Este trabajo de fin de grado tiene como objetivo continuar con el desarrollo de una segunda versión de equipo, que catalogue según el módulo de elasticidad mediante una técnica no invasiva la calidad de la madera. Entendiendo como madera: árboles, trozas y tablas. De esta forma se busca organizar y caracterizar la madera, y en concreto la de chopo.

Para estimar el módulo de elasticidad, el tiempo de vuelo es determinante para conocer mediante una onda acústica la londitud interior del material. Para ello se usan transductores piezoeléctricos y la señal que llega es procesada con distintos algoritmos.

El presente proyecto contiene las etapas de análisis de un equipo realizado por un anterior alumno, e identificación de errores para así realizar una propuesta de mejora en cada aspecto que involucra.

Juan Manuel Martín Lucena es un recién graduado en Ingeniería electrónica industrial nacido en Granada. Con esta tesis acaba su grado en la Universidad de Granada. Se encuentra colegiado en el Colegio Oficial de Peritos e Ingenieros Técnicos Industriales de Granada (COGITI).

www.linkedin.com/in/juan-manuel-martin-lucena

Andrés María Roldán Aranda es el tutor académico del presente proyecto. Profesor en el Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada y creador del grupo de electronica aeroespacial GranaSAT.

TESIS DE GRADO

Desarrollo de equipo para la medición del módulo de elasticidad en madera

JUAN M. MARTÍN LUCENA

INGENIERÍA ELECTRÓNICA IND.

23/24

UNIVERSIDAD DE GRANADA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL

TRABAJO DE FIN DE GRADO

Desarrollo de equipo para la medición del módulo de elasticidad en madera

JUAN MANUEL MARTÍN LUCENA

2023/2024

Tutor: Andrés María Roldán Aranda

Impreso en Granada, septiembre 2024.

” Desarrollo de equipo para la medición del módulo de elasticidad en madera. ”

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL

” Desarrollo de equipo para la medición del módulo de elasticidad en madera. ”

AUTOR:

Juan Manuel Martin Lucena

TUTOR:

Prof. Andrés Roldán Aranda

Dpto:

Electrónica y Tencologías de Computación

Curso Académico:

2023/2024

Juan Manuel Martin Lucena, 2023/2024

Development of Equipment for Measuring the Modulus of Elasticity in Wood

Juan Manuel Martin Lucena

KEYWORDS:

TIK, LIFE Wood For Future, , GranaSAT, UIMA poplar, END, acoustic wave, Embedded System, RTOS, Firmware, PCB, CAD, EDA, Time of Flight.

ABSTRACT:

This Final Degree Project aims to continue the development of a portable electronic device capable of measuring the stiffness of wood in trees, logs, and boards using non-destructive testing (END) by analyzing acoustic waves. The Modulus of Elasticity is used as an indicator. This work focuses on the development and improvement of the hardware, software, and mechanics, leaving the process of signal acquisition and subsequent analysis to my colleague Antonio Morales, as described in his Final Degree Project in Telecommunications also completed during the 2023/2024 academic year.

This project is part of the LIFE Wood For Future project at the University of Granada (UGR), financially supported by the program of the European Union (EU). The goal is to revive poplar cultivation, which has significantly declined in Spain and specifically in the Vega of Granada. The project seeks to demonstrate the viability of using poplar wood in the production of structures for homes and fruit crates, among many other applications. Restoring poplar groves in Granada would economically benefit the city by adding an industry to the metropolitan area, thereby creating new local jobs. Additionally, it would fundamentally support the creation of a sustainable industry, improve biodiversity, and increase carbon dioxide absorption rates to reduce the pollution present in Granada, which is exacerbated by its poor orographic position. The work has been developed in collaboration with the aerospace electronics group GranaSAT and the research group UIMA.

The equipment being further developed, known as the Tree Inspection Kit (TIK), is an innovative portable tool that, through a non-invasive diagnostic technique, aims to measure the time of flight of an acoustic wave inside the material being tested. Additionally, by knowing its MoE, we can classify the wood piece. This is achieved using two piezoelectric probes embedded in the wood and separated by a known distance. The propagation speed translates into the modulus of elasticity, given that the material's density is known.

We have worked with equipment whose functional status was initially unknown. Through investigation and testing, we overcame and resolved identified errors, moving on to a new design phase to correct the errors found in the electronic and mechanical sections, and developing the software section. This process followed stages of requirements specification, design, and manufacturing of the printed circuit boards (PCB), as well as the continuation of the initially implemented real-time operating system (RTOS).

The objective of this Final Degree Project is to demonstrate the knowledge acquired during the engineering degree, encompassing the capabilities of analysis, design, and manufacturing of a product in its pre-commercialization phase. Therefore, knowledge of both mechanical and electronic design software that was not covered during the degree was required. Finally, the entire process is documented in this report.

” Desarrollo de equipo para la medición del módulo de elasticidad en madera. ”

Juan Manuel Martin Lucena

PALABRAS CLAVE:

TIK, LIFE Wood For Future, , GranaSAT, UIMA chopo, END, onda acústica, Sistema empotrado, RTOS, Firmware, PCB, CAD, EDA, Tiempo de vuelo.

RESUMEN:

El presente Trabajo de Fin de Grado busca continuar con el desarrollo de un equipo electónico portátil que sea capaz de medir la rigidez de la madera de árboles, trozas y tablas usando ensayos no destructivos (END) utilizando para ello el análisis de ondas acústicas. Como indicador se usa el Módulo de Elasticidad. El presente trabajo se centra en el desarrollo y mejora del hardwara, software y mecánica dejando el proceso de adquisición de señales y su posterior análisis al compañero Antonio Morales redactado en su Trabajo de Fin de Grado de Telecomunicaciones también realizado el curso 2023/2024

Este trabajo se enmarca en el proyecto LIFE Wood For Future de la Universidad de Granada UGR sustentado económicamente por el programa de la Unión Europea UE. El objetivo es recuperar el cultivo de chopo tan mermado en España y en concreto, en la Vega de Granada. Con ello se busca demostrar la viabilidad de la madera en la producción de estructuras para los hogares y de cajas para la fruta (entre otros muchos). Recuperar las choperas en Granada beneficiaría económicamente la ciudad añadiéndole una industria en el área metropolitana de la ciudad creando nuevos puestos de trabajo locales. Además, fundamentalmente se apoyaría la creación de una industria sostenible mejorando la biodiversidad y aumentando la tasa de absorción del dióxido de carbono para reducir la contaminación presente en la ciudad de Granada por su mala posición orográfica. El trabajo se ha desarrollado con la colaboración del grupo de electrónica aeroespacial GranaSAT, y el grupo de investigación UIMA.

Tree Inspection Kit (TIK) es una herramienta innovadora portátil que realizando el diagnóstico mediante una técnica no invasiva, busca medir el tiempo de vuelo de una onda acústica en el interior del material a ensayar. Además conociendo su MoE somos capaces de clasificar la pieza de madera. Para ello se utilizan dos sondas piezoeléctricas clavadas en la madera y separadas una distancia conocida. La velocidad de propagación deriva en el módulo de elasticidad conociendo previamente la densidad del material.

Se ha trabajado con un equipo que inicialmente desconocemos su estado de funcionalidad (proceso de ingeniería inversa). Investigando y ensayando sobre él se han superado y solucionado errores encontrados para así, pasar a una etapa de nuevo diseño remediando los errores encontrados en el apartado electrónico y mecánico y desarrollando además, el apartado de software. Para ello se han seguido unas etapas de especificación de requisitos, diseño y fabricación de las placas de circuitos impresos PCB, así como la continuación del sistema operativo implementado inicalmente RTOS.

Este proyecto final de grado tiene como objetivo demostrar los conocimientos adquiridos durante el grado de ingeniería realizado abarcando las capacidades de análisis, diseño y fabricación de un producto en su fase previa a la de comercialización. Por tanto, se han necesitado conocimientos en programas de diseño asistido; tanto mecánico como electrónico, que no habían sido dados durante el grado. Y por último, la documentación de todo el proceso en la presente memoria.

Acknowledgements

In the hustle and bustle of everyday life, we often forget those who accompany us throughout our journey. Without a doubt, this Final Degree Project would not have been possible without their support.

I wholeheartedly thank my parents, Carmen and Juan Manuel, for their daily efforts and unconditional love. To my brother David, for his support and the passion for knowledge that he has shared with me for as long as I can remember. And to Cristina, for the laughter and love she gives me every day, thank you.

To my friends from the Granada neighborhood, for the unforgettable memories we've shared. To all the friends from my degree, with whom we have endured this challenging journey, and with whom we have also shared great moments that we will carry with us for the rest of our lives. And to all the people who will appear along the way—this is just the beginning of it all.

To the colleagues from the GranaSAT laboratories, for all their knowledge and assistance, as well as the great debates during those magnificent breakfasts. And of course, I must thank Juan's father for those 10 a.m. pizzas, slowly cooked in the oven where we solder the PCBs. A special thanks to my "second project tutor,Ïrene Gil, for her effort and help. And last, but certainly not least, I want to thank my tutor, Andrés, for his dedication, energy, and characteristic ambition. All of us who have passed through his laboratory are left with that drive to grow both professionally and personally.

Thank you all very much.

Agradecimientos

En las vicisitudes del día a día, olvidamos a quienes nos acompañan en nuestra vida. Sin duda, este Trabajo de Fin de Grado no habría sido posible sin su apoyo.

Agradezco de todo corazón a mis padres, Carmen y Juan Manuel, por los esfuerzos diarios y el cariño incondicional. A mi hermano David, por su apoyo y la pasión por el conocimiento que, desde que tengo uso de razón me ha brindado. Y a Cristina, por las risas y el amor que me regala en el día a día, gracias.

A mis amigos del barrio de Granada, por las anécdotas inolvidables vividas. A todos los amigos de la carrera, con quienes he sufrido este duro camino y con quienes también hemos compartido grandes momentos que nos llevaremos para el resto de nuestras vidas. Y también, a todas las personas que aparecerán en el camino. Esto es solo el principio de todo.

A los compañeros de los laboratorios de GranaSAT, por todo su conocimiento y la ayuda, además de los grandes debates durante esos magníficos desayunos. Y cómo no, agradecer al padre de Juan por esas pizzas a las 10 de la mañana, cocinadas a fuego lento en el horno donde soldamos las PCBs. Un especial agradecimiento a mi "segunda tutora"de proyecto, Irene Gil, por su esfuerzo y ayuda. Y por último, pero no menos importante, agradecer a mi tutor Andrés, por su dedicación, energía y ambición que le caracteriza. Todos los compañeros que pasamos por su laboratorio nos quedamos impregnados de ese ímpetu por crecer en el ámbito profesional y personal.

Muchas gracias a todos.

Índice general

Índice de figuras

Índice de tablas

Glosario

Food and Agriculture Organization En mayo de 1963, la 16ª Asamblea Mundial de la Salud aprobó el establecimiento del Programa Conjunto FAO/OMS sobre Normas Alimentarias y adoptó los Estatutos de la Comisión del Codex Alimentarius..

La Comisión Internacional de Chopos y Otros Árboles de Rápido Crecimiento para la Sostenibilidad de las Personas y el Medio Organismo estatutario basado en un tratado bajo el marco de la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO)..

LIFE Wood For Future proyecto dentro del Programa LIFE que pretende ser un instrumento clave para la recuperación de las plantaciones de chopos, para frenar su sustitución por cultivos herbáceos intensivos.El objetivo último es dotar al sector forestal de herramientas que garanticen un suministro sostenible de madera local a la industria, que conduzca a la generación de madera certificada de alta calidad..

LTspice Es un software de simulación de circuitos desarrollado por Analog Devices. Permite realizar simulaciones de circuitos analógicos y digitales con énfasis en circuitos integrados..

Mode ROLL Modo de algunos osciloscopios que nos permite ver lentamente el cambio que se producen en las señales desadas. Para ello la división por década temporal se vuelve muy alto..

Programa LIFE es la herramienta financiera de la Comisión Europea que apoya los proyectos en los ámbitos del medio ambiente y del clima.Estos proyectos se denominan “proyectos LIFE”..

Siglas

ABS Polímero termoplástico.

ADC Analog-Digital converter.

AIC Criterio de Información de Akaike.

APB Advanced Peripheral Bus.

Arduino Microcontroller Platform for Prototyping.

BJT Bipolar Junction Transistor.

BNC Bayonet Neill-Concelman.

CAD Computer Aided Design.

CESE Comité Económico y Social Europeo.

COMPOP Chopo and composites.

CPU Core Processor Unit.

EDA Electronic Design Automation.

END Ensayos no destructivos.

ESD Electrostatic Discharge.

EU Union Europea.

FFT Fast Fourier Transform.

Firmware Software orientado a un sistema embebido.

FRP Fiber Reinforced Polymer.

GCT Global Connector Technology.

GPIB General Purpose Interface Bus.

GPIO General Purpose Input/Output.

GranaSAT Grupo Electrónica Aerospacial.

I2C Inter-Integrated Circuit.

IDE Integrated Development Environment.

ISR Interrupt service routine.

IVA Impuesto Valor Añadido.

LAN Local Area Network.

LDO Low Drop-Out regulator.

LED Light Emitting Diode.

Li-Ion Litio-Ion.

LiPo Lithium polymer.

MoE Módulo de Elasticidad.

MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor.

NiCd Níquel-Cadmio.

NiMH Nickel-Metal Hydride.

NTC Negative Temperature Coefficient.

NXP Next eXPerience Semiconductors.

Pb Plomo.

PCB Printed Circuit Board.

PLA Ácido Poliáctico.

PVPC Precio Voluntario para el Pequeño Consumidor.

PWM Pulse Width Modulation.

RAM Random-Access Memory.

RMS Root Mean Square.

ROM Read-Only Memory.

RTC Real Time Clock.

RTOS Real-Time Operating System.

SCPI Standard Commands for programmable Instruments.

SMD Surface-Mount Device.

SoC System on Chip.

SPI Serial Peripheral Interface.

THT Through-Hole Technology.

TIK Tree Inspection Kit.

ToF Time of Flight.

UART Universal Asynchronous Receiver-Transmitter.

UE Unión Europea.

UGR Universidad de Granada.

UIMA Unidad de Investigación de Madera Estructural de Andalucía.

USB Universal Serial Bus.

Capítulo 1

Introducción

En el presente documento se detalla el Trabajo de Fin de Grado ralizado como resultado del conocimiento aprendido durante el grado de Ingeniería Electrónica Industrial, y en especial, se muestra lo aprendido durante la realización de este proyecto. Se detallarán todos los pasos seguidos, así como vicisitudes y logros obtenidos al retomar un proyecto anterior buscando solucionar problemas del actual modelo y avanzar en el mismo para aportar mayor funcionalidad. En general, el objetivo es mejorar el trabajo del anterior alumno Juan del Pino Mena, así como hacerlo completamente funcional. El equipo desde el que se parte es llamado TIK, creado para medir el MoE de árboles, tablas de madera y trozas, para así dotar al agricultor de una herramienta capaz de proporcionar el estado de esta materia prima.

