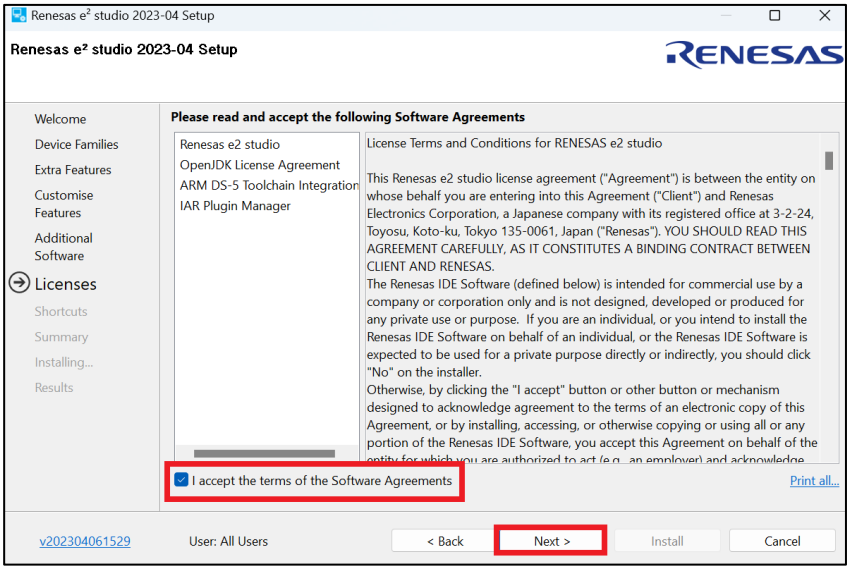


Figura 152. Aceptación de los términos de licencia

La siguiente página pregunta por la ubicación de los archivos y proyectos en los que se crearán accesos directos. Se ofrece una ubicación por defecto, por lo que se presiona el botón de “Next”.

Una vez se carga el resumen de todos los apartados finales y todo es correcto, se presiona el botón de “Install”.

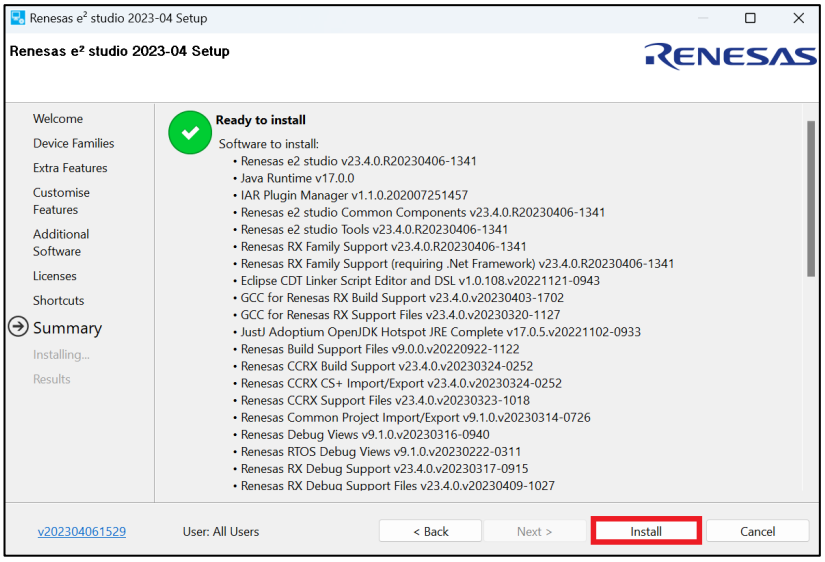


Figura 153. Aceptación de los términos de licencia

A continuación, comienza la instalación del entorno de desarrollo.

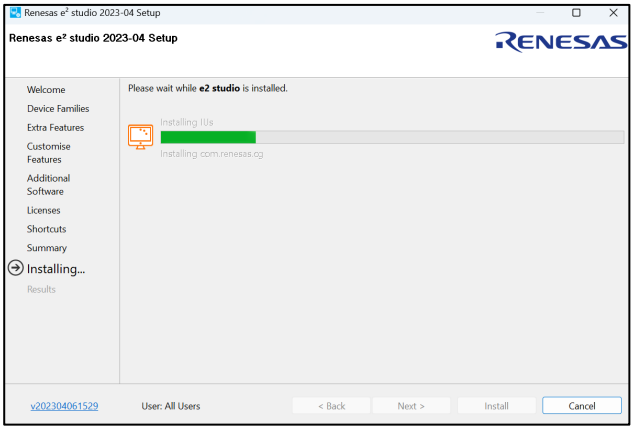


Figura 154. *Instalación del programa*

Para completar la instalación del programa que se puede ver en la Figura 12, van a aparecer una serie de ventanas más pequeñas para las configuraciones finales.

La primera ventana en aparecer es sobre el compilador CC-RX. Primero se pulsa el botón de “Next”.

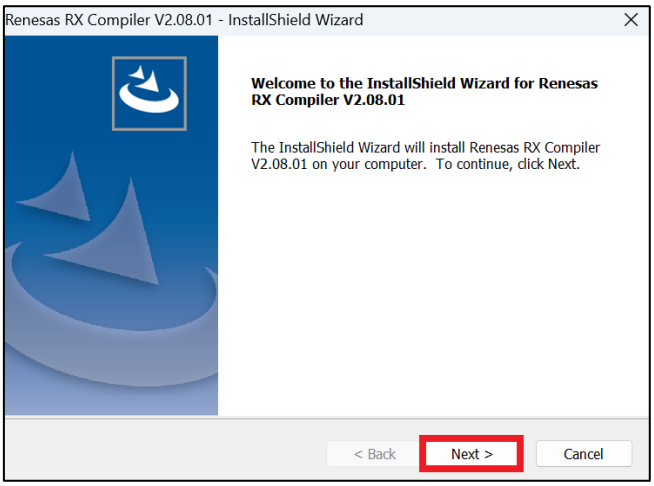


Figura 155. *Compilador CC-RX*

En la misma ventana, aparecen los términos del compilador, se deben aceptar y pulsar el botón de “Next” para continuar con la instalación.

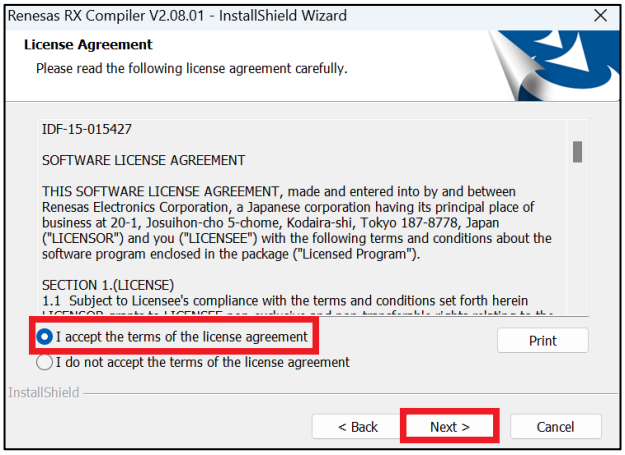


Figura 156. Términos del compilador CC-RX

A continuación, es necesario escoger la localización en la que instalar el compilador, se puede usar la opción que viene por defecto, y una vez elegido el destino, se hace clic sobre “Next”.

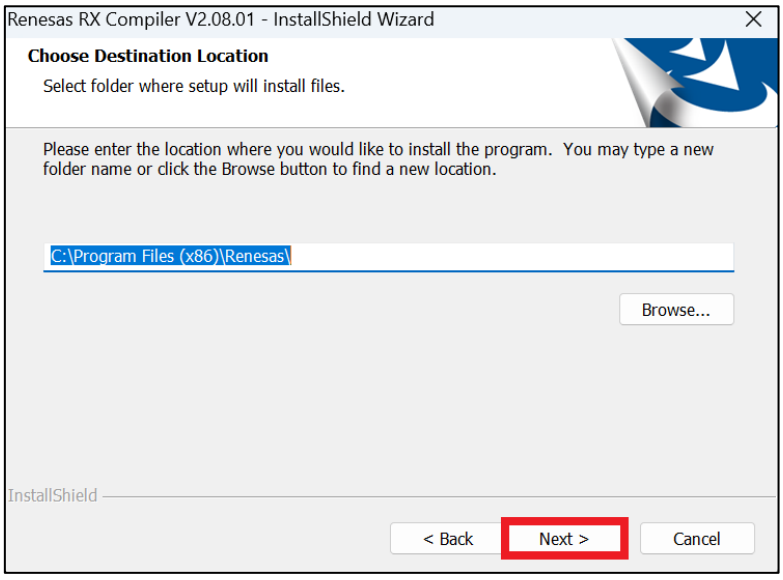


Figura 157. Localización para instalar el programa

Se pasará a la penúltima página para la instalación del compilador, que da la opción de modificar alguno de los pasos anteriores o continuar adelante. Si todo es correcto, se selecciona “Install”.