Este proyecto surge con el proyecto LIFE Wood For Future [19] dentro del programa Programa LIFE [20] y se enmarca dentro de la acción C4: «Desarrollo de herramienta TIK para evaluar la calidad de la madera» del proyecto europeo LIFE Wood For Future (Referencia: LIFE20 CCM/ES/001656 [21]) «Recuperación de las choperas de Granada-Vega para potenciar la biodiversidad y la captura de carbono a largo plazo mediante bioproductos estructurales».

La segunda versión del proyecto TIK ha sido realizado gracias a la colaboración de dos grupos de investigación de la UGR: GranaSAT y UIMA.

(a) *Logo GranaSAT* **(b)** *Logo UIMA*

Figura 1.1 – *Logos de los departamentos colaboradores en este trabajo de fin de grado*

1.1. Motivación

Granada se encuentra entre las zonas más contaminadas de España [22]. Uno de los factores que contribuyen a esta situación es la posición geográfica del área metropolitana que al estar rodeada de sierras, incluida la famosa Sierra Nevada, dificulta la ventilación natural del aire lo que provoca un estancamiento de la contaminación. Este

(a) *Logo Programa LIFE* **(b)** *Logo LIFE Wood For Future*

Figura 1.2 – *Logos oficiales del programa y uno de sus proyectos: LIFE Wood For Future y Programa LIFE, respectivamente.*

problema se ve agravado por la alta concentración de actividad humana en la zona [23]. Además, en las últimas décadas, la silvicultura del chopo ha sido altamente devaluada, lo que ha llevado a la pérdida de los importantes beneficios ambientales que estos árboles proporcionan. Los chopos actúan como depuradores de agua conocidos como filtros naturales y también funcionan como grandes sumideros de carbono, sumado a que es un árbol de crecimiento rápido (de 10 a 15 años), [24] lo hace el candidato ideal para repoblar las 4500 hectáreas perdidas en los últimos 20 años, consiguiendo recuperar «el auténtico pulmón verde de Granada» [25].

En el plano económico, el desarrollo de un instrumento que mida la calidad estructural de la madera da una oportunidad a la Vega de Granada de iniciar una industria local. Con esto conseguimos incentivar la creación de un ecosistema empresarial que procese la materia prima autóctona y evitar costes de envío. Para que surja todo lo mencionado es necesario que la madera tenga la calidad suficiente y cumpla las características de: densidad, rigidez y resistencia mecánica.

(a) *Contaminación en el área metropolitana de Granada* **(b)** *Población de Chopos en la Vega de Granada*

Figura 1.3 – *Paisajes de Granada.*

Las directivas 210/30 [26] y 2012/27 [27] posicionan la madera como material clave para la eficiencia energética con mayor relevancia en el mediterráneo, donde estas propiedades de eficiencia se acentúan cuando llega el invierno. Por otro lado, vemos como la Unión Europea (concretamente el CESE) indica que la madera como material de construcción representa una gran oportunidad por su sostenibilidad y rentabilidad. Además permite mayor rapidez en construcción de edificios ayudando también a impulsar la economía de las zonas rurales. Como principal beneficio del uso de madera en la construcción, tiene la reducción del CO2 además de proporcionar un comportamiento antisísmico [28].

El área de la Ingeniería Electrónica soluciona problemáticas como esta mediante el desarrollo de sistemas embebidos que requieren firmware y hardware específicos. Además, se llevará a cabo un diseño mecánico del producto para garantizar que sea cómodo y se ajuste adecuadamente a la fisionomía humana. En cuanto al manejo de señales, se requieren conocimientos en adquisición, análisis y procesado de señal propio de la Ingeniería de Telecomunicaciones. Esta última parte ha sido trabajada por el compañero de laboratorio Antonio Morales, haciendo uso de competencias mayormente adquiridas en el Grado de Telecomunicaciones. Por lo tanto, para comprender el

proyecto en su totalidad se deben de revisar tanto el presente documento (diseño de hardware, Firmware y mecánico) como el suyo (adquisición de señales mediante los ADC del ESP32 y representación), ambos trabajos de fin de grado.

1

1.2. Estado actual del chopo

En esta parte se introducen las características del chopo, así como la población actual en distintos lugares del mundo para comprender la relevancia del desarrollo de esta industria de forma local.

1.2.1. El chopo

Los chopos, también conocidos como álamos, son árboles robustos con dos géneros: Salix (Sauces) y Populus (chopos o álamos), originarios de zonas templadas y subtropicales. Las zonas templadas tienen veranos cálidos e inviernos frescos y fríos, mientras que las zonas subtropicales tienen veranos largos y calurosos e inviernos suaves y frescos. España queda dividida entre zonas templadas en el norte y centro, mientras que en la costa mediterránea tenemos zonas subtropicales como las Canarias y la Costa mediterránea. Los chopos se caracterizan por su alta necesidad de agua y sol para su rápido crecimiento generalmente, en hábitats húmedos, llanuras inundables y fluviales riveras. [29]

Las cualidades del chopo son las siguientes:

- **Tallaje**: Alcanzan de 10 a 30 metros dependiendo de la especie, con porte cilíndrico.
- **Corteza**: Corteza inicalmente lisa y gris verdosa que con la edad se agrieta y se oscurece.
- **Rama y hoja**: Hoja simple caediza y ancha, con bodes enteros aserrados. El peciolo es largo y comprimido lateralmente limitando el movimiento de la hoja.

Los álamos y sáuces son fáciles de multiplicar con estaquilla y esquejes facilitando así la labor de los cultivadores. La clonación permite una rápida reproducción del chopo con la contrapartida de que, si se clona de un chopo susceptible a una determinada plaga o enfermedad podemos perder toda esa población. Por ello se ha conservado en los últimos años poblaciones naturales de álamos y sauces como reserva de la variación genética conseguiendo responder así ante futuras necesidades. Los clones permitidos se ubican en el BOE nº179 de 27 de julio de 1992 [30], duplicados en el BOE nº 63 de 14 de marzo de 2003 [31].

1.2.2. Futuro sostenible mediante el chopo.

El chopo es demandado por agricultores y la industria de la madera en el mundo entero por, como hemos mencionado por su rápido crecimiento y ciclo corto, ya que pueden ser cosechados a partir de los 9 años permitiendo una rápida rotación entre chopos cosechados y los que se recosechan. Este es el principal motivo por el cual los agricultores e industria optan por esta madera ya que el beneficio económico es mayor. La madera de chopo es versátil y utilizada en amplia gama de productos desde la pulpa de papel hasta el contrachapado, pasando con especial mención por la madera en la ingeniería. [32]

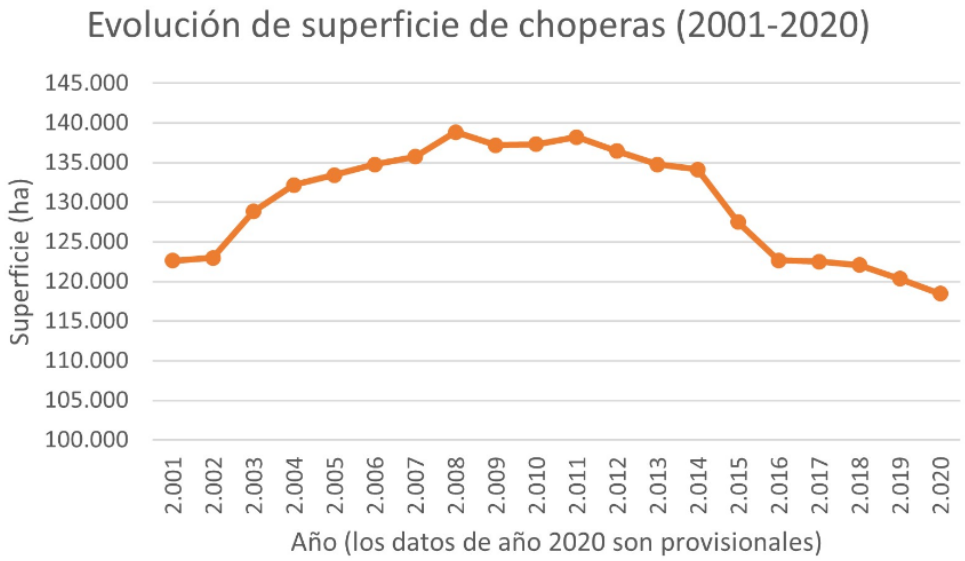
El cultivo de chopo es respaldado por la Food and Agriculture Organization [33] estableciendo la La Comisión Internacional de Chopos y Otros Árboles de Rápido Crecimiento para la Sostenibilidad de las Personas y el Medio Ambiente [34] su promoción como forma de explotación forestal sostenible combatiendo con la desirtificación. Desde esta comisión se respalda su rentabilidad económica además de ayudar a las tierras degradadas y contribuir a la biodiversidad.

Desde el ámbito de la investigacción se debe de destacar el papel de COMPOP [35] desde donde se evalúan mecánicamente los derivados de la madera de chopo y materiales reforzadores como la fibra FRP. El proyecto se desarrolla en la UGR en la facultad de ingeniería de Edificación con numerosos apoyos del sector público y privado.

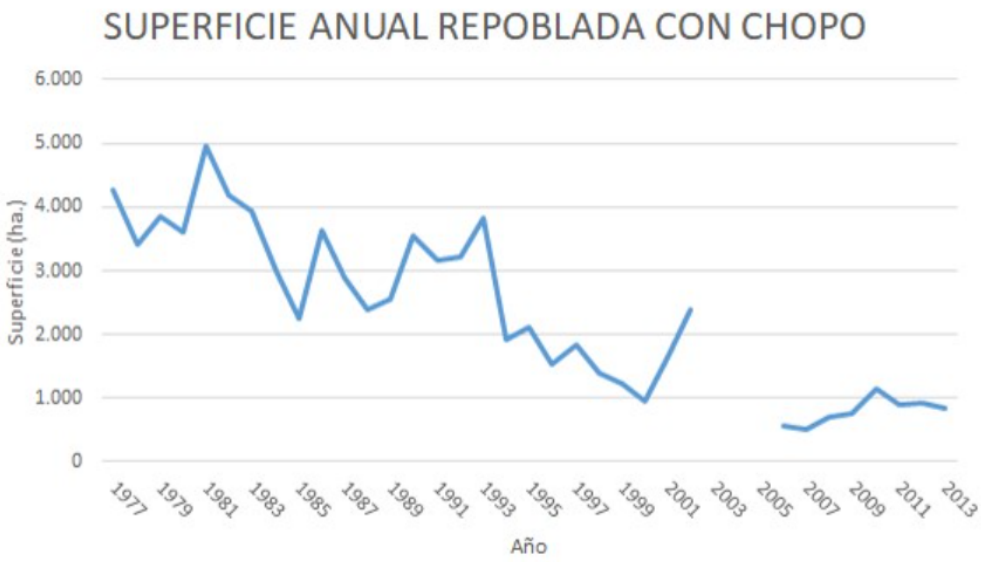
1.2.3. Población del chopo: España y Euroasia.

En España con los datos del Ministerio para la Transición Ecológica y el Reto Demográfico, acerca de la superficie, cortas, producción de planta, superficie repoblada y comercio exterior, podemos afirmar que el descuido por la población del chopo es evidente.

En la1.4 vemos como la superficie del chopo ha disminuido en los últimos 20 años. Además la repoblación ha tenido una tendencia claramente bajista desde los años 70.



(a) *Variación de la superficie de choperas a nivel nacional. FUENTE:ESYRCE/Anuario Estadística Agraria. [1]*



(b) *Superficie anual repoblada con chopo, a nivel nacional (años 2003, 2004 y 2005 ausentes por falta de datos). FUENTE:Anuario Estadística Agraria. [1]*

Figura 1.4 – *Variación de la superficie así como la repoblación.*

1

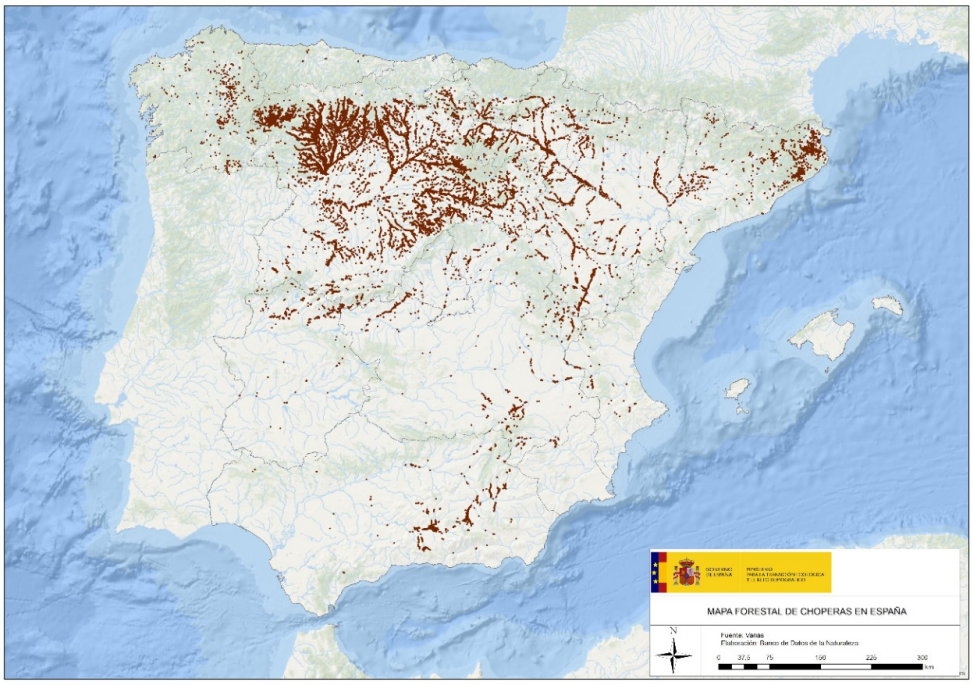


Figura 1.5 – *Mapa forestal de plantaciones de Chopo 2020 [1]*

1.3. Propagación de las ondas acústicas en la madera

El dispositivo a desarrollar, utiliza una de las técnicas no invasivas denominada caracterización de materiales, la cual nos permite hallar fenómenos fisicos y químicos del material. En especial, nosotros estamos interesados en el MoE (también conocido como módulo de Young). Es una medida de rigidez del material y por tanto una propiedad mecánica, la cual nos permite permite clasificar la madera en función de este módulo para uno u otro uso.

Las ondas usadas son acústicas, eso quiere decir que la frecuencia de la propagación de la misma no supera los 20kHz (de 20Hz a 20kHz) y se enmarca en el límite de la audición humana. Por otro lado, tenemos las ondas ultrasónicas cuya frecuencia supera al límite audible por el humano y son capaces de propagarse en mayor longitud que las anteriores en materiales uniformes o de baja viscosidad. Nuestro equipo se encargará de recoger estas ondas generadas con un golpe externo por lo que debe de tener dos terminales, uno que percibe la onda enviada al inicio y otro que la recibe al final de su recorrido por el material a medir.