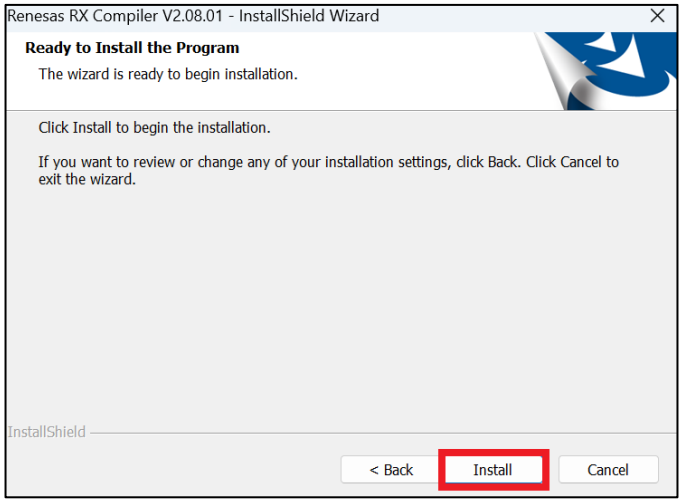


Figura 158. *Preparado para la instalación del compilador CC-RX*

Por último, se hace clic sobre “Finish” para instalar el compilador.

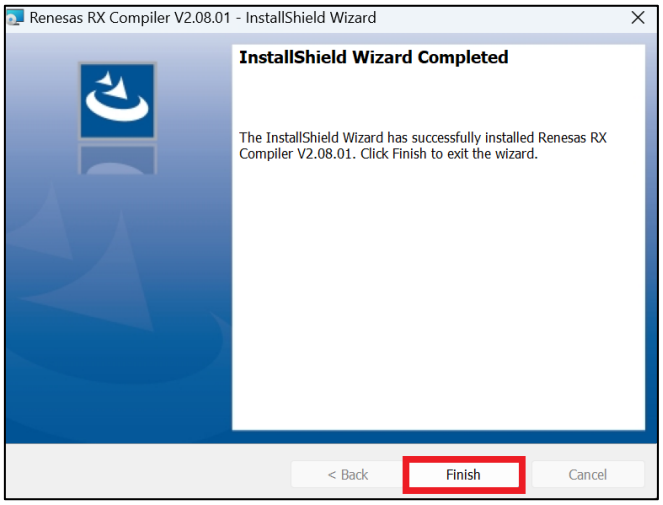


Figura 159. *Finalización de la instalación del compilador CC-RX*

Después de unos instantes aparecerá otra ventana, el asistente de instalación para administrar las licencias. Se pulsa el botón de “Next” para comenzar.

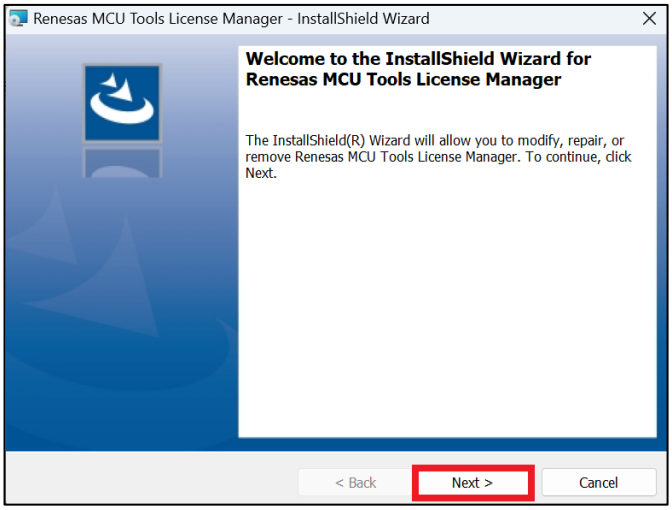


Figura 160. *Asistente de instalación para el administrador de licencia*

Se accede a la página para comprobar si los pasos anteriores son correctos y da la opción de modificarlos o continuar adelante. Si todo es correcto, se selecciona “Install”.

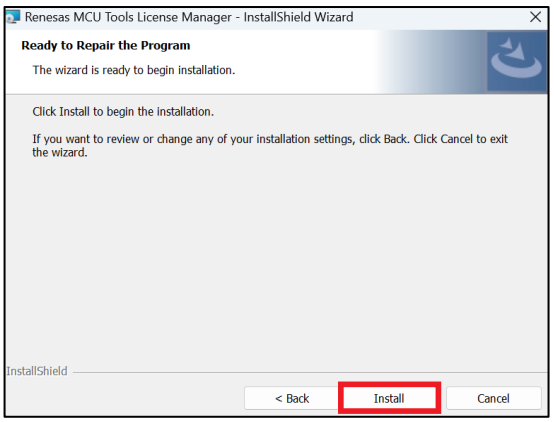


Figura 161. *Preparado para instalar la licencia*

Por último, se hace clic sobre “Finish” para instalar el compilador.

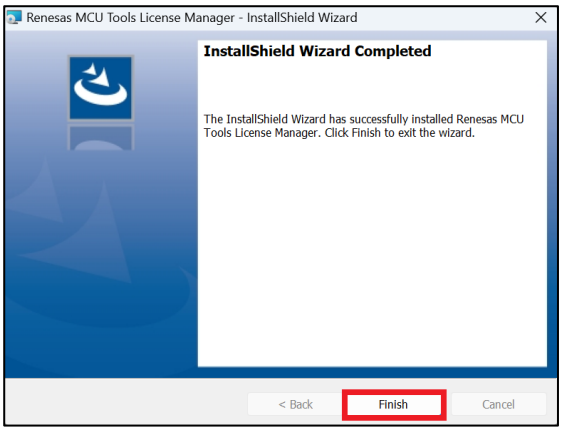


Figura 162. *Finalización de la instalación de la licencia*

A continuación, aparece la opción del idioma. Se selecciona el idioma deseado y se pulsa “OK”.

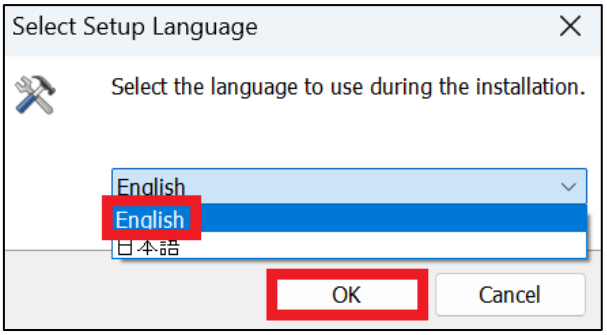


Figura 163. *Selección del idioma*

Por último, aparece la última ventana, para instalar GCC para Renesas RX. Este es un compilador de software libre para C, C++, Objective C y Fortran. Lo primero es seleccionar “Next” para pasar a la siguiente página.

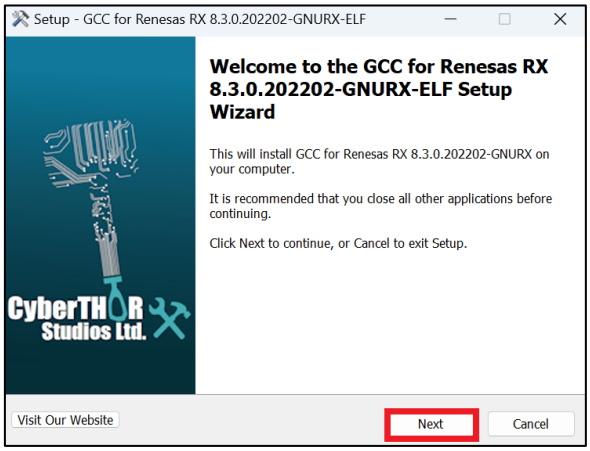


Figura 164. *Compilador GCC*

Se aceptan los términos para el compilador GCC y se hace clic sobre “Next”.

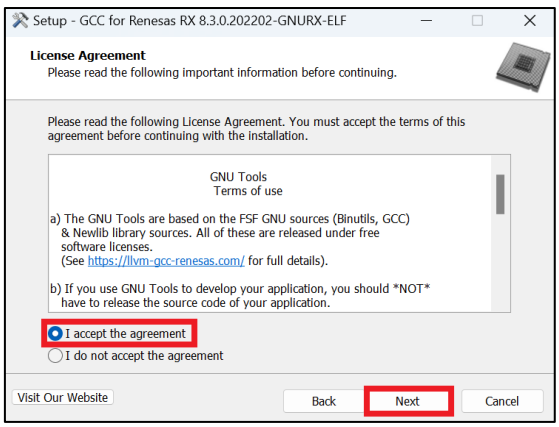


Figura 165. *Términos del compilador GCC*

Hay que esperar a que se instale el compilador, aparecerá una barra horizontal de carga.

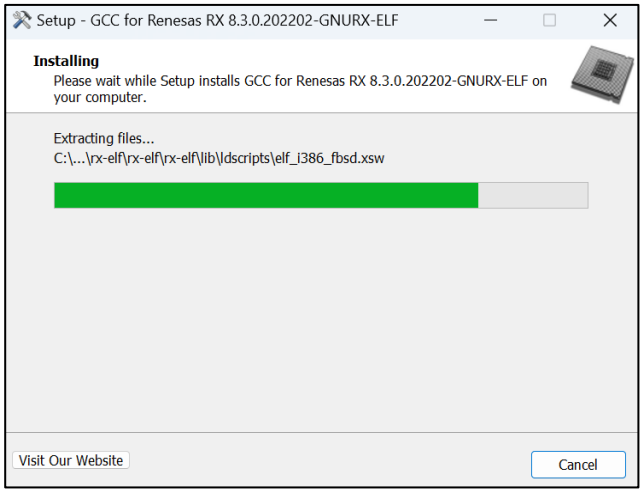


Figura 166. *Carga de instalación del compilador*

Una vez finalizada la carga de instalación, aparece en la ventana información importante para continuar con la descarga. Al terminar de leerlo, se pulsa “Next”.

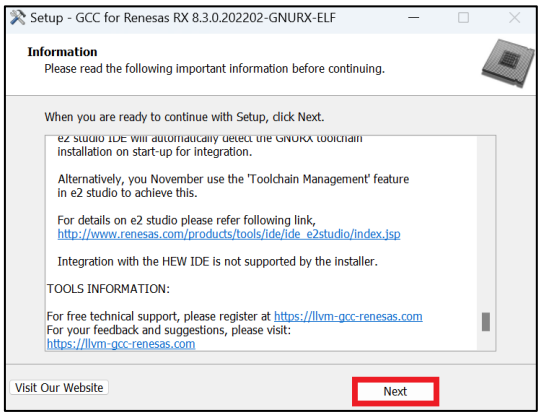


Figura 167. *Resumen de la instalación del compilador GCC*

Para finalizar, se hace clic sobre “Finish” para instalar el compilador.

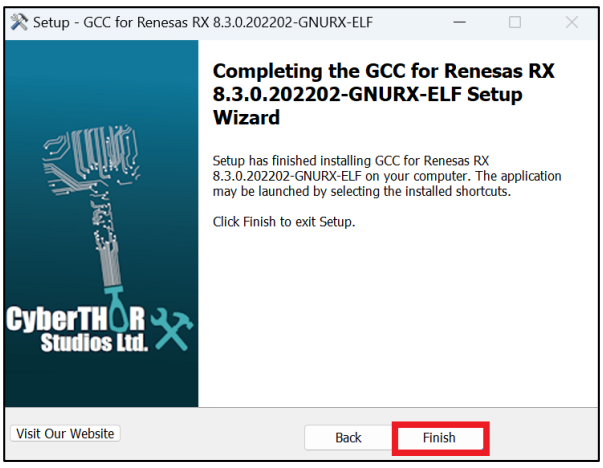


Figura 168. *Finalización de la instalación del compilador GCC*

En la ventana original, una vez termina la carga del entorno de desarrollo, aparecerán unas opciones, de las cuales, se deseleccionan todas. Se hace clic sobre “Ok”.

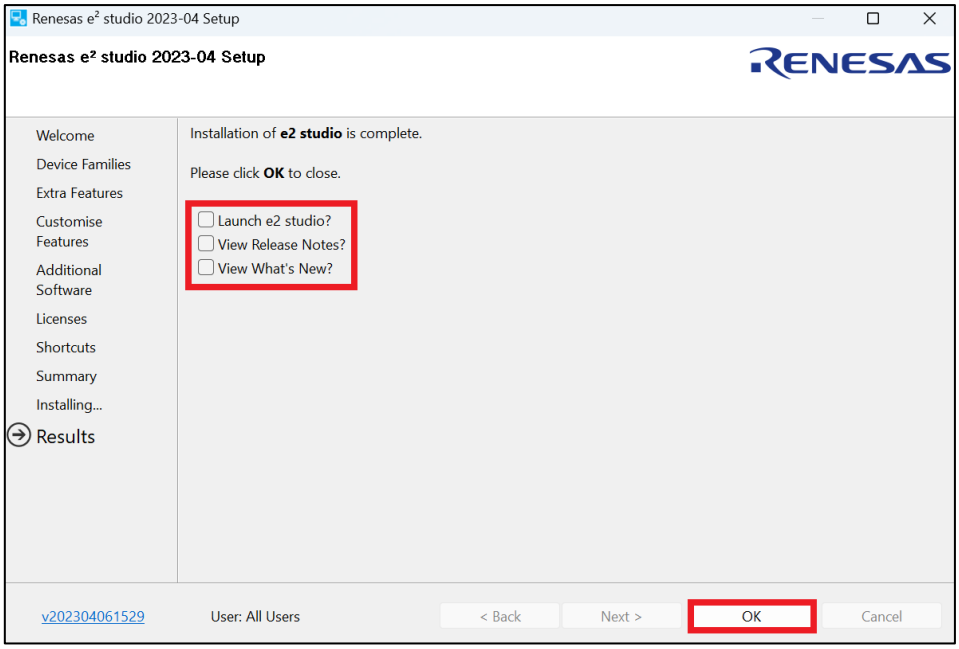


Figura 169. *Finalización de la instalación del programa*

Tras realizar todos los pasos anteriores, ya se puede acceder al entorno de desarrollo y comenzar a programar.

12.1.2. Guía rápida de instalación del entorno de desarrollo IconEdit

Para descargar el programa IconEdit, es necesario buscar en el navegador “IconEdit”, acceder a la página de RAMTEX, y hacer clic sobre el botón verde situado debajo de la imagen “Go To Download”.

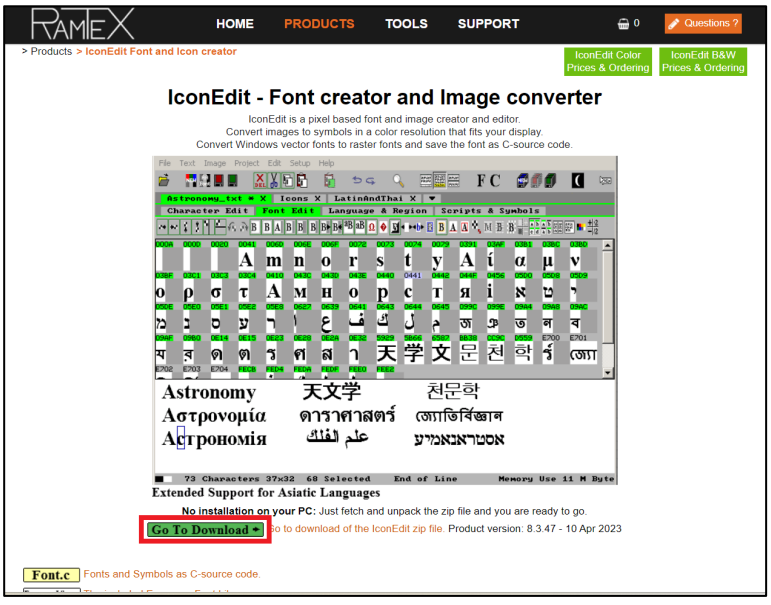


Figura 170. *Página de RAMTEX*

Se abrirá otra página en la misma ventana, y se hace clic sobre el botón rojo situado debajo de la imagen “Download”. A continuación, comenzará la descarga del programa en la computadora.

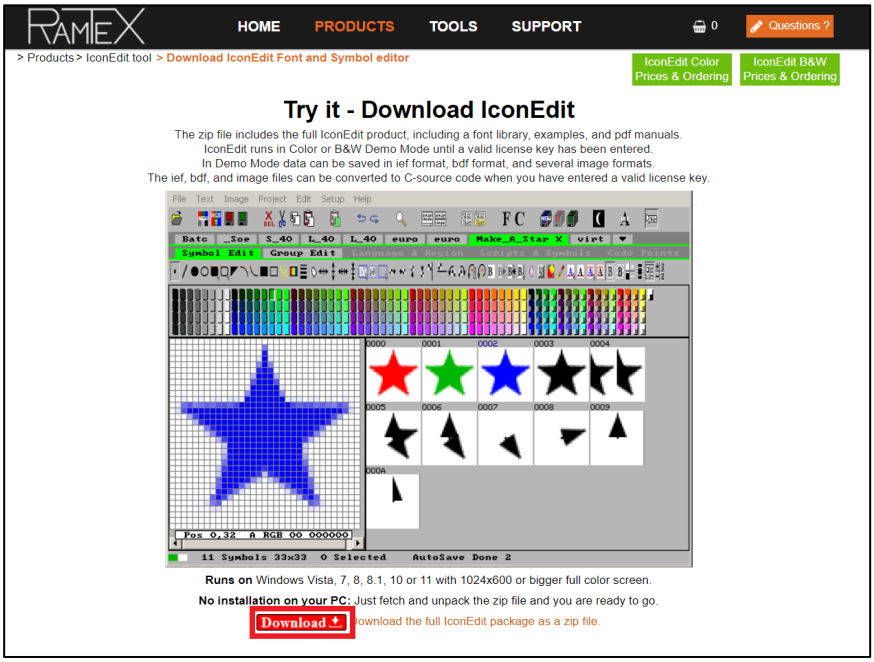


Figura 171. Descargar el programa

Una vez se ha descargado completamente el entorno de desarrollo, se accede a la carpeta “Descargas”, se pulsa con el botón derecho sobre la carpeta “iconedit.zip”, se selecciona “Extraer todo…” y se generará automáticamente una nueva carpeta llamada “iconedit”.