Las ondas tienen distintas formas de propagación y se basan en la dirección de la onda y en el desplazamiento de las moléculas en el interior de la pieza de ensayo. Estas son:

- **Ondas longitudinales**: El desplazamiento de las moléculas es paralelo a la direción del frente de onda y son las que buscamos en nuestro caso.
- **Ondas transversales**: En este caso las partículas se propagan perpendicularmente a la dirección de onda con una baja velocidad y longitud de onda más corta a la misma frecuencia que las ondas longitudinales.
- **Ondas superficiales**: También conocidas como ondas de Reyleigh, presentan una oscilación a través de la superficie de la pieza de ensayo y son comúnmente usadas para detectar grietas superficiales.

Para conocer más acerca de somo medir el módulo de elasticidad MoE con las ondas longitudinales, consultar el capítulo 3 de la adquisisción de señales [6].

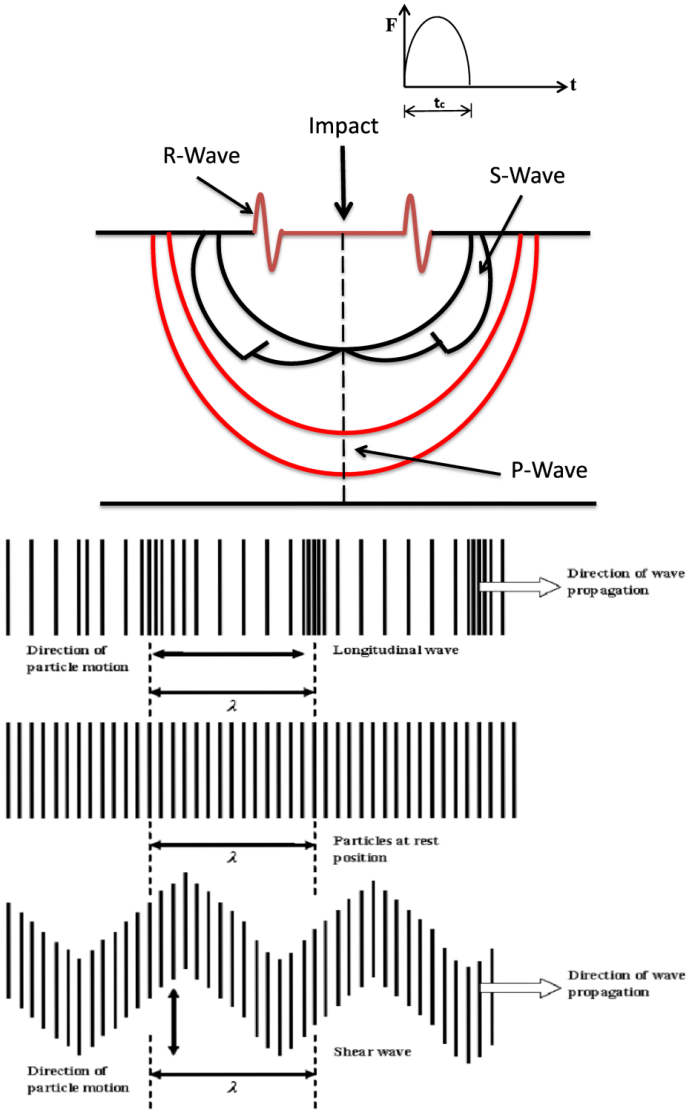


Figura 1.6 – *(a) Onda de estrés en una superficie dura de un material sólido. R wave representa las ondas de Reyleigh, S wave, las ondas transversales o de cizalla, y P wave las ondas longitudinales. Todas las ondas comparten la misma frecuencia. [2] (b) La longitud de onda de las ondas longitudinales y transversales es la misma. En el caso de la onda transversal, la pérdida de energía es mayor que en la longitudinal por su naturaleza de propagación. [3]*

1.4. Ensayo no destructivo

1.4.1. Tiempo de Vuelo

En este enfoque, se insertan dos sondas (una etiquetada como emisor y la otra como receptor) en la corteza, y luego se introduce energía mecánica en la madera mediante el impacto de un martillo en la sonda emisora. El método de ToF consiste esencialmente en medir el tiempo que tarda la onda de tensión en viajar entre dos puntos, desde el emisor hasta el receptor. La velocidad de la onda acústica se calcula a partir de este intervalo de tiempo usando la siguiente ecuación:

$$C_T = \frac{d}{\Delta T} \tag{1.4.1}$$

donde C_T es la velocidad acústica en el árbol, d es la distancia entre las dos sondas y ΔT es el tiempo de viaje.

Como se puede ver en la imagen 1.7, velocidad muy inferior puede indicar que el árbol no está sano y presenta problemas en cuanto a la estrutura de la madera. Hay múltiples formas de detectar y clasificar el tronco. Aquí ha sido solo propuesta esta, pero en el apartado 1.2.2.2 Measurement procedures de la versión anterior se detallan más ejemplos [6].

Figura 1.7 – *Medida de Fakopp más elemental. Créditos [4]*

Capítulo 2

Requerimentos del sistema

En primer lugar necesitamos realizar un proceso de ingeniería inversa al producto. Para ello necesitamos estudiar la memoria realizada ([6]) y comenzar con el análisis eléctrónico en el laboratorio.

2.1. TIK Módulo 1

Este primer modelo es el único desarrollado por Juan del Pino Mena, en el curso 2021/2022 como trabajo de fin de grado [6]. Hasta el momento, Fakopp microsecond timer es el dispositivo de referencia al que se le aplicó ingeniería inversa, y se encuentra bien detallado en el capítulo 2 de la memoria. También son analizados y caracterizados los Fakopp piezoelectric transducers SD-02 donde podemos ver tanto la resistencia y capacitancia del sensor y del cable coaxial. Cuanta menor longitud del cable hay más resistencia y capacitancia (**Figura 2.14 [6]**).

En la imagen2.1 vemos un ensayo no destructivo de un tablón. Se conectan los transductores al dispositivo y al material. Se enciende el temporizador y golpeamos el sensor con un martillo. El golpe ha de ser lo más paralelo posible a la dirección del clavo, en ese momento el dispositivo recoge el tiempo que tarda la onda longitudinal en recorrer el material.

En la figura 2.1 y la figura 2.3 podemos ver las diferencias evidentes en la tabla 2.1 (valores de peso sin incluir los transductores conectados al dispositivo). El coste tan elevado de la versión anterior frente a la competencia se debe a la investigación inicial frente a un dispositivo como es el Fakopp que se encuentra ya en proceso de venta al público y por tanto, industrializado con un stock determinado, cosa que no es nuestro caso.

Funcionalidad	Fakopp Microsecond Timer	TIK Module 1
Dimensiones	45x82x150mm	24x58x136mm
Peso	347g	210g
Pantalla	4 dígitos LCD	Táctil LCD 2.8“
Memoria	No disponible	448 KB ROM y 520 kB RAM
Consumo	50 mW	469.6 mW
Baud Rate	2400 bps	9600 a 921 600 bps
Conectividad	RS232	Wi-Fi 802.11 b/g/n, Bluetooth y USB
Material	Plástico	PLA Material de impresión 3D
Temperatura operativa	-10ºC a 40ºC	No disponible
Coste	2553 Euros	29.663,22 Euros

Tabla 2.1 – *Comparación de funcionalidades entre Fakopp y TIK Módulo 1*

Figura 2.1 – *Fakopp Microsecond-Timer con los transductores anclados [5].*

Figura 2.2 – *Primera versión desarrollada TIK [6].*

2.1.1. Hardware de TIK Módulo 1

El hardware gira en torno a las especificaciones del módulo y microcontrolador ESP32-WROOM-32D.

- **Power**: En el plano de alimentación, el dispositivo cuenta con 3 voltajes distintos. A la entrada la tensión de alimentación de 5V y 1A (con esa tensión no alimentamos ningún integrado), que pasa a los 3.3V mediante un convertidor AP3429 reductor DC/DC con una capacidad máxima de intensidad de 2A. A su vez, esa tensión mediante dos LDO NCP562SQ18T1G es reducida a 1.8V para polarizar los conectores hembra BNC.

- **Programming**: Se hace a través de un USB mini B que pasa al microcontrolador con las líneas RX/TX gracias al convertidor C340C que pasa de USB a UART.

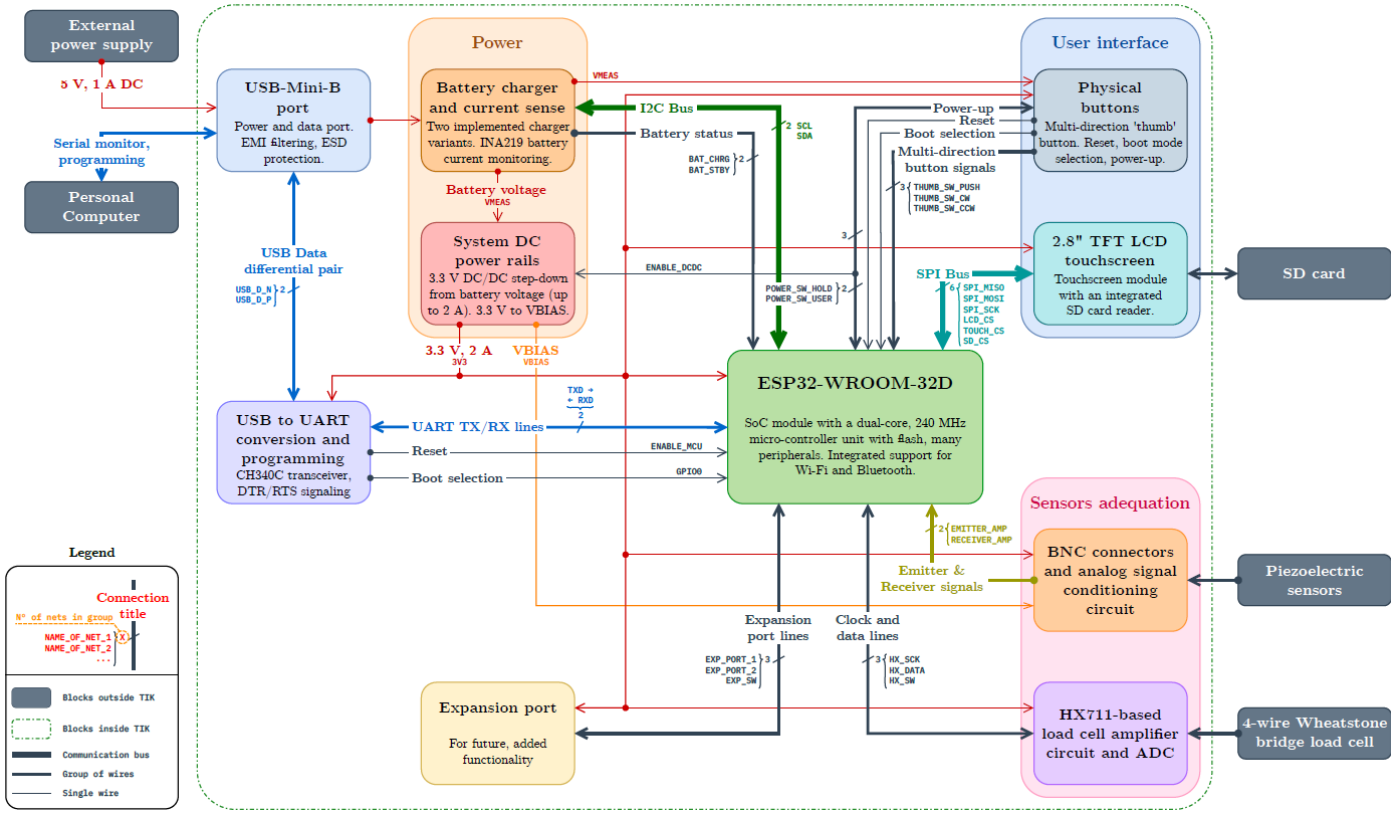


Figura 2.3 – *Diagrama de bloques circuital del TIK Modelo 1 [6].*

- **User interface**: Tenemos un botón que a su vez tiene 3 posiciones distintas y el módulo de pantalla ILI9341 de 2.8“ que cuenta con control táctil mediante el integrado XPT2046.
- **Expansion port**: Posibilidad de expansión y conexión externa con otros dispositivos mediante dos Jack de 3.5mm.
- **Sensors adequation**: Contamos con los BNCs hembra que recogen la información de las señales del material a ensayar así como la célula de carga HX711 que mediante puente de wheastone podemos medir peso, deformación ... etc para estimar correctamente la densidad y usarlo en el cálculo del MoE.

2.1.2. Software de TIK Módulo 1

En el lado del software, este dispositivo necesita una respuesta prácticamente momentánea y eso se consigue a través de la programación de RTOS. Este tipo de programación es usada en base a un hardaware con determinados núcleos y conseguimos alternar una u otra tarea en base a la prioridad que le damos para que se ejecute en un nucleo en específico o en cualquiera de ellos. Gracias a esta programación hacemos un dispositivo determinista y todo lo que sea capaz de hacer lo hará sin lugar a fallos. En resumen, este sistema operativo no lleva a errores y consigue robustez en cada acción que realiza. Para conocer más sobre este tipo de programación consultar el Apéndice F.

Por otro lado, comentar que el lenguaje de programación usado es C++, ideal para la manipulación de objetos y necesario para los sistemas embebidos como es el caso.

Para más detalles de hardaware y software del TIK Model 1, deberá consultar los capítulos 4 y 5 de la memoria

anterior [6].

2.1.3. Estado y problemas de TIK Módulo 1

El dispositivo fué llevado a acabo y se alcanzó un grado de funcionalidad muy alto. En el se tenía la posibilidad de navegar por distintos menús como ajustes, idioma, apagado y lo más importante; acceso a vicualizar las ondas recibidas y emitidas obteniendo así los tiempos y demás parámetros, pero esto último de forma artificial ya que no se llegó a producir ningún ensayo real. En resumen el dispositivo necesitaba aún progresos y la adición de funcionalidades que el hardware permitía como: almacenamiento de datos en la SD, graficar y calcular con algún algoritmo como la FFT o en el caso del trabajo de Juan el algoritmo AIC las señales, controlar el brillo de pantalla, monitorizar el estado de la batería, controlar la conectividad bluetooth y Wi-Fi . . . etc.