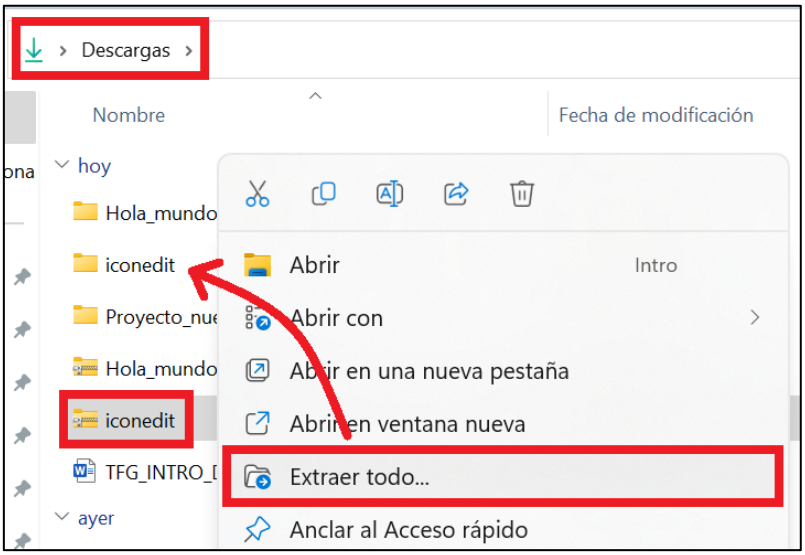


Figura 172. Extraer el programa

Una vez generada la carpeta, se hace doble clic sobre esta para ver su contenido. Dentro de la carpeta hay varios documentos PDF, que poseen guías de uso, manuales, tipos de fuente que ofrece el entorno de desarrollo y demás información útil para comenzar a usar el programa.

También hay carpetas con ejemplos, ejecutables y tipos de letra diferentes. Por último, hay un fichero tipo “exe” que es el programa, se llama “IconEdit.exe”.

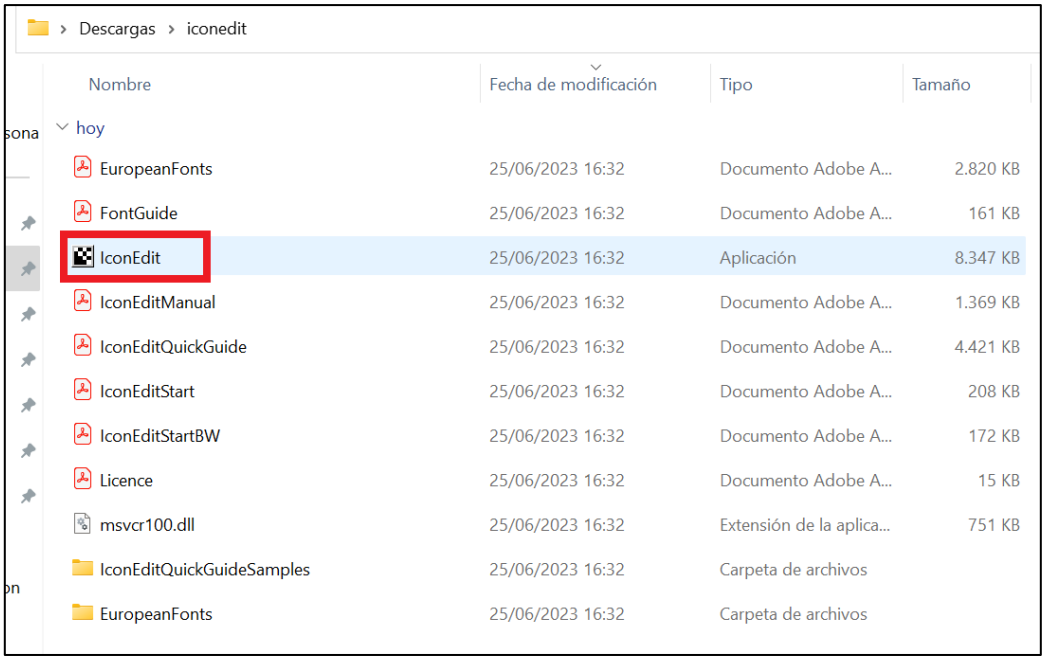


Figura 173. *Archivos que se ofrecen al descargar el programa.*

Se localiza el archivo ejecutable del programa, se hace doble clic sobre este y se iniciará la aplicación.

Al abrir IconEdit, se mostrará una ventana principal donde se requiere ingresar una clave de licencia para desbloquear todas las funciones y características del entorno de desarrollo. Una vez que se ingresa la clave de licencia, se debe hacer clic en el botón "Input License Key" para validarla.

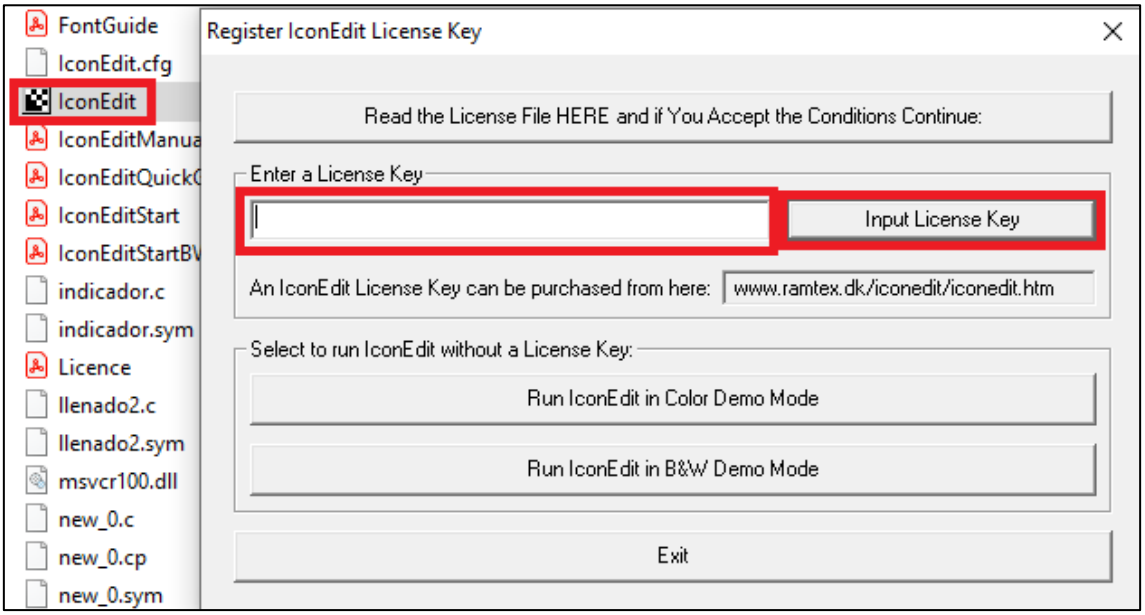


Figura 174. *Clave de licencia, IconEdit.*

Una vez ingresada la clave de licencia y confirmada, se puede proceder a crear un nuevo símbolo. Se deben especificar las dimensiones y el nombre del nuevo símbolo que se desea crear y luego hacer clic en el botón "Create New Color SYMBOL".

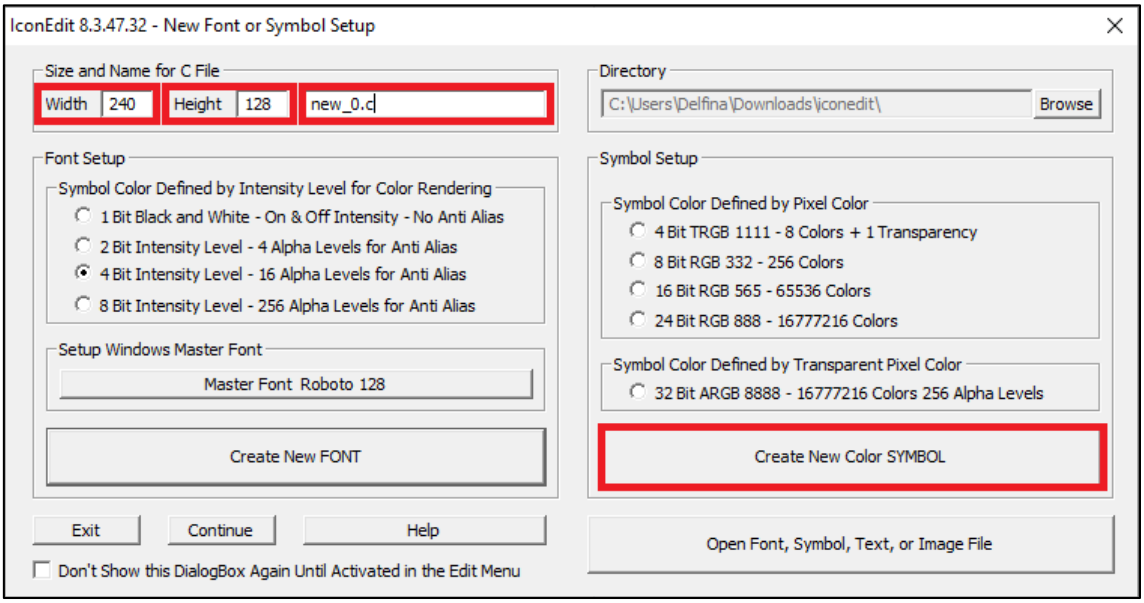


Figura 175. Configuración inicial para el símbolo.

Tras realizar todos los pasos anteriores, ya se puede acceder al programa y comenzar a diseñar símbolos e imágenes.

12.1.3. Guía para guardar las imágenes una vez creadas

Una vez se ha creado una imagen, es necesario guardarla de la siguiente manera. Se debe hacer clic sobre “File” y sobre “Modify Exsting Font, Symbol Group or Palette…”.