Y por otro lado solucionar los siguientes errores presentes:

- **LEDs de carga**: Cargador TP5056 no controla correctamente los LEDs de estado de carga. Estos LEDs funcionan como indicador y controlar si el dispositivo está ya cargado o no.
- **LDO inestable con osciloscopio**: El LDO que aporta los 1.8V para polarizar y asegurar la esbilidad en los conectores BNCs hembra no funcionan correctamente al conectar el osciloscopio a la salida de estos mencionados conectores.
- **Problema de alimentación del INA219**: Este integrado diseñado para monitorizar el estado de la batería no tiene el pin de suplimentación a los 3.3V, lo tiene conectado al polo positivo de la batería haciendo que si la batería baja de los 3V este se apague.
- **Circuito de encendido y apagado**: El circuito que se encarga de encender y apagar el dispositivo con un botón no funciona y por lo tanto no manda la señal al enable del DC/DC manteniendo el dispositivo apagado.

El resto de problemas son corregidos y detallados en la memoria de Antonio Morales, y son los relacionados con la manipulación de datos del ADC integrado en el ESP32 WROOM 32D, circuito de amplificación y acondicionamiento, así como representar los datos en la pantalla del dispositivo.

2.2. Requerimientos

Desde GranaSAT se demandan los siguientes requerimientos en cuanto a hardware, software y mecánica.

2.2.1. Requerimientos hardware

- **Modificación de tensiones de polarización de los conectores**: Debemos de evitar el LDO implantado por su generación de ruido, y en su defecto usaremos un divisor de ressitencia con un gran condensador de desacoplo que nos asegura una señal muy estable. Evitar de regerencia GND, nos ayuda a evitar todo el ruido de retorno que trnasmiten los integrado que están conectados.
- **Monitorización de la temperatura de la batería**: En la versión anterior no se tuvo en cuanta la RTC que tienen implementadas algunas baterias y que nos ayudan a controlar la temperatura de la batería de LiPo
- **Cambio de indicadores LEDs de batería**: Actualmente, no hay forma de conocer el estado de la batería cuando está siendo cargada. Un rediseñño tanto mecánico como de PCB será necesario para conseguirlo.
- **Adición de buzzer para alarmas**: Un buffer nos ayuda emitir sonidos en momentos concretos como cuando el dispositivo se queda encendido y no se está usando, o para prgramar algún tipo de alarma en un día concreto gracias al RTC.

- **Añadir RTC para control de calendario**: Una RTC es esencial para tener un control horario y mensual como cualquier dispositivo inteligente actual.
- **Rediseño de botón de encendido**: Puesto que el botón de encendido genera problemas, se rediseñarán las conexiones y el tipo de botón utilizado consiguiendo así tener un un lugar más cómodo para nuestra mano.
- **Eliminación de circuitos de célula de carga**: Ya no son necesarios los circuitos utilizados para pesar el material a ensayar ya sí conseguir su densidad.
- **USB tipo C**: Actualizar el puerto USB no es solo un requerimiento desde GranaSAT, si no también de la UE.
- **LEDs de control para la depuración de señales digitales conflictivas**:Los LEDs nos ayudan visualmente a sabes que pistas están en un valor alto o bajo y es muy útil para el proceso de validación. Se han añadido LEDs en señales críticas como el enable el DC/DC que lo mantiene encendido.
- Consultar para todo ello la sección 4.1.

2.2.2. Requerimientos software

- **Organizar todo el proyecto**: El proyecto actualmente cuenta con ficheros de funcionaes que finalmente no han sido desarrollados. Como por ejemplo el algoritmo de Akaike. Para es necesario borrar todo aquello que no necesitemos y dejar únicamente lo implementado y utilizado.
- **Documentar el código con algún software de forma profesional**: Con algún software, es necesario generar un documento PDF que relate sin necesidad de observar el código como se organiza es fundamental. Apéndice D
- **Gestionar la carga**: Actualmente no hay forma alguna de conocer el estado de la batería mientras el dispositivo es usado. Para ello se utilizará algún integrado que monitorice el estado de la misma y nos arroje datos sobre la autonomía destante. Revisar **??**
- **Ampliar y darle funcionalidad a la interfaz gráfica así como funcionalidades**: Todo lo implementado en la versión anterior tiene actualmente solo capacidad informativa, realmente no actúa sobre el sistema y no aporta funcionalidad real. sección 5.2.1
- **Conectividad**: El SoC nos arroja connectividad Wi-Fi y bluetooth pero no se ha interactuado con ellos. Se necesitará desarrollar alguna pantalla en la interfaz gráfica que nos aporte la capcidad de gestionar el equipo.

2.2.3. Requerimientos mecánicos

- **Orificio para el lápiz táctil**: Este orificio puede que al ser imprimido sin soporte (ya que dudo que la impresora sea capaz de hacer un soporte que se adentre tanto en el material) hace que los hilos del filamento no estén pegados entre sí quedando una red muy frágil de filamento (ver 2.4a).
- **Rotura de carcasa en tornillo M3 para sujección**: Esta rotura puede deberse a que el detalle de corte para la entrada del cabezal del tornillo hace que quede muy poco espacio con el borde y la rotura sea muy probable. Ver 2.4b
- **Rotura de carcasa en la zona del puerto USB**:En la imagen 2.4c podemo ver esta rotura similar a la descrita anteriormente. Dejar delgadas capas de material en zonas donde la tensión mecánica es mayor además de zonas de mucho uso puede provocar este tipo de desperfectos.
- **Botón de encendido y logo de LWFF**: El botón de encendido no es el más comodo. SI nos fijamos en un dispositivo móvil, los botones se ubican en el lateral. Además, el logo de la parte superior no queda bien definido y debe ser modificado.
- **Separación entre carcasas**: Hay cierto desajuste entre la carcasa superior e inferior, esto se puede deber a que los anclajes no son los ideales y por lo tanto será uno de los objetivos a mejorar. Ver 2.4d

2.2.4. Otros requerimientos

- **Presentación RTOS**: Para dar a conocer el Firmware implmentado se hizo una presentación a los compañeros del laboratorio.

2.3. Procesos de la ingeniería inversa

El trabajo de un ingeniero se centra en gran medida en el diseño. El diseño establece y define soluciones para problemas que no se han resuelto antes, o nuevas soluciones para problemas que previamente se han resuelto de una manera diferente. Un buen diseño requiere creatividad, toma de decisiones basada en muchos parámetros y compromisos entre requisitos contradictorios. El diseño es el producto de la planificación y el trabajo, y generalmente se lleva a cabo descomponiendo el problema en partes manejables.

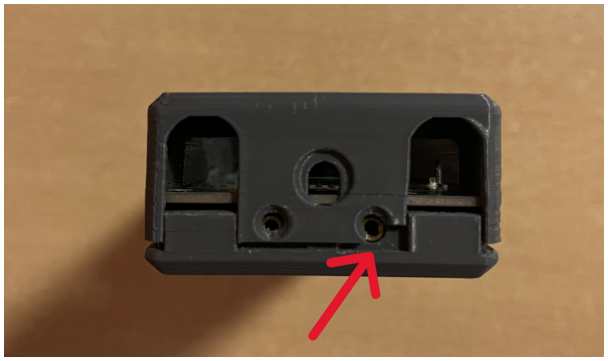
El Proceso de Diseño de Ingeniería, consiste en una serie de pasos que los ingenieros siguen para resolver un problema.:

- **Definición del problema**: La respuesta a qué es el problema, quién lo tiene y por qué es importante resolverlo.
- **Investigación y análisis de antecedentes**: Consiste en observar más de cerca el problema, averiguando cuál es el estado del arte y otras soluciones existentes, y evitando errores cometidos en el pasado.
- **Ingeniería de Requisitos**: Consiste en el análisis y especificación de los requisitos para establecer el comportamiento del sistema, sus funciones y limitaciones operativas.
- **Lluvia de ideas y evaluación de conceptos**: Propuesta de muchas soluciones posibles y creativas. Se debe realizar una evaluación de las ventajas y desventajas de cada alternativa.
- **Especificar la arquitectura del producto**: Qué tecnologías va a usar la solución y en qué configuración.
- **Desarrollo y creación de prototipos**: Implementación, refinamiento y mejora de la solución elegida en la arquitectura física, hardware y software.
- **Pruebas y validación**: Prueba y evaluación de si el prototipo cumple con las necesidades del usuario y los requisitos.
- **Retroalimentación**: El proceso de diseño implica múltiples iteraciones y rediseños de la solución final.
- **Comunicación de resultados**: Como último paso, se comparten los resultados de una o varias iteraciones.

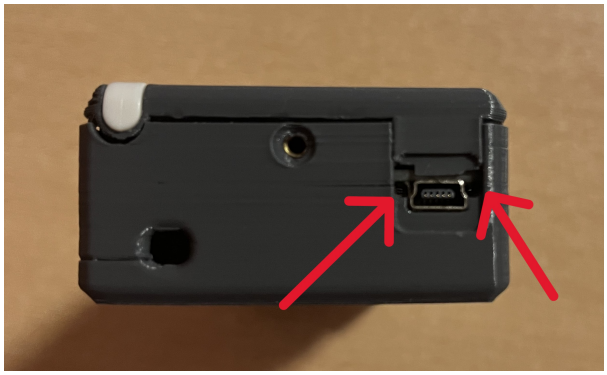
2.4. Diagrama de Gantt

En este apartado de muestra un diagrama de Gantt con la temporización de las tareas de este trabajo.

(a) *Ubicacion del lapiz para pantalla táctil.*



(b) *Orificio para la ubicación de tornillo de sujección.*



(c) *Detalle para el puerto hembra USB Mini B.*

(d) *Separación entre carcasa superior e inferior.*

2

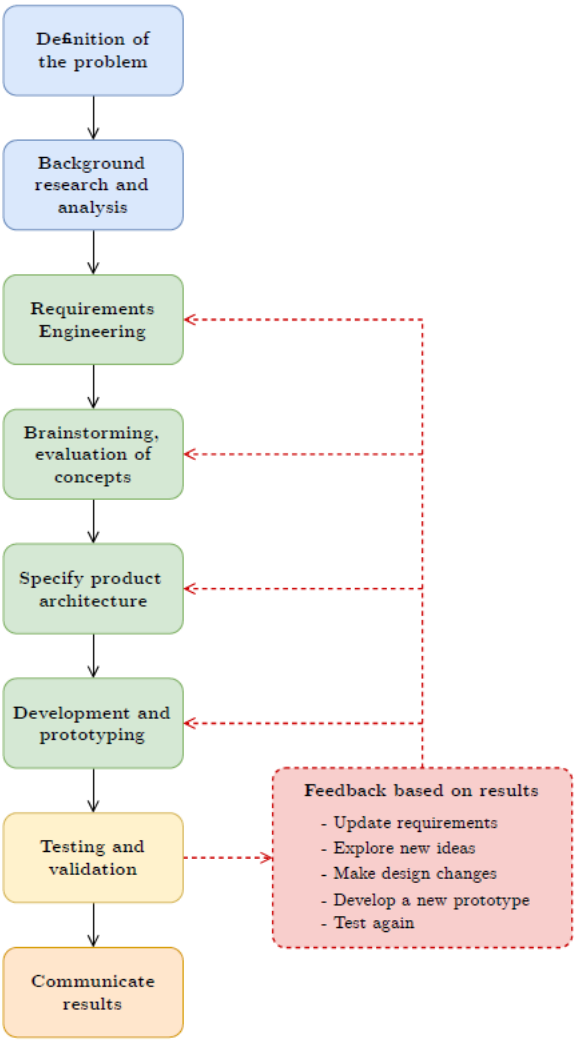
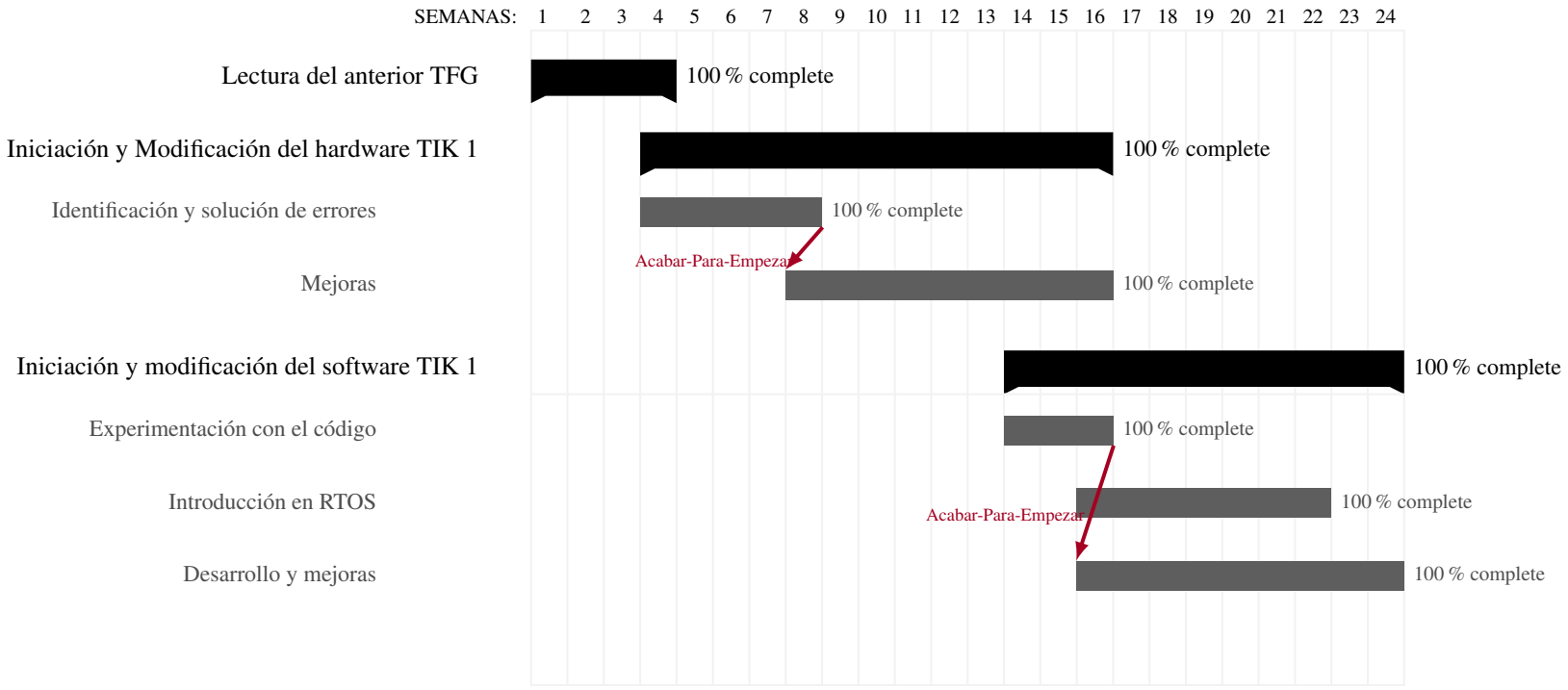
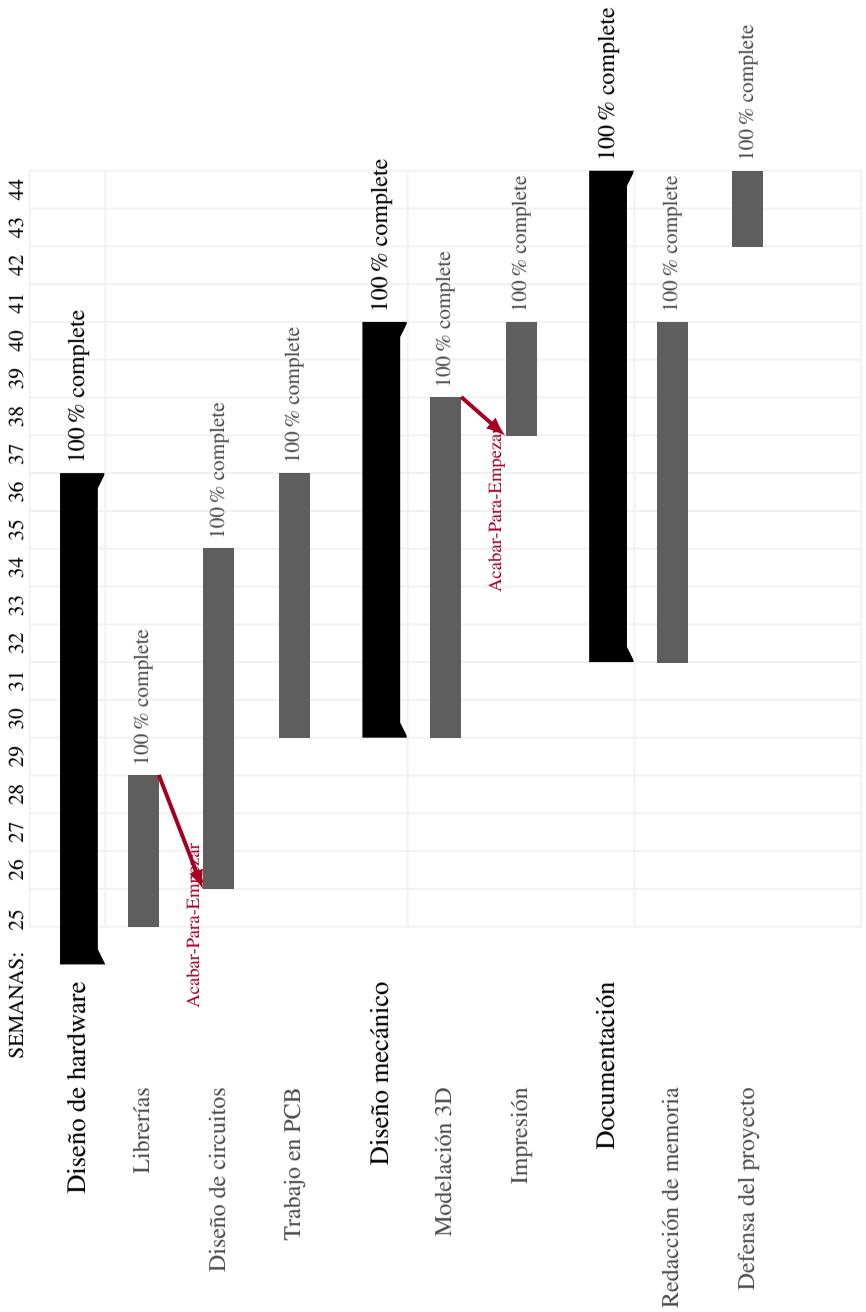