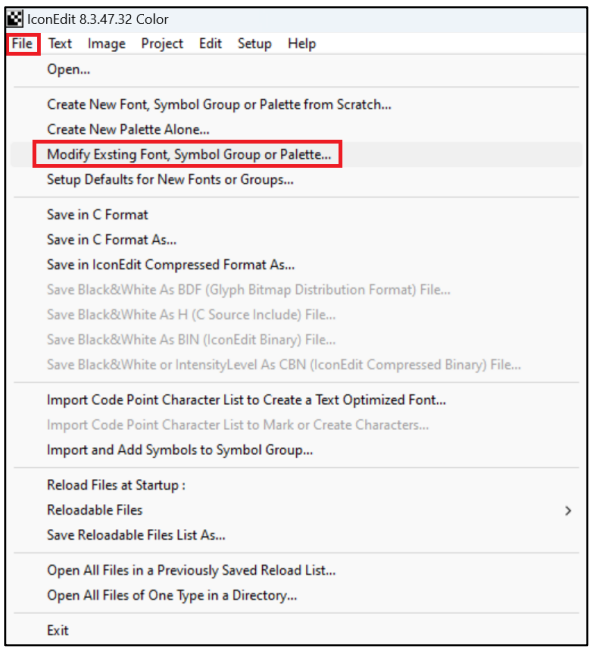


Figura 176. Selección a realizar para modificar el formato en el que guardar el símbolo.

A continuación, se abrirá la ventana de la figura 177. Es necesario seleccionar la opción de “4 Bit Grey Tone – 16 Grey Tones”, para que guarde el símbolo como una matriz, en la que cada byte representa dos píxeles.

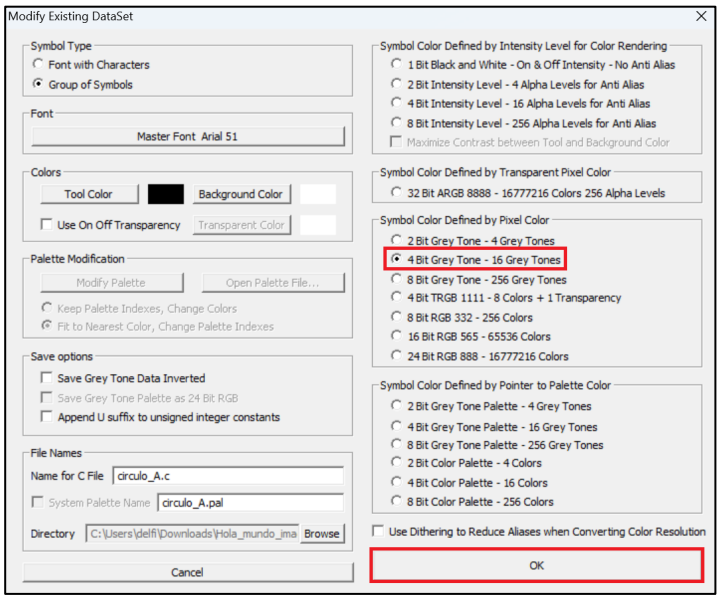


Figura 177. Opción para guardar el símbolo como una matriz que cada byte representa 2 píxeles.

Una vez guardado de esta forma, se debe de guardar el nuevo cambio en el archivo .c.

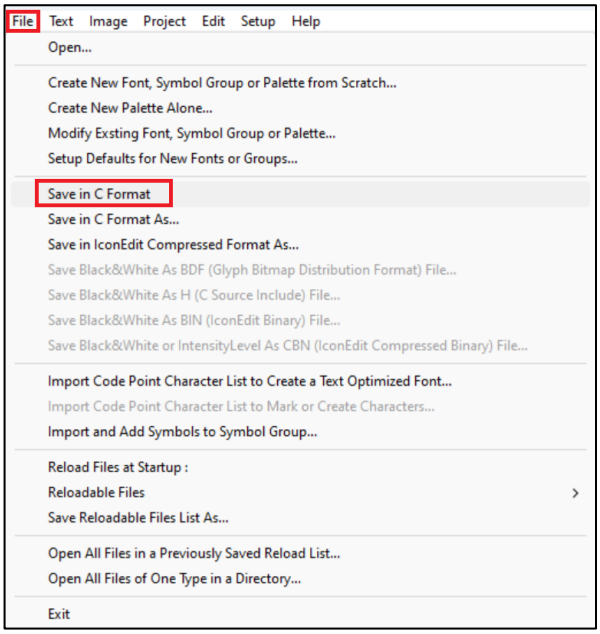


Figura 178. Selección a realizar para guardar el nuevo el formato del símbolo.

12.2. Controlador ST7529

Memory Map (2B3P, 8-bit mode)

			Column									
LCD read direction	CI = 0		0			1				84		
	CI = 1		84			83				0		
	Pixel		P0	P1	P2	P3	P4	P5		P252	P253	P254
	Data Line		$D7'_{1,0}$ $D6'_{1,0}$ $D5'_{1,0}$ $D4'_{1,0}$ $D3'_{1,0}$	$D2'_{1,0}$ $D1'_{1,0}$ $D0'_{1,0}$ $D7'_{2,0}$ $D6'_{2,0}$	$D4'_{2,0}$ $D3'_{2,0}$ $D2'_{2,0}$ $D1'_{2,0}$ $D0'_{2,0}$	$D7'_{1,1}$ $D6'_{1,1}$ $D5'_{1,1}$ $D4'_{1,1}$ $D3'_{1,1}$	$D2'_{1,1}$ $D1'_{1,1}$ $D0'_{1,1}$ $D7'_{2,1}$ $D6'_{2,1}$	$D4'_{2,1}$ $D3'_{2,1}$ $D2'_{2,1}$ $D1'_{2,1}$ $D0'_{2,1}$		$D7'_{1,84}$ $D6'_{1,84}$ $D5'_{1,84}$ $D4'_{1,84}$ $D3'_{1,84}$	$D2'_{1,84}$ $D1'_{1,84}$ $D0'_{1,84}$ $D7'_{2,84}$ $D6'_{2,84}$	$D4'_{2,84}$ $D3'_{2,84}$ $D2'_{2,84}$ $D1'_{2,84}$ $D0'_{2,84}$
Block	LI = 0	LI = 1										
0	0	159										
	1	158										
	2	157										
	3	156										
1	4	155										
	5	154										
	6	153										
	7	152										
2	8	151										
	9	150										
38	152	7										
	153	6										
	154	5										
	155	4										
39	156	3										
	157	2										
	158	1										
	159	0										
SEGout			0	1	2	3	4	5		252	253	254

Figura 179. *Visualización en escala de grises del mapa de memoria*

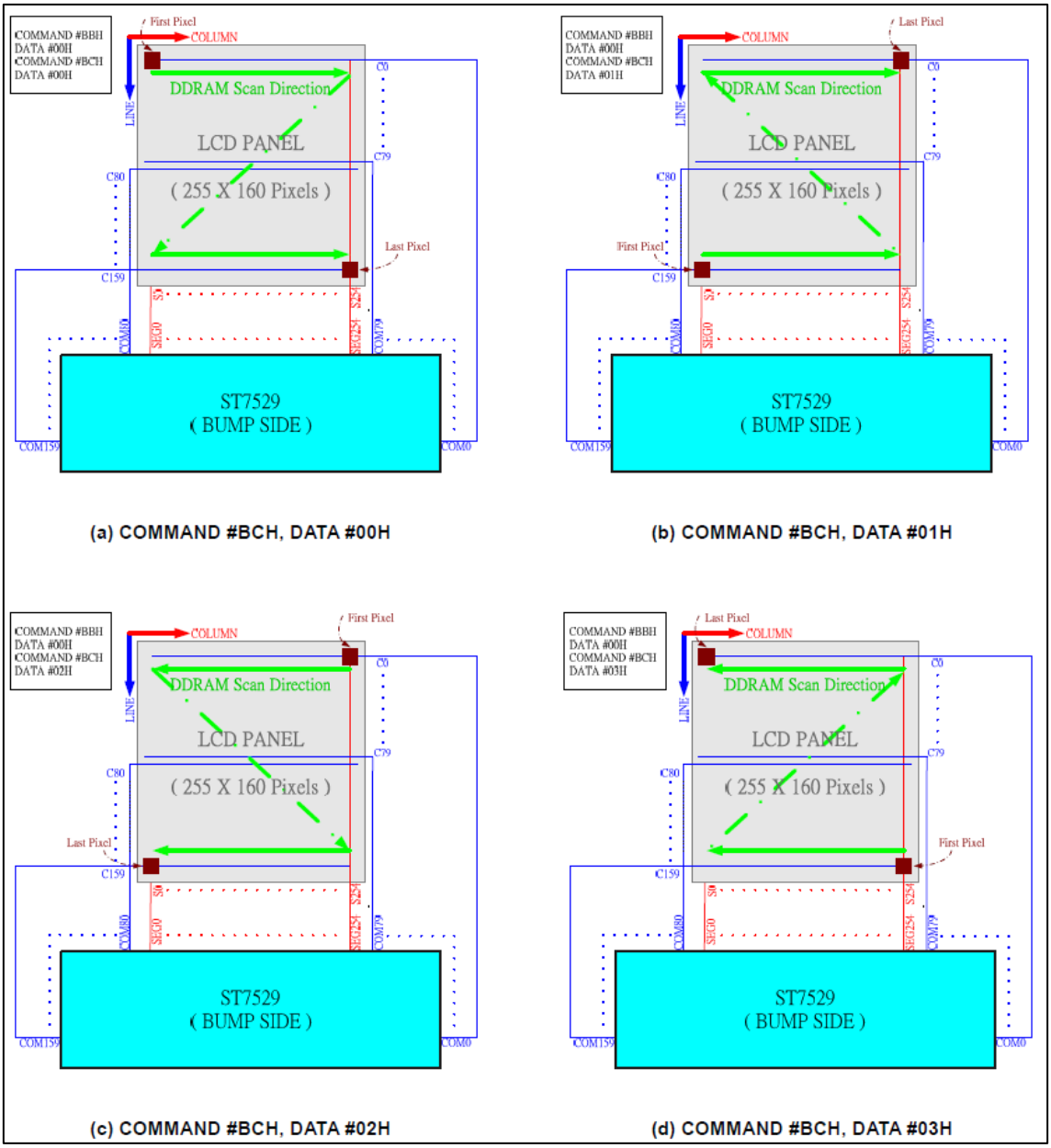


Figura 180. *Tipos de disposición de P1, P2 y P3, primero*

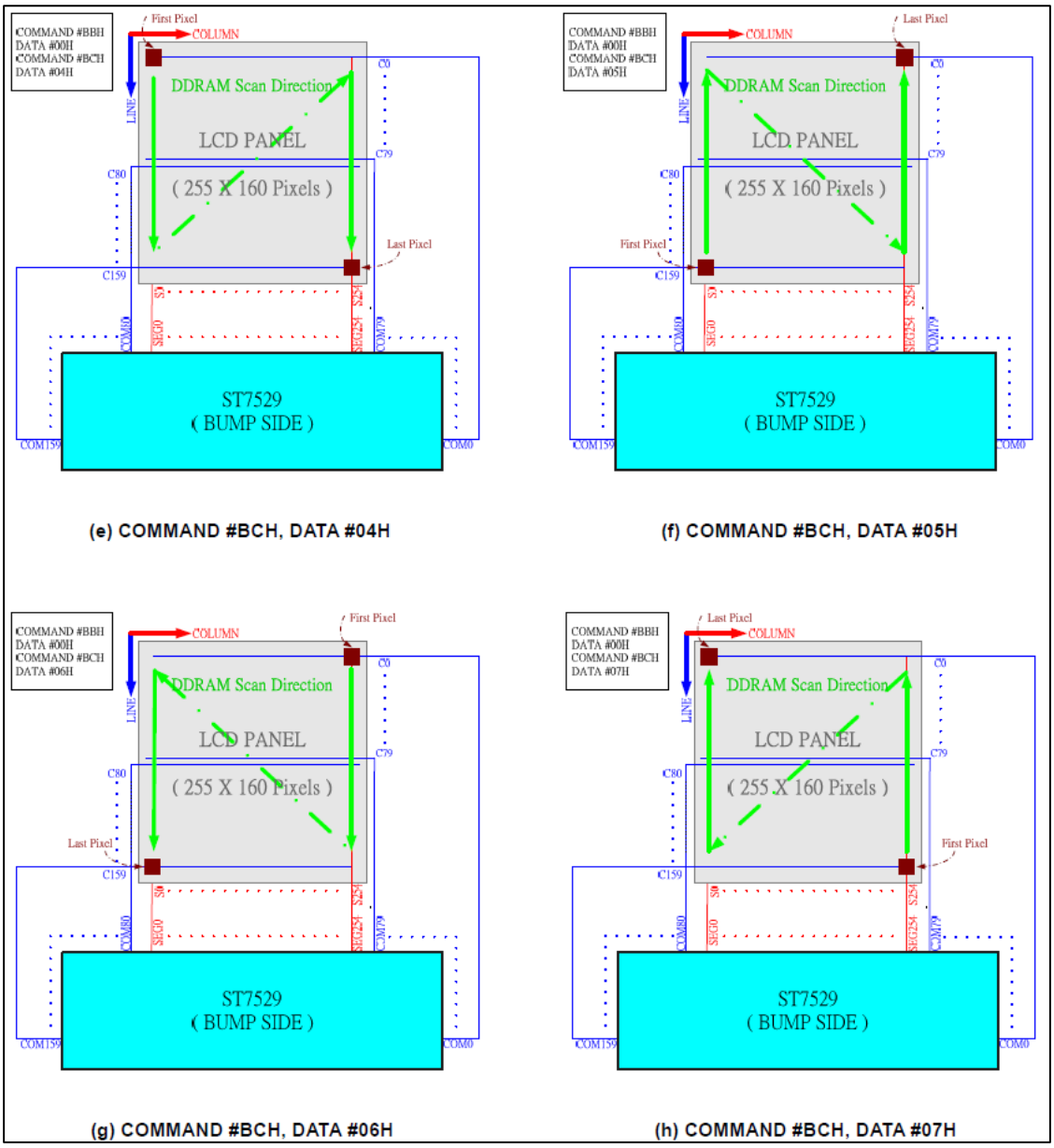


Figura 181. *Tipos de disposición de P1, P2 y P3, segundo*

12.3. Relación del trabajo con los objetivos de desarrollo sostenible de la agenda 2030

Objetivo de desarrollo sostenible	Alto	Medio	Bajo	No procede
ODS 1. Fin de la pobreza				X
ODS 2. Hambre cero				X
ODS 3. Salud y bienestar				X
ODS 4. Educación de calidad				X
ODS 5. Igualdad de género				X
ODS 6. Agua limpia y saneamiento				X
ODS 7. Energía asequible y no contaminante	X			
ODS 8. Trabajo decente y crecimiento económico				X
ODS 9. Industria, innovación e infraestructuras	X			
ODS 10. Reducción de las desigualdades				X
ODS 11. Ciudades y comunidades sostenibles		X		
ODS 12. Producción y consumo responsables		X		
ODS 13. Acción por el clima		X		
ODS 14. Vida submarina				X
ODS 15. Vida de ecosistemas terrestres				X
ODS 16. Paz, justicia e instituciones sólidas				X
ODS 17. Alianzas para lograr objetivos				X

Tabla 25. *Relación trabajo con objetivos de desarrollo sostenible de la agenda 2023*

Al relacionar los Objetivos de Desarrollo Sostenible (ODS) con el desarrollo de una interfaz gráfica para el control de grupos electrógenos, podríamos categorizar su impacto directo como sigue:

ODS 7. Energía asequible y no contaminante: Alto. El control eficiente de los grupos electrógenos puede contribuir a un uso más eficiente de la energía y reducir la contaminación.

ODS 9. Industria, innovación e infraestructuras: Alto. La interfaz gráfica para el control de grupos electrógenos es un claro ejemplo de innovación tecnológica aplicada a la industria.

ODS 11. Ciudades y comunidades sostenibles: Medio. Aunque indirectamente, un mejor control de los grupos electrógenos puede ayudar a reducir su impacto ambiental y contribuir a la sostenibilidad de las ciudades y comunidades.

ODS 12. Producción y consumo responsables: Medio. Una gestión más eficiente de los grupos electrógenos puede contribuir a un consumo de energía más responsable.

ODS 13. Acción por el clima: Medio. Mejorar la eficiencia y control de los grupos electrógenos puede contribuir a reducir las emisiones de gases de efecto invernadero.

Los demás ODS (1, 2, 3, 4, 5, 6, 8, 10, 14, 15, 16, 17) tienen un impacto directo bajo o nulo en este contexto, ya que la creación de una interfaz gráfica para grupos electrógenos no afecta directamente estos aspectos. Sin embargo, es importante tener en cuenta que los ODS están interconectados y progresos en unos pueden llevar a progresos en otros.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y desarrollo de librería de elementos gráficos para interfaz de usuario en sistemas empotrados para el control de grupos electrógenos

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Documento 2: Planos

AUTOR/A: Badiola Scarcella, Delfina Amaia

Tutor/a: Blanes Noguera, Juan Francisco

CURSO ACADÉMICO: 2022-2023

Tabla de contenidos

1. Planos

A continuación, se muestra los planos de las conexiones correspondientes entre la placa RX671 (en la parte de la izquierda de la figura 182) y Display de 240 x 128 píxeles (en la parte de la derecha).

Para comprender mejor cómo se realizan las conexiones vistas en la figura 182, se puede visualizar el hardware con todas las conexiones de la figura 2.