Figura 2.4 – *Diagrama de bloques sobre el proceso de ingeniería. Figura 1.9 [6].*


SEMANAS: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Lectura del anterior TFG
100 % complete
Iniciación y Modificación del hardware TIK 1
100 % complete
Identificación y solución de errores
100 % complete
Acabar-Para-Empezar
Mejoras
100 % complete
Iniciación y modificación del software TIK 1
100 % complete
Experimentación con el código
100 % complete
Introducción en RTOS
Acabar-Para-Empezar
100 % complete
Desarrollo y mejoras
100 % complete


SEMANAS: 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
Diseño de hardware
100 % complete
Librerías
100 % complete
Diseño de circuitos
100 % complete
Acabar-Para-Empezar
Trabajo en PCB
100 % complete
Diseño mecánico
100 % complete
Modelación 3D
100 % complete
Acabar-Para-Empezar
Impresión
100 % complete
Documentación
100 % complete
Redacción de memoria
100 % complete
Defensa del proyecto
100 % complete

Capítulo 3

Ingeniería inversa

3.1. Circuito de encendido y apagado

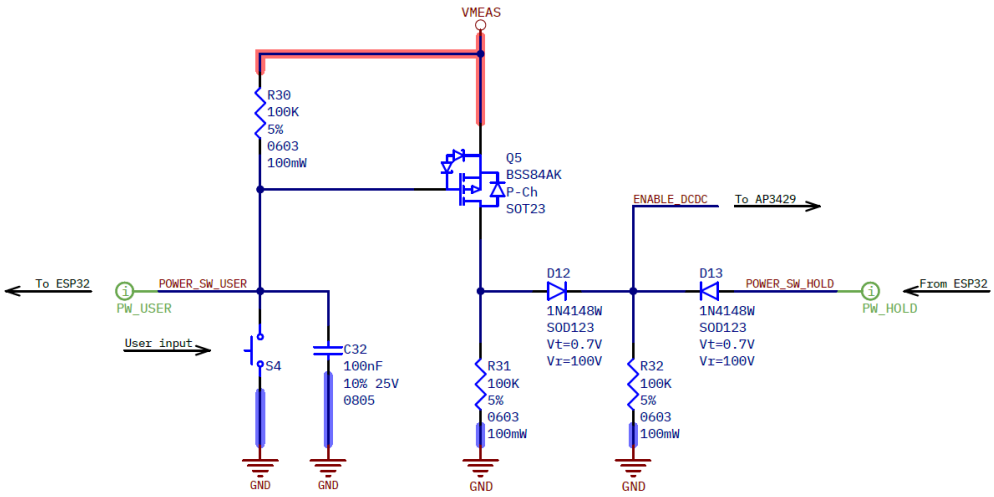
Como se ha mencionado en la sección 2.1.3 el circuito que se puede ver en la imagen 3.1a funciona como control de encedido y apagado del dispositivo. Al pulsar el botón S4, la puerta del MOSFET pasa de Vmeas a GND, el transistor empieza a conducir ya que la tensión en la puerta baja de la tensión indicada como se puede ver en las ecuaciones 3.1.3. La tensión supera la de conducción en polarización directa del diodo D12 y por lo tanto, la señal enable pasa a nivel alto y conseguimos los 3.3V que alimentan a todo el dispositivo. A su vez, cuando se ha pulsado el botón, este nivel bajo lo recoge el pin POWER_SW_USER que está conectado a un GPIO del microcontrolador y manda una señal a nivel alto con el pin declarado como salida manteniendo el enable de forma constante.

Con este circuito conseguimos encender el dispositivo de forma mecánica y mantenerlo encendido de forma digital gracias al microcontrolador que pone en alto el pin POWER_SW_HOLD. La señal Vmeas, como vemos en la figura 3.38, es prácticamente la tensión de la batería ya que la resistencia de shunt es muy baja, en concreto 200 mΩ. Con ello podemos controlar el eapagado y mandar una alerta de apagado de dispositivo si se recoge la intención del usuario, avisando así si no se han guardado los datos de las medidas del END.

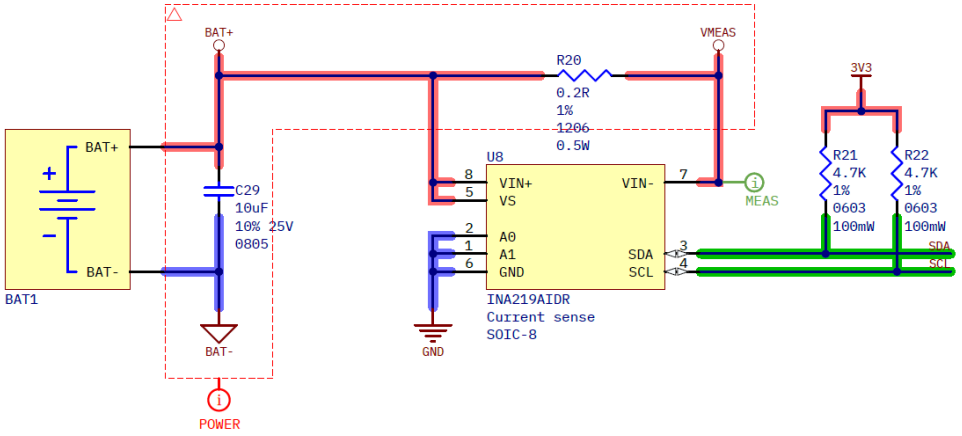
$$V_G - V_S < -1{,}5V \tag{3.1.1}$$
$$V_G < -1{,}5V + V_{Meas} \tag{3.1.2}$$
$$V_G < 2{,}2V \tag{3.1.3}$$

En los laboratorios de GranaSAT, se hizo un pedido de varías PCBs y para familiarizarme antes de usar la placa que funciona y empezar a modificarla, decido revisar los fallos de las demás placas. En el caso de la PCB# 2 el circuito no funciona por los siguientes motivos encontrados al depurar pistas en busca de cortos:

- Cortocircuito en DC/DC: En el convertidor hay dos patillas que están en corto, LX y GND, las cuales son la salida del convertidor y el pin que va a masa, respectivamente. Como se puede ver en la figura 3.2, en los pines mencionados debe de haber una resistencia del orden de kohms cuando está en corte y en este caso es de 0.2-0.3 ohms.
- Cortocircuito en LDO: Entre los pines de masa y 3.3V hay otro corto. El digrama interno del NCP562SQ18T1G, es imposible de encontrar. Sin embargo, contrastando con el resto de reguladores NCP562SQ18T1G del laboratorio, este corto no está presente.
- Cortocircuito de ByPass: Como vemos en la figura 3.3, ese divisor tiene el objetivo de obviar si queremos el uso



(a) *Circuito de encendido y apagado. Figura 5.18 [6]*



(b) *Circuito de monitorización de batería. Figura 5.4 [6]*

Figura 3.1 – *Comparación de circuitos: (a) Encendido y apagado, (b) Monitorización de batería.*

del LDO, y poner otra tensión distinta de polarización. En este caso está circuitado sin resistencias entre sí por lo que, se están cortocircuitando dos pistas de potencia.

- Tensión de elimentación Vmeas inestable: Puesto que en esta versión hay ciertas particularidades al contar con la disponibilidad de usar batería de Litio o batería de NiMH, hay una resistencia R25 (página 8 de 21 apartado de esquemáticos [6]) que debe de estar soldada en el caso de usar la última opción de batería. Desoldado este cortocircuito entre VBAT- y GND desaperece esta inestabilidad que se ve en la figura 3.7.

Finalmente, esto es solucionado desoldando y eliminando el cortocircuito de VBias como se ve en la imagen 3.5:

Una vez hallado el error de alimentación de la PCB#2 sin opción a si realmente el proble ha sido solucionado por falta de DC/DC en el laboratorio, paso a la #0 que actualmente es la funcional. En esta placa persevera el error de alimentación comentado obteniendo en la puerta constantemente 665mV y haciendo así que conduzca el MOSFET constnatemente sin control de encendido.

Comienzo por simular el circuito de la figura 3.8 en LTspice y analizo que pasa si la resistencia de pull up R30 varía su valor.

Como podemos ver en la figura 3.9a el tiempo de respuesta es menor que en la 3.9b pasando de unos 5ms a los

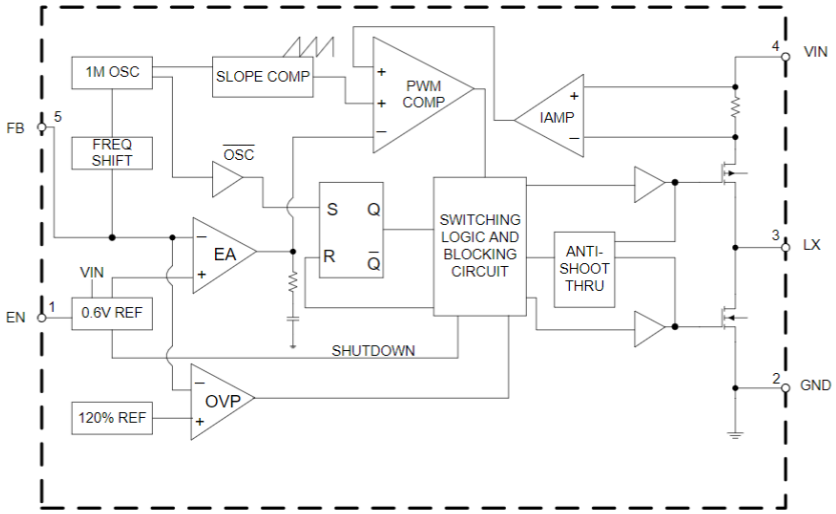


Figura 3.2 – *Diagrama de bloques interno DC/DC AP3429. Página 3/11 [7]*

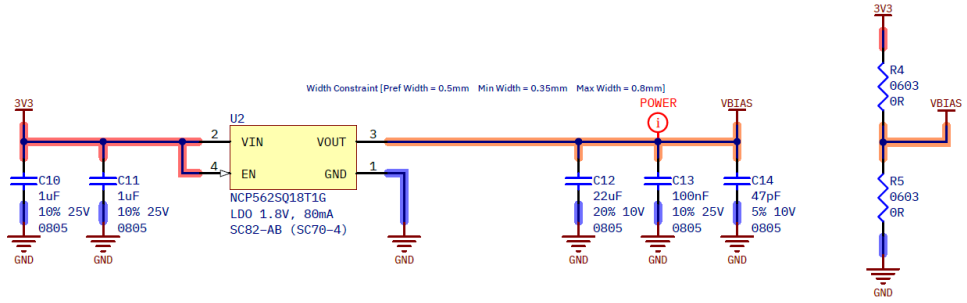


Figura 3.3 – *Circuito LDO junto con el de byPass para elegir otra tensión de polarización distinta a los 1.8V. Figura 5.18 [6]*

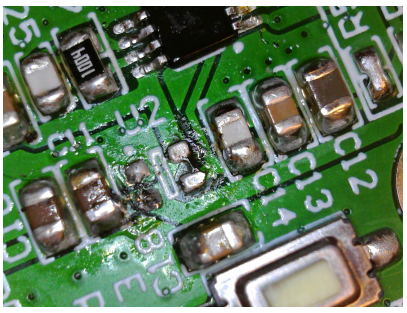


Figura 3.4 – *Desoldadura de LDO*

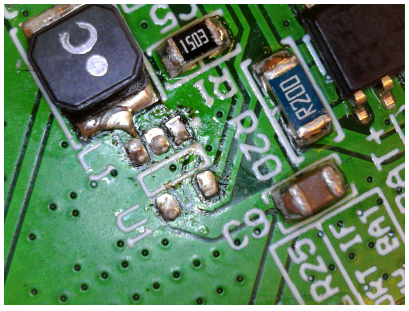


Figura 3.5 – *Desoldadura de DC/DC*

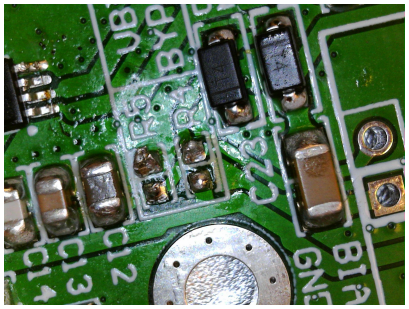


Figura 3.6 – *Eliminación de cortocircuito entre resistencias de Bypass*

12ms, pero en ambos casos la resistencia no es el problema y el circuito tiene el comportamiento esperado. Con ello aseguramos que el valor de la resistencia afecta al tiempo que tarda en mandar la señal de encendido al DC/DC. En cuanto a la tensión que introducimos en el enable, nos encontramos dentro del valor recomendado por el fabricante, ya que se indica en la página 4/11 [7] que el rango es de Vin - 0.3V a Vin + 0.3V, siendo Vin los 5V del puerto USB y

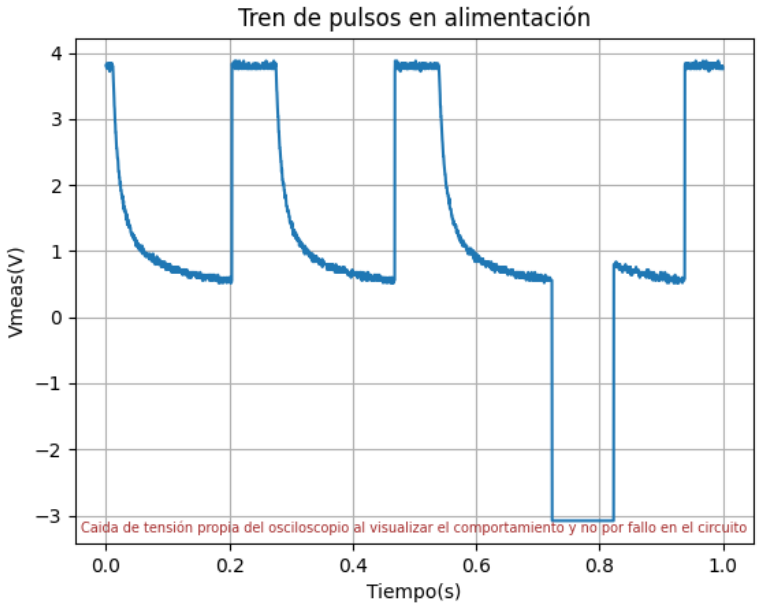


Figura 3.7 – *Tren de pulsos en Vmeas.*

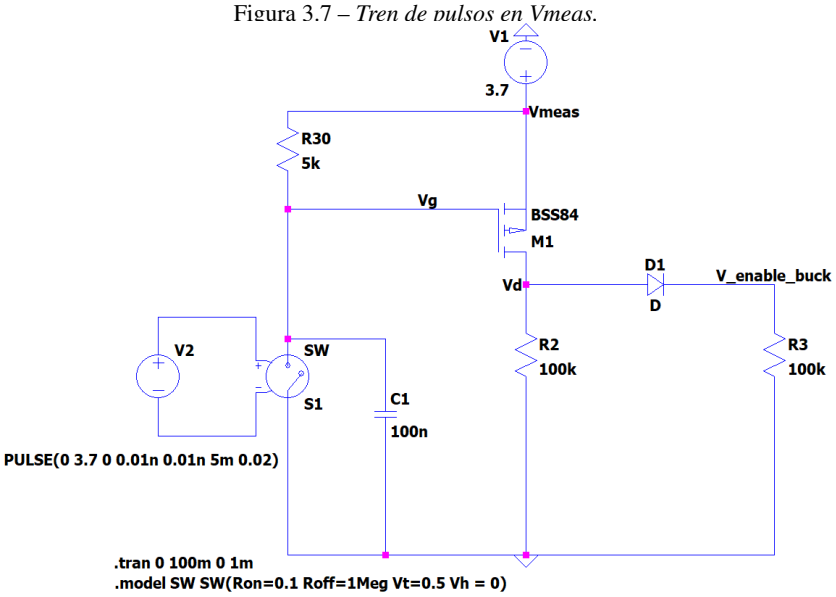
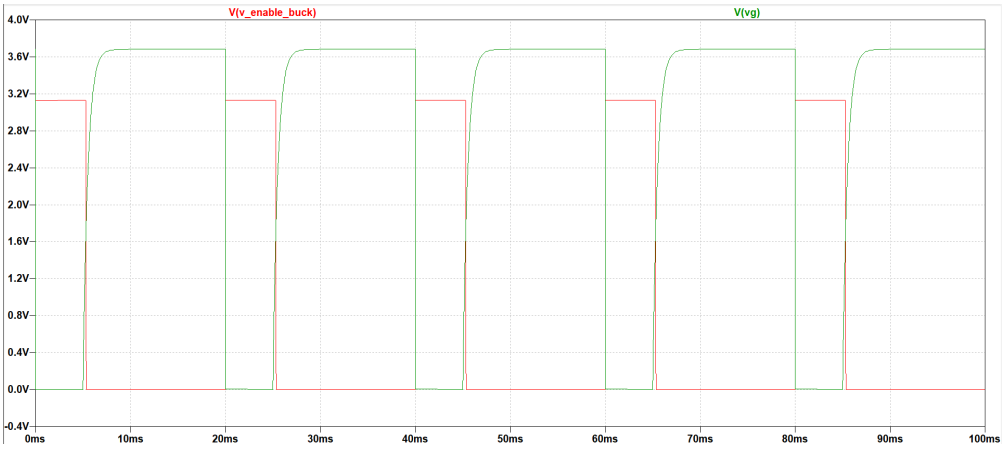


Figura 3.8 – *Circuito de encedido y apagado.*

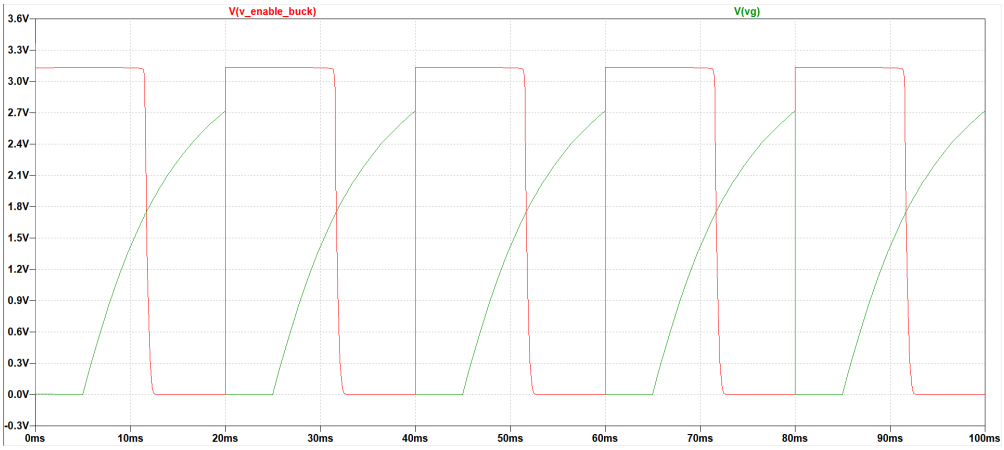
nuestro valor será como máximo el de la tensión de batería Vmeas que en su punto más alto y en un corto tiempo será de 4.2V para una batería de litio de 3.7V.

Se decide montar ahora el circuito en una de las placas vacías para evidenciar que el problema es del resto componentes que al recbir la alimentación de 5V pueden estar realizando acoplamiento de algún tipo impidiendo que la tensión de la puerta varíe aunque se pulse el botón y quede fija a los 665 mV en la puerta 3.10.

Podemos ver en la figura 3.11a, como el circuito montado responde como debe de ser y envía la señal de enable en un breve periodo de tiempo, el necesario para que el reductor DC/DC se encienda y ponga en la salida los 3.3V, como conclusión podemos afirmar que ni el diseño de pistas en la PCB ni el propio circuito tienen el problema.



(a) *Respuesta circuito de encendido y apagado con R30 de 5kΩ.*



(b) *Respuesta circuito de encendido y apagado con R30 de 100kΩ.*

Figura 3.9 – *Comparación de circuitos: (a) 5kΩ. (b) 100kΩ.*

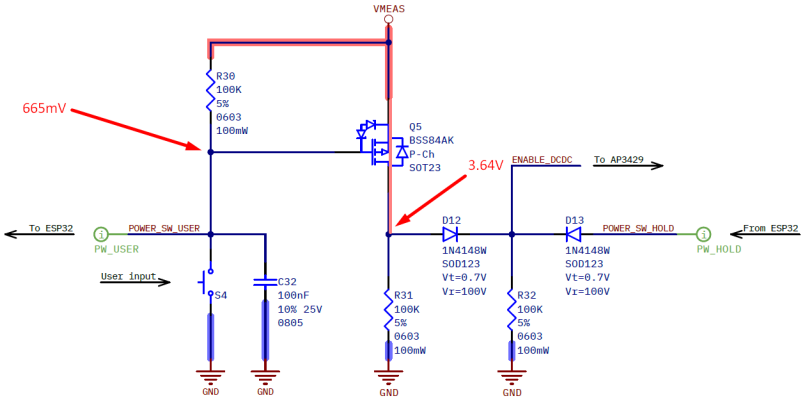
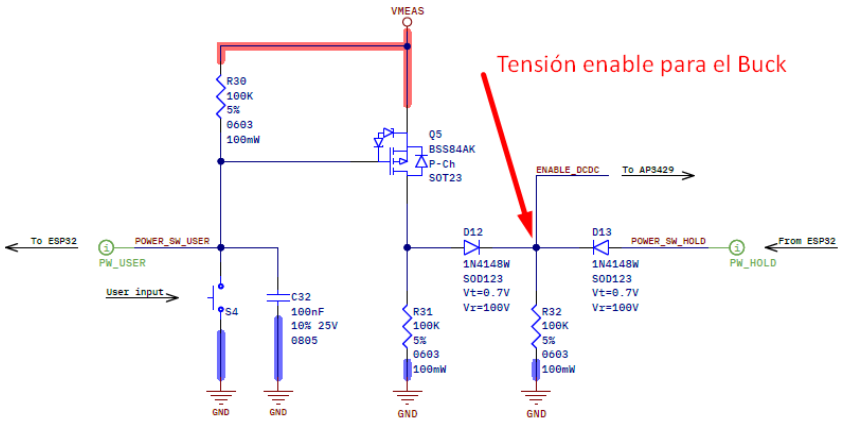
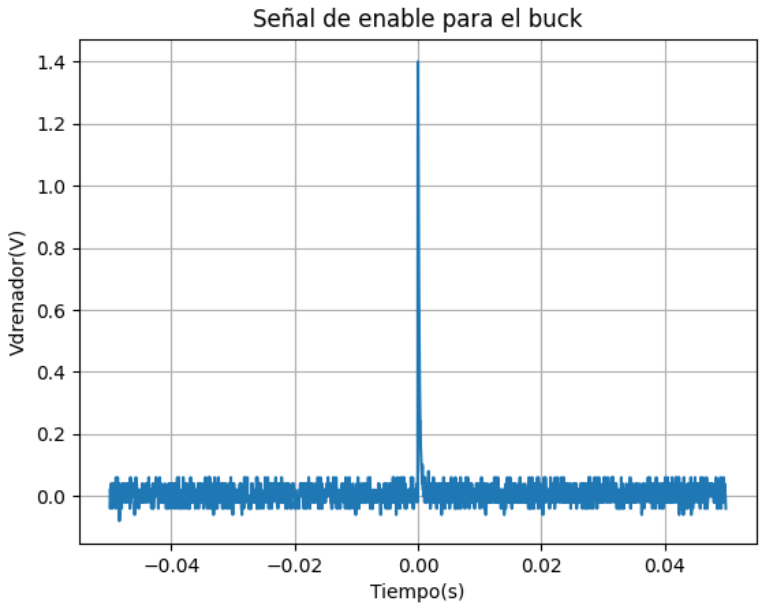


Figura 3.10 – *Tensión del circuito de alimentación en la PCB#0 pulsando o no el botón S4.*

Por lo tanto, debe de haber un problema de soldadura, tras volver a soldar todos los puntos de soldadura de cada componente (diodos, trnasistores, resistencias, botón S4 de la figura 3.1a) y colocando dos canales del osciloscopio, el canal 1 en la puerta del transistor para recoger la actividad del botón y otro canal 2 en el drenador del MOSFET Q5



(a) *Circuito de encendido y apagado con la señal de enable para el DC/DC marcada.*



(b) *Respuesta circuito de encendido y apagado en la señal de enable para el DC/DC.*

Figura 3.11 – *Circuito y respuesta de encendido del reductor DC/DC.*

como se ve en la figura 3.12.

Como podemos ver, el circuito ahora responde correctamente. Sin estar pulsado el botón S4 de la figura 3.13a la tensión en la puerta (CH1) tiene un valor alto (el que proporcione la batería) y en el drenador (CH2) tenemos una valor alto gracias a que el ESP32 WROOM 32D mantiene este pin en alto para mantener el dispositivo encenddido, al ser pulsado el botón S4 de la figura 3.13b cambian los valores y la tensión que se envía al enable del DC/DC (CH2) pasa a un valor alto mientras que la puerta del transistor (CH1) queda cortocircuitada a masa.

En la figura 3.13a, la señal CH2 es inestable y eso se debe a que la señal digital proporcionada con un pull up no es limpia. Probablemente sea porque este pin no tiene una gran impedancia en el encendido, y tampoco una baja impedancia al mantener la tensión en valor alto. Por ello en el rediseño, este pin ha sido cambiado como se puede ver en el capítulo 3 en los esquemáticos del nuevo modelo 5.1.1.1.

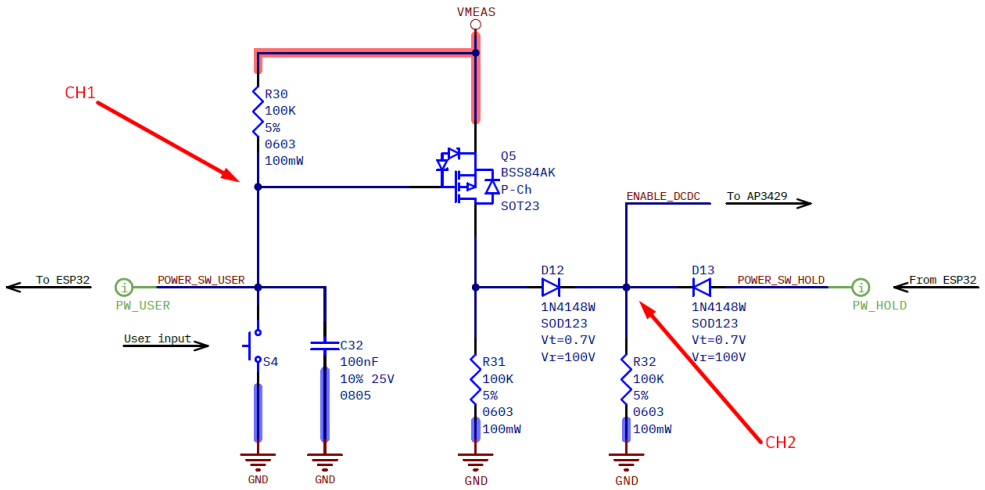


Figura 3.12 – *Tensión del circuito de alimentación en la PCB#0 pulsando o no el botón S4.*

El tiempo de descarga del condensador también puede ajustarse si buscamos mayor rapidez al tocar el botón. Este tiempo es el denominado tiempo de rebote y como se puede ver en la siguiente expresión 3.1.6.

$$V_{th,low-high}/V_{cc} = e^{-T/RC} \tag{3.1.4}$$

$$R_{30} = \frac{-T_{Bounce}}{Cln(\frac{V_{th,low-high}}{V_{CC}})} \tag{3.1.5}$$

(3.1.6)

3.1.1. Tiempo de reacción del circuito de encendido y apagado

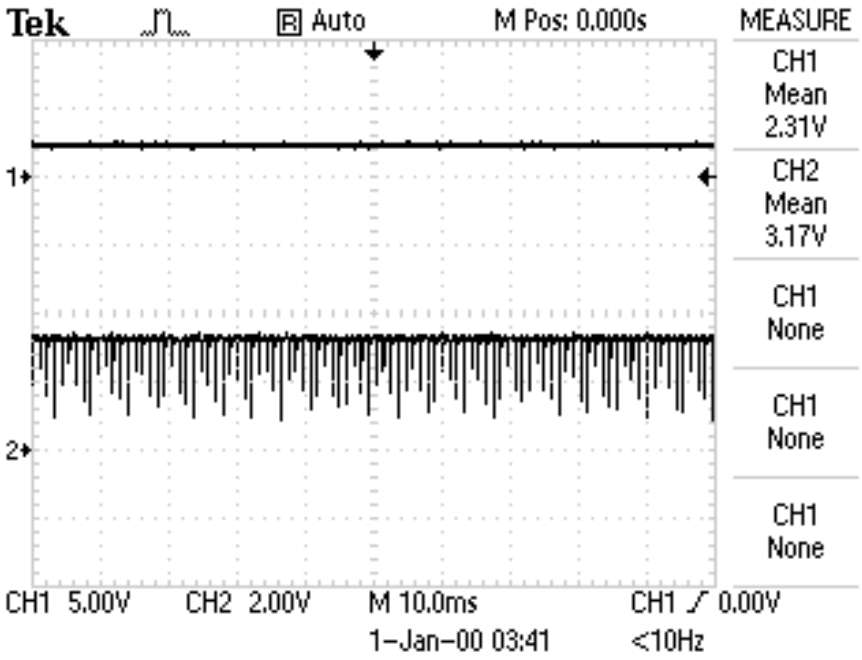
Una vez que hemos resuelto el problema del encendido del dispositivo, pasamos a analizar el tiempo que tarda el microcontrolodar en encender y apagar el equipo. De color amarillo recogemos la interacción del botón que pasa de valor alto a valor bajo cuando se pulsa el botón, hasta que el pin digital se pone en alto y enciende el dispositivo, tarda 356 ms como se ve en la figura 3.14a. Por otro lado, la lectura del pin reacciona instantáneamente cuando detecta de nuevo el pulso y es capaz de bajar el valor sin apenas tiempo de diferencia 3.14b. La configuración de pines la tenemos en la figura 3.15.

3.1.2. Alternativa de circuito de encendido

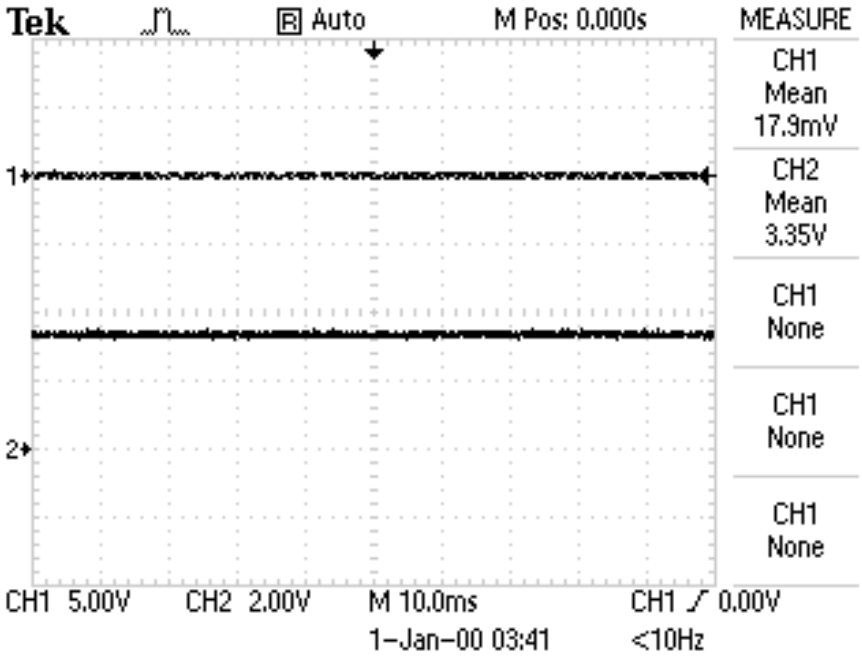
Al depender de los pines digitales del microcontrolador y por tanto, su comportamiento puede no ser el más optimo como hemos visto. He encontrado distintas alternativas para encender y apagar el dispositivo de con dos botones, uno para encender y otro para apagar. Necesitamos dos pulsadores ya que en este caso como se ve en la figura 3.16b encontramos un primer botón S1 que al ser pulsado pone la tensión de base a masa y, cuando es pulsado S2 la tensión de puerta pasa a ser la de alimentación de 3.7V. En la figura 3.16b vemos que la tensión de emisor es la señal de de ON/OFF para el enable del DC/DC y conseguimos por tanto un correcto funcionamiento.

3.2. Testeo de LEDs de carga

Es importante indicar al usuario cuando el dispositivo está cargado, para ello el cargador incorporado TP4056 ya se encarga de hacer esta gestión y por lo tanto la elección de conectar estos LEDs al microcontrolador no es la más



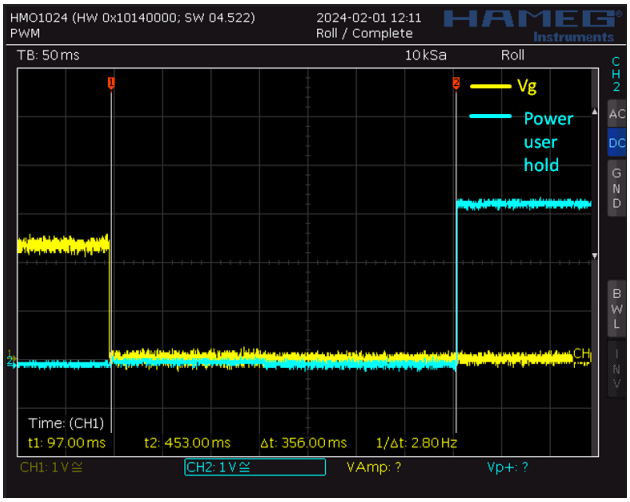
(a) *Respuesta circuito de encendido y apagado S4 sin pulsar.*



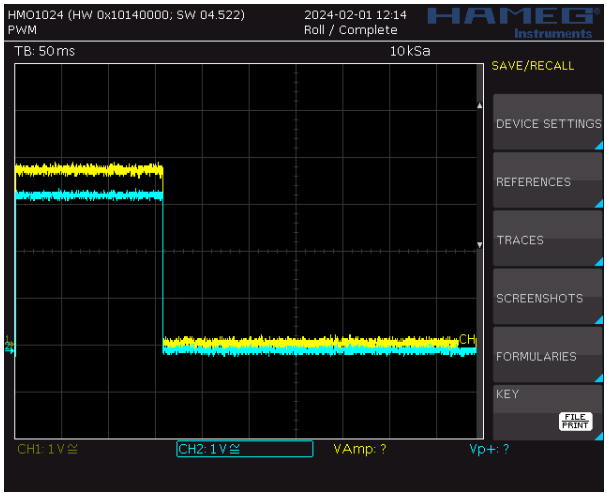
(b) *Respuesta circuito de encendido y apagado con S4 pulsado.*

Figura 3.13 – *Respuesta circuito de encendido y apagado S4 pulsado.*

acertada. Además, el comportamiento interno de los pines como ya hemos visto en de la imagen 3.13a es inestable y podría hacer parpadear la luz del diodo. En este caso, he comprobado programando la alternancia de parpadeo entre un LED y otro como podemos ver en las imágenes 3.18b.



(a) *Tiempo de detección del microcontrolador en mantener en valor alto el enable del DC/DC.*



(b) *Tiempo de detección del microcontrolador en detectar la nueva pulsación y bajar el valor del enable del DC/DC.*

Figura 3.14 – *Respuesta del circuito de encendido y apagado al hacer doble pulso en S4 para encender y apagar el dispositivo.*

3.3. Control de conexión de la batería

Puesto que la batería no tiene ningún conector del tipo JST XH2.54mm que se ve abajo en la imagen 3.20, solo hay dos pines para el polo positivo y negativo, que en el lugar de la batería son dos pines hembra, que se conectan independiente como se ve en la imagen 3.19. En mi caso, como solución alternativa, he decidido poner un jumper y dos pines machos eliminando los anteriores. Soldando los dos cables de la batería y conectando un jumper decidimos cuando queremos controlar la conexión de la batería al circuito y cuando no, así evitamos descargas de batería por descuido.

3.4. Eficiencia en el consumo

Actualmente, en el TIK Modelo 1, la pantalla al alimentarse proporciona el máximo brillo posible. Esto es algo que se debe solucionar ya que, el brillo de la pantalla supone uno de los mayores gastos energéticos de un dispositivo. Controlando este brillo conseguimos que nuestro equipo se encuentre en los estándares actuales de dispositivos

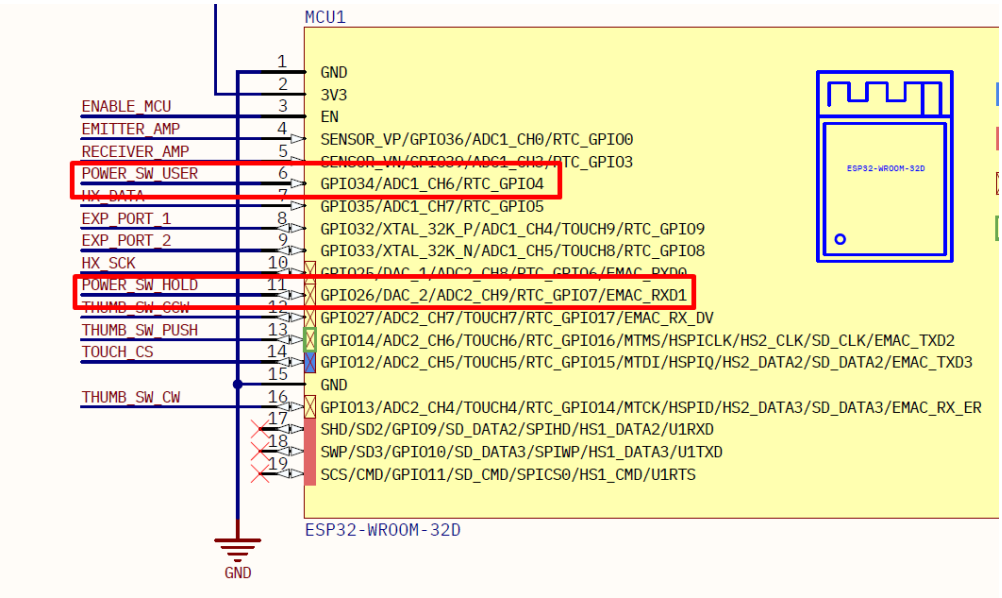


Figura 3.15 – *Pin de lectura PIN_SW_USER y de salida PIN_SW_HOLD. Página 6 de 21 del apartado de esquemáticos [6]*

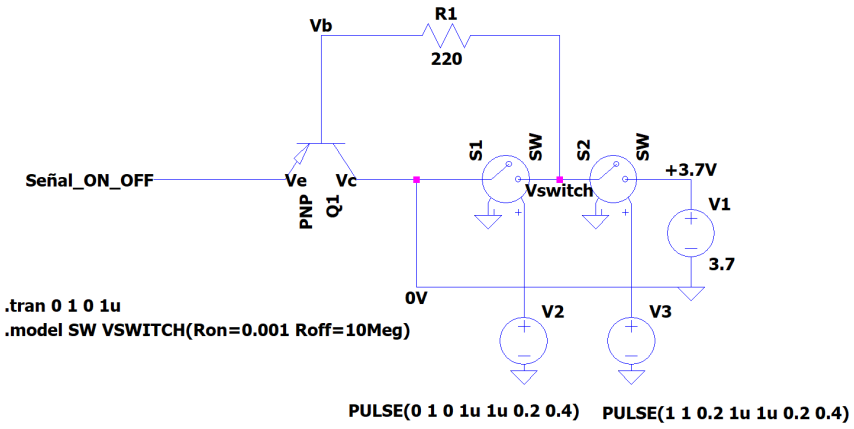
máviles, y quedando por encima de la actual competencia de dispositivos de END como el de Fakopp, cuya pantalla es muy pobre y ha sido comparada en la 2.1.

Para analizar el consumo del dispositivo, necesitamos un dispositivo como el Power Analyzer (revisar la sección E.1) para controlar con exactitud la medición de consumo al pulsar el botón S4 de encendido. Este consumo dependerá de los integrados que demanden intensidad, en el caso de 3.23 tan solo tenemos conectada la pantalla y toda ella en blanco. Este valor se encuentra entre el típico y máximo (tabla 5.5 [6]). Sin emabrgo, se encuentra algo elevado, sobre los 250 mW a partir de una batería de 3.7 V y alcanzando una demanda de intensidad de 67 mA.

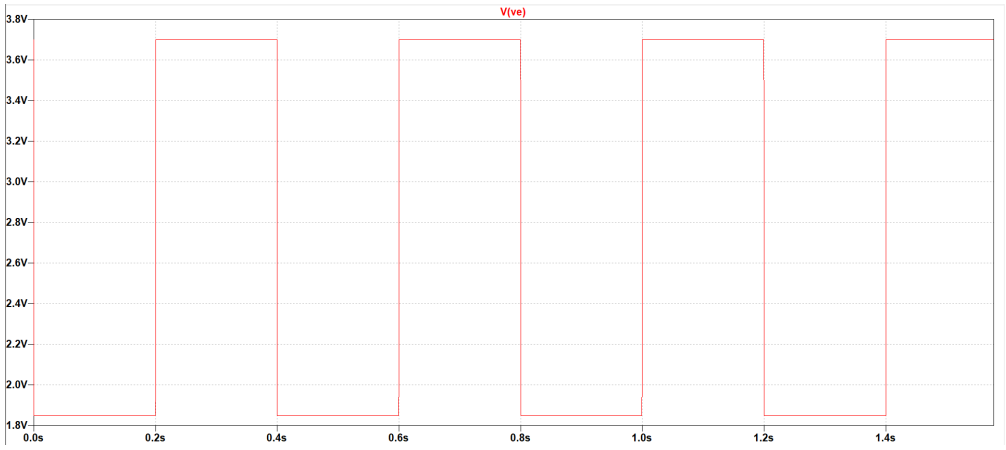
Como vemos en la figura 3.23, tenemos dos picos de demanda al encender y apagar el equipo, esos picos corresponden en la pantalla a una sobreiluminación de pocos milisegundos. Este comportamiento se deben a que los condensadores de la pantalla inicialmente se encuentran descargados, y necesitan estar cargados para valores internos de referencia para el integrado de la pantalla táctil, además de los reguladores de 5V a 3,3V. Una posible solución es añadir más condensadores de desacoplo cerca del pin de alimentación y/o añadir un integrado DC/DC reductor como el AP3429 que usamos pero con la función especial de suave arranque. El Texas Instruments TLV62568 es un ejemplo que nos puede servir para suplir este problema contando con arranque suave y alta eficiencia del 95 % con una frecuencia de conmutación de 2MHz fija. [36]

También podemos observar como una vez se estabiliza la demanda de intensidad, se mantiene constante y lo ideal sería poder controlar esa intensidad para reducir la potencia de consumo. Para ello necesitamos aplicar una señal PWM que nos aporte más o menos tensión en el pin LED del módulo ILI9341. Para conseguir esa señal, necesitamos que nuestro módulo integre un transistor con una resistencia Pull-Down en la puerta. Todo ello se debe encontrar entre el pin externo del módulo y la terminación final de la pantalla. Depende del módulo de pantalla ILI9341 que tengamos, hay algunos que lo integran y otros que no como se ve en la figura 3.24a. Como se ve, nuestro módulo si cuenta con el transistor.

El TIK Modelo 1 V0.6, no cuenta con el enrutado necesario y tenemos el pin LED cortocircuitado con el pin de alimentación Vcc (observar la figura 3.25a), por lo tanto, recibimos la tensión de alimentación y por ello el brillo siempre es el máximo. Cortando la pista físicamente de la PCB y uniendo el pin del ESP32 destinado para controlar la iluminación del LED de carga (3.25b), y el pin LED de la propia batería mediante uno de los finos cables que tienen aislamiento en todo su recorrido excepto en las puntas que han sido peladas con el soldador, conseguimos realizar la conexión para controlar el brillo mediante la señal PWM mencionada desde el microcontrolador como se puede ver en la figura 3.26a.



(a) *Alternativa de circuito de encendido y apagado.*



(b) *Respuesta del emisor del BJT ante la pulsación de uno u otro botón*

Figura 3.16 – *Respuesta del circuito de encendido y apagado al hacer doble pulso en S4 para encender y apagar el dispositivo.*

3.5. Curvas de carga y descarga de la batería

La batería usada en la versión anterior es la MIKROE-4473 MLP604060 de la figura 3.27 cuyo estado es desconocido. Las baterías de LiPo si son almacenadas completamente cargadas o descargadas producen una autodescarga que sucede como reacción química interna y funciona como circuito cerrado para la batería disminuyendo la vida útil de la misma y haciendo que la batería indique más capacidad de la que realmente tiene. Como se ve en la tabla 3.1, las baterías de LiPo pierden capacidad cada mes de desuso de entorno el 5 %.

El tiempo también produce en la batería mayor resistencia interna provocando una menor eficiencia ya que se pierde energía en forma de calor. Tambien la degradación química provoca que las reacciones que se dan en la carga y descarga pasen a ser reacciones inactivas reduciendo la capacidad y eficiencia.

Por lo tanto, se debe de analizar la batería haciendo un ciclo de carga y descarga. Para ello se pueden usar dos alternativas:

- Lecturas de tensión e intensidad mediante un analizador de potencia: Usando el Power Analyzer N6750A del laboratorio de GranaSAT podemos controlar la tensión de la batería conectando los polos de la batería con una de las fuentes del analizador. Dejando el dispositivo encendido podemos registrar estos valores como se ven en

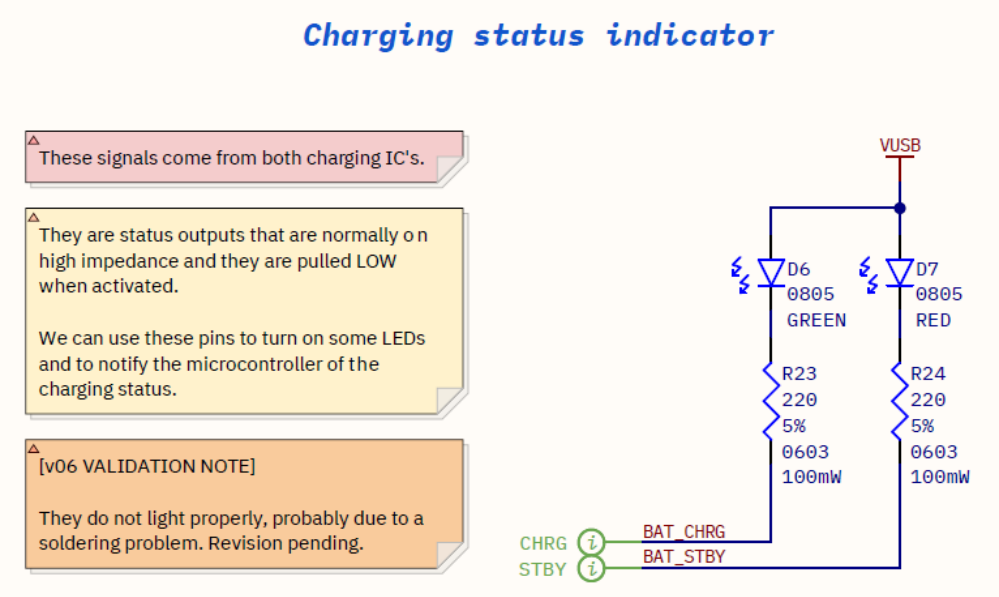
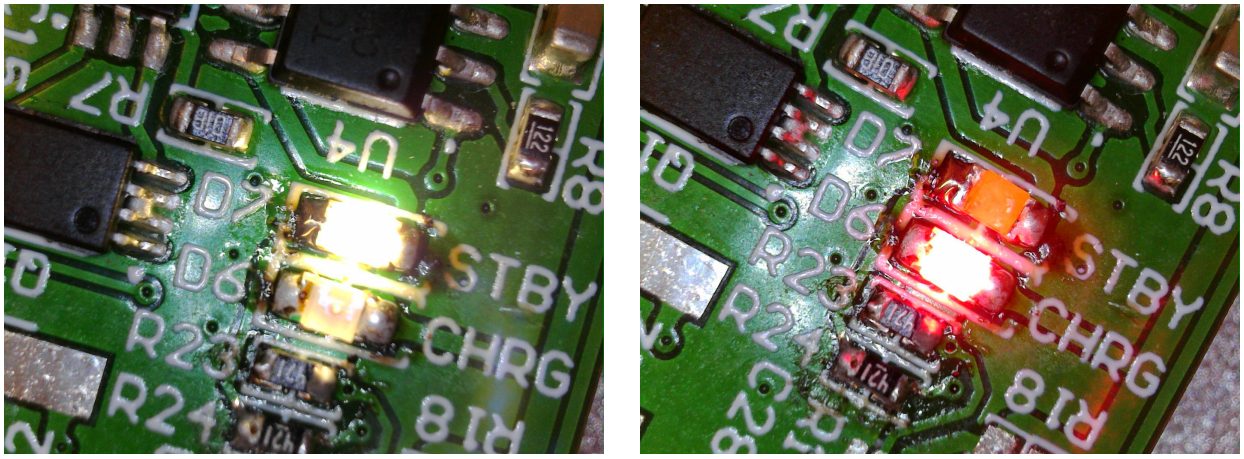


Figura 3.17 – *Luces LEDs de carga en la versión TIK Modelo 1. Página 8 de 21 del apartado de esquemáticos [6]*



(a) *Led de reposo.* (b) *Led de carga.*

Figura 3.18 – *LEDs de carga.*

la figura 3.28a. Asociando las resistencia de potencia 3.28b necesarias para una descarga de C/2 (750 mAh) con los 4.2V como máxima tensión ya que, mientras que con la descarga y la disminución de tensión la intensidad irá a menos asegurando que no hay descarga mayor de C/2.

- Cargador de baterías ALC8500: Con este equipo podemos cargar y descargar la batería a los C (carga de batería completa en una hora) deseadas.

$$R_{eq} = \left(\frac{1}{5\,\Omega + 5\,\Omega} + \frac{1}{4\,\Omega} \right)^{-1} = 2,8\Omega \tag{3.5.1}$$

Finalmnte, nos quedamos con la segunda opción ya que el Power Analyzer no funciona como un osciloscopio y sus terminales no muestran alta impedancia y por lo tanto, puede estar introduciendo cierta intensidad de demás

Figura 3.19 – *Pines de la versión anterior. Figura 6.20 [6]*

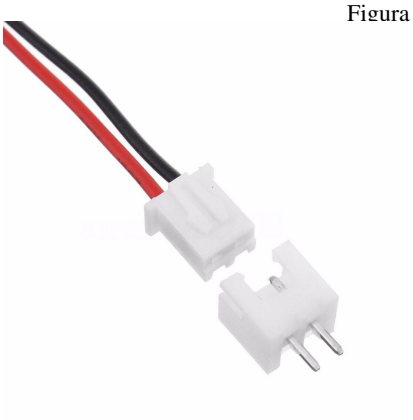


Figura 3.20 – *Conector adecuado JST HX 2.54mm y propuesto para el siguiente modelo.*

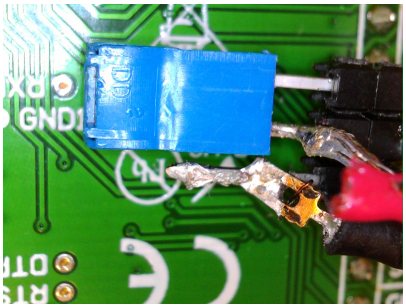


Figura 3.21 – *Alimentación mediante jumper.*

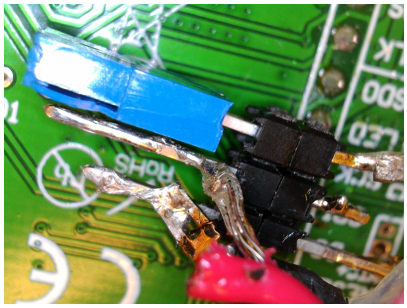


Figura 3.22 – *Alimentación retirada con desconexión de jumper.*

Tipo de batería	Recargable	Autodescarga típica o vida útil
Metal litio	No	10 años de vida útil
Alcalina	No	5 años de vida útil
Cinc-carbono	No	2-3 años de vida útil
Ion litio	Sí	2-3 % por mes; aprox. 4 % p. m.
Polímero de litio	Sí	~5 % por mes
Baterías NiMH de baja autodescarga	Sí	Hasta un 0,25 % por mes
Plomo-ácido	Sí	4-6 % por mes
Níquel-cadmio	Sí	15-20 % por mes
Níquel-metal hidruro (NiMH)	Sí	30 % por mes

Tabla 3.1 – *Autodescarga típica o vida útil de diferentes tipos de baterías [18].*

haciendo que nuestras medidas sean incorrectas. Nos quedamos la segunda opción que se encuentra en los laboratorio de GranaSAT.

El fabricante MIKROE indica que la rapidez de carga o descarga estándar es de 0.5C y máxima de 1C (página 4 de 15 [37]), nos quedamos con la carga y descarga estándar. El proceso de medición queda explicado en el apéndice

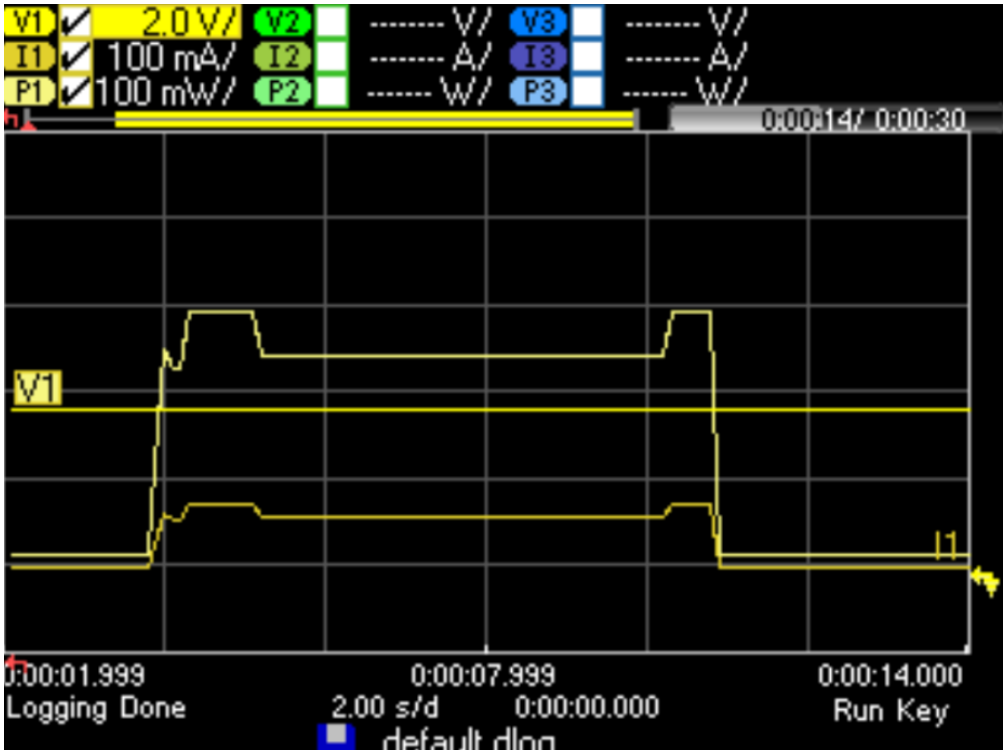