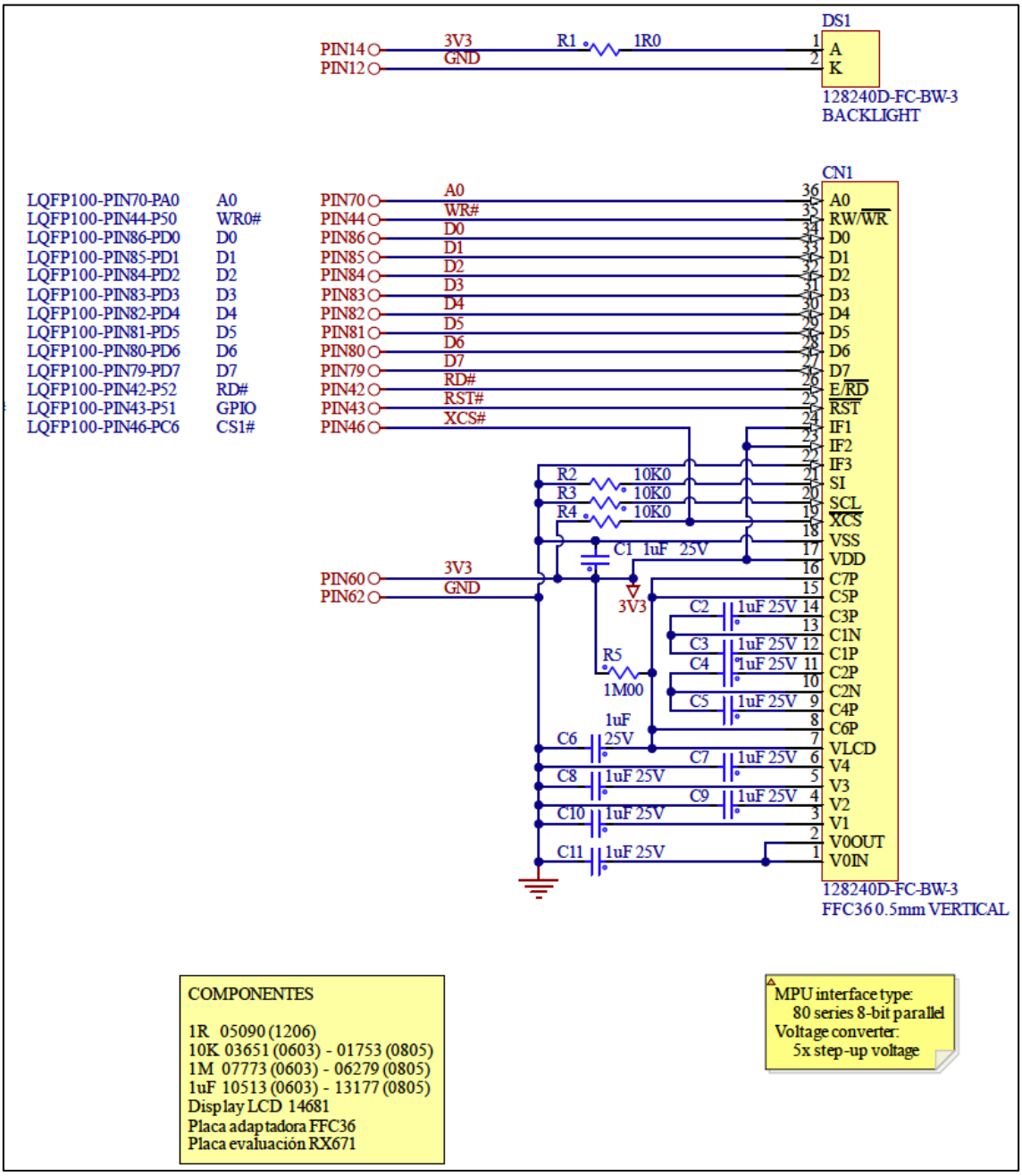
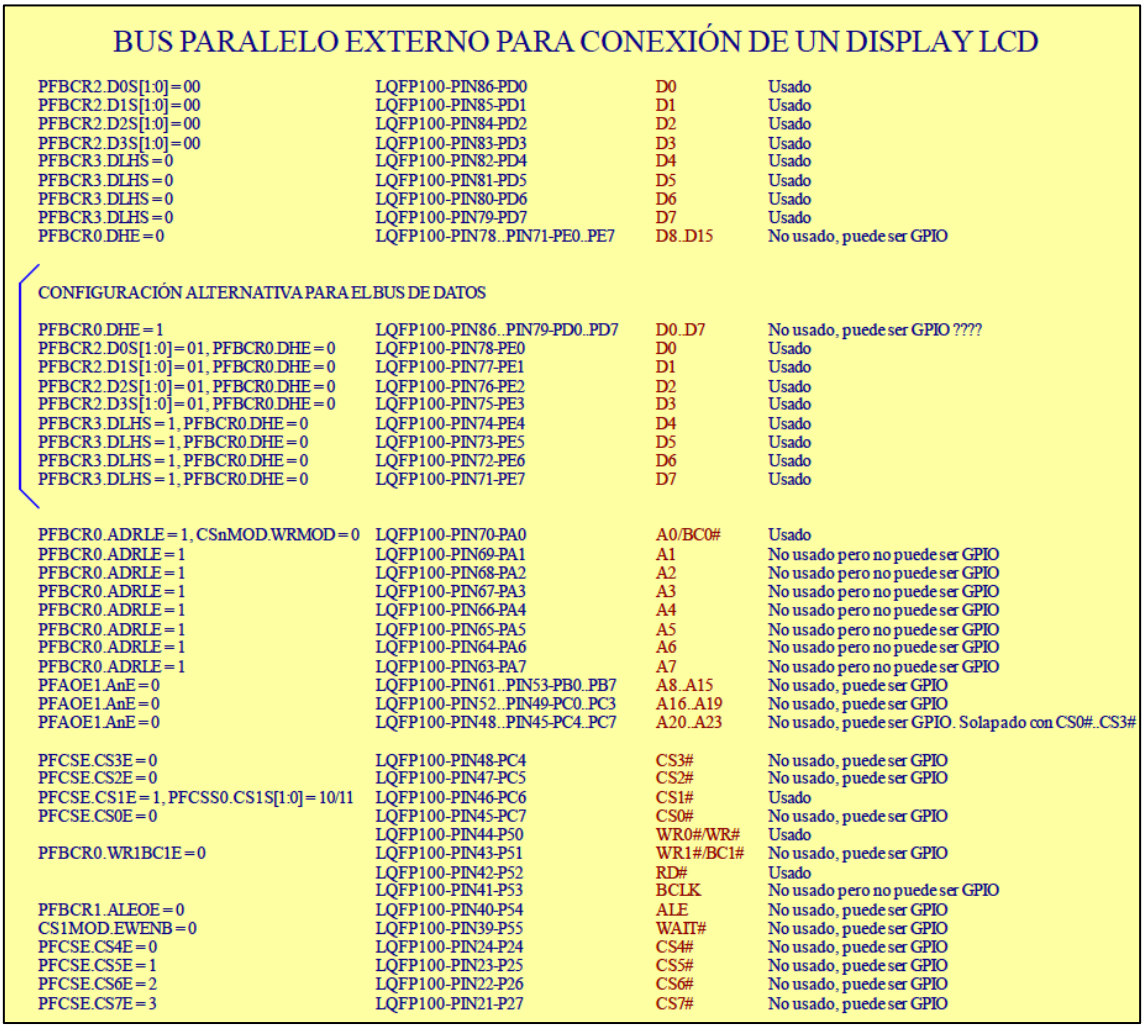


Figura 182. *Display 240 x 128*



BUS PARALELO EXTERNO PARA CONEXIÓN DE UN DISPLAY LCD

PFBCR2.D0S[1:0] = 00	LQFP100-PIN86-PD0	D0	Usado
PFBCR2.D1S[1:0] = 00	LQFP100-PIN85-PD1	D1	Usado
PFBCR2.D2S[1:0] = 00	LQFP100-PIN84-PD2	D2	Usado
PFBCR2.D3S[1:0] = 00	LQFP100-PIN83-PD3	D3	Usado
PFBCR3.DLHS = 0	LQFP100-PIN82-PD4	D4	Usado
PFBCR3.DLHS = 0	LQFP100-PIN81-PD5	D5	Usado
PFBCR3.DLHS = 0	LQFP100-PIN80-PD6	D6	Usado
PFBCR3.DLHS = 0	LQFP100-PIN79-PD7	D7	Usado
PFBCR0.DHE = 0	LQFP100-PIN78..PIN71-PE0..PE7	D8..D15	No usado, puede ser GPIO

CONFIGURACIÓN ALTERNATIVA PARA EL BUS DE DATOS

PFBCR0.DHE = 1	LQFP100-PIN86..PIN79-PD0..PD7	D0..D7	No usado, puede ser GPIO ????
PFBCR2.D0S[1:0] = 01, PFBCR0.DHE = 0	LQFP100-PIN78-PE0	D0	Usado
PFBCR2.D1S[1:0] = 01, PFBCR0.DHE = 0	LQFP100-PIN77-PE1	D1	Usado
PFBCR2.D2S[1:0] = 01, PFBCR0.DHE = 0	LQFP100-PIN76-PE2	D2	Usado
PFBCR2.D3S[1:0] = 01, PFBCR0.DHE = 0	LQFP100-PIN75-PE3	D3	Usado
PFBCR3.DLHS = 1, PFBCR0.DHE = 0	LQFP100-PIN74-PE4	D4	Usado
PFBCR3.DLHS = 1, PFBCR0.DHE = 0	LQFP100-PIN73-PE5	D5	Usado
PFBCR3.DLHS = 1, PFBCR0.DHE = 0	LQFP100-PIN72-PE6	D6	Usado
PFBCR3.DLHS = 1, PFBCR0.DHE = 0	LQFP100-PIN71-PE7	D7	Usado
PFBCR0.ADRLE = 1, CSnMOD.WRMOD = 0	LQFP100-PIN70-PA0	A0/BC0#	Usado
PFBCR0.ADRLE = 1	LQFP100-PIN69-PA1	A1	No usado pero no puede ser GPIO
PFBCR0.ADRLE = 1	LQFP100-PIN68-PA2	A2	No usado pero no puede ser GPIO
PFBCR0.ADRLE = 1	LQFP100-PIN67-PA3	A3	No usado pero no puede ser GPIO
PFBCR0.ADRLE = 1	LQFP100-PIN66-PA4	A4	No usado pero no puede ser GPIO
PFBCR0.ADRLE = 1	LQFP100-PIN65-PA5	A5	No usado pero no puede ser GPIO
PFBCR0.ADRLE = 1	LQFP100-PIN64-PA6	A6	No usado pero no puede ser GPIO
PFBCR0.ADRLE = 1	LQFP100-PIN63-PA7	A7	No usado pero no puede ser GPIO
PFAOE1.AnE = 0	LQFP100-PIN61..PIN53-PB0..PB7	A8..A15	No usado, puede ser GPIO
PFAOE1.AnE = 0	LQFP100-PIN52..PIN49-PC0..PC3	A16..A19	No usado, puede ser GPIO
PFAOE1.AnE = 0	LQFP100-PIN48..PIN45-PC4..PC7	A20..A23	No usado, puede ser GPIO. Solapado con CS0#..CS3#
PFCSE.CS3E = 0	LQFP100-PIN48-PC4	CS3#	No usado, puede ser GPIO
PFCSE.CS2E = 0	LQFP100-PIN47-PC5	CS2#	No usado, puede ser GPIO
PFCSE.CS1E = 1, PFCSS0.CS1S[1:0] = 10/11	LQFP100-PIN46-PC6	CS1#	Usado
PFCSE.CS0E = 0	LQFP100-PIN45-PC7	CS0#	No usado, puede ser GPIO
	LQFP100-PIN44-P50	WR0#/WR#	Usado
PFBCR0.WR1BC1E = 0	LQFP100-PIN43-P51	WR1#/BC1#	No usado, puede ser GPIO
	LQFP100-PIN42-P52	RD#	Usado
	LQFP100-PIN41-P53	BCLK	No usado pero no puede ser GPIO
PFBCR1.ALEOE = 0	LQFP100-PIN40-P54	ALE	No usado, puede ser GPIO
CS1MOD.EWENB = 0	LQFP100-PIN39-P55	WAIT#	No usado, puede ser GPIO
PFCSE.CS4E = 0	LQFP100-PIN24-P24	CS4#	No usado, puede ser GPIO
PFCSE.CS5E = 1	LQFP100-PIN23-P25	CS5#	No usado, puede ser GPIO
PFCSE.CS6E = 2	LQFP100-PIN22-P26	CS6#	No usado, puede ser GPIO
PFCSE.CS7E = 3	LQFP100-PIN21-P27	CS7#	No usado, puede ser GPIO

Figura 183. *Bus paralelo externo para conexión*

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y desarrollo de librería de elementos gráficos para interfaz de usuario en sistemas empotrados para el control de grupos electrógenos

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Documento 3: Pliego de condiciones

AUTOR/A: Badiola Scarcella, Delfina Amaia

Tutor/a: Blanes Noguera, Juan Francisco

CURSO ACADÉMICO: 2022-2023

Tabla de contenidos

1. Pliego de condiciones

Cabe destacar que la alumna no realizó el montaje de ningún dispositivo. La empresa facilitó el proyecto completamente montado.

Pero los componentes hardware utilizados han sido el microcontrolador RX671, el controlador de LCD (ST7529), la placa de desarrollo para RX671 (RTK5RX6710C00000BJ), un adaptador a LCD y el LCD (Displaytech 128240).

1.1. Microcontrolador RX671

El microcontrolador RX671, de 32 bits y basado en el núcleo de CPU RXv3, destaca por su frecuencia de operación máxima de 120 MHz, la compatibilidad con la unidad de protección de memoria (MPU) y con las interfaces de depuración JTAG y FINE. Este microcontrolador tiene una arquitectura de bajo consumo, operando con una fuente de alimentación única de 2.7 a 3.6 V, dispone de hasta 2 Mbytes de memoria flash interna para código, 8 Kbytes de memoria flash interna para datos, y 384 Kbytes de SRAM interna. Además, cuenta con un espacio de dirección externa que incluye buses de alta velocidad, 8 áreas de Chip Select (CS) y un área independiente de Memoria Dinámica de Acceso Aleatorio Síncrona (SDRAM) de 128 Mbytes. Incorpora una función de cifrado y dispone de hasta 114 pines para puertos de entrada/salida, con un rango operativo de temperatura que varía dependiendo de la versión.

1.2. Controlador LCD (ST7529)

El ST7529 es un controlador y driver LSI para sistemas de pantalla de cristal líquido (LCD) de matriz de puntos gráficos a escala de grises, capaz de conectarse directamente con un microprocesador y aceptar datos de visualización a través de interfaces como SPI y paralelo de 8 bits. Esta información se almacena en una memoria RAM interna del chip, lo que permite operaciones de lectura y escritura sin necesidad de un reloj operativo externo, optimizando así el consumo energético. Este chip incorpora también los circuitos de suministro de energía necesarios para impulsar el cristal líquido, lo que facilita la construcción de un sistema de visualización con un mínimo de componentes. Entre sus principales características, destacan su capacidad de salida de 255 segmentos/160 salidas comunes, el soporte para diferentes relaciones de deber para el display parcial, una RAM de datos de visualización en chip de hasta 204000 bits y un rango de tensión de funcionamiento de 2.4 a 3.3V. En cuanto a los modos de visualización, ofrece dos para la escala de grises de 32 bits: 2B3P y 3B3P, siendo el primero el seleccionado para el proyecto en cuestión.

1.3. Target board para Rx671 (RTK5RX6710C00000BJ)

La RTK5RX6710C00000BJ es una placa de destino diseñada para el microcontrolador RX671 de Renesas, siendo una herramienta fundamental para la evaluación, el prototipado y el desarrollo con este microcontrolador. Las especificaciones notables de la placa incluyen su MCU de evaluación con 2-MB ROM, 384-KB RAM, y 8-KB de memoria flash de datos, un tamaño compacto de 54.0 mm x 90.0 mm y un grosor de 1.6 mm, además de un circuito de suministro de energía versátil. Esta placa permite realizar tareas como la programación del MCU de Renesas, la depuración del código del usuario, y la implementación de circuitos de usuario para conmutadores y LEDs, proporcionando también ejemplos de aplicaciones y código de

inicialización de funciones periféricas. La conexión con el ordenador se realiza a través de un conector USB micro-B, que también sirve para el suministro de energía.

1.4. Adaptador a LCD

El adaptador a LCD o placa convertidora FPC/FFC, utilizada en aplicaciones electrónicas de alta densidad, es compatible con PCB de un solo lado y posee 36 pines con un paso de 0,5 mm a 2,54 mm. Su tamaño total es de aproximadamente 48 x 26 x 3,5 mm. Aunque las medidas se realizan manualmente, pudiendo variar en +/- 0,1 pulgadas, este dispositivo fabricado por Sourcing Map y con un peso de aproximadamente 10 gramos, cumple su función como convertidor, facilitando la conexión entre diferentes componentes electrónicos.

1.5. LCD (Displaytech 128240)

El Displaytech 128240 es una pantalla LCD de 240 x 128 puntos con un tamaño de punto de 0.325 x 0.325 mm y un área de visualización de 92.0 mm x 53.0 mm. El módulo tiene dimensiones de 98.7 mm x 67.7 mm x 9.5 mm. La tecnología FSTN, complementada con un polarizador transflectivo, se utiliza en esta pantalla y tiene un ángulo de visión de 6/12 en punto. Incluye retroiluminación LED blanca alimentada por una fuente externa. El controlador LCD es el ST7529-G, y el método de conducción es de 1/144 Duty y 1/12 Bias, referentes al ciclo de trabajo y punto de operación del dispositivo, respectivamente. Sin embargo, la pantalla se divide en dos partes, lo que puede generar problemas y requiere la configuración correcta para una visualización adecuada de la información.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y desarrollo de librería de elementos gráficos para interfaz de usuario en sistemas empotrados para el control de grupos electrógenos

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Documento 4: Presupuesto

AUTOR/A: Badiola Scarcella, Delfina Amaia

Tutor/a: Blanes Noguera, Juan Francisco

CURSO ACADÉMICO: 2022-2023

Tabla de contenidos

1. Presupuesto

En este apartado se desarrolla el presupuesto del proyecto, donde se muestra la estimación de los costes de ejecución del proyecto.

1.1. Precios de los componentes

Artículo	Unidad	Precio Unitario (€)	Unidades	Subtotal (€)
Microcontrolador RX671	u	25,00 €	1	25,00 €
Controlador LCD ST7529	u	5,00 €	1	5,00 €
Target board Rx671	u	30,00 €	1	30,00 €
Adaptador a LCD	u	2,00 €	1	2,00 €
LCD Displaytech 128240	u	50,00 €	1	50,00 €
Cables	u	0,50 €	15	7,50 €
			TOTAL (€)	**119,50 €**

Tabla 26. *Precios de los componentes*

1.2. Desarrollo y mano de obra

Artículo	Unidad	Precio Unitario (€)	Unidades	Subtotal (€)
E2 Studio	u	- €	1	- €
Librerías Contr. ST7529	u	25,00 €	1	25,00 €
Licencia IconEdit	u	805,00 €	1	805,00 €
Tipos de letra	u	5,00 €	70	75,00 €
Alumno prácticas	h	4,12 €	250	1.030,00 €
			TOTAL (€)	**1.935,00 €**

Tabla 27. *Desarrollo y mano de obra*

1.3. Presupuesto

Artículo	Unidad	Precio Unitario (€)	Unidades	Subtotal (€)
Coste Material	u	119,50 €	1	119,50 €
Desarrollo y Mano de obra	u	1.935,00 €	1	1.935,00 €
			TOTAL (€)	**2.054,50 €**

Tabla 28. *Presupuesto total*

El presupuesto general del proyecto asciende a la cantidad de: DOS MIL CINCUENTA Y CUATRO EUROS (2.054,50 €).

Todos los importes mostrados incluyen el IVA.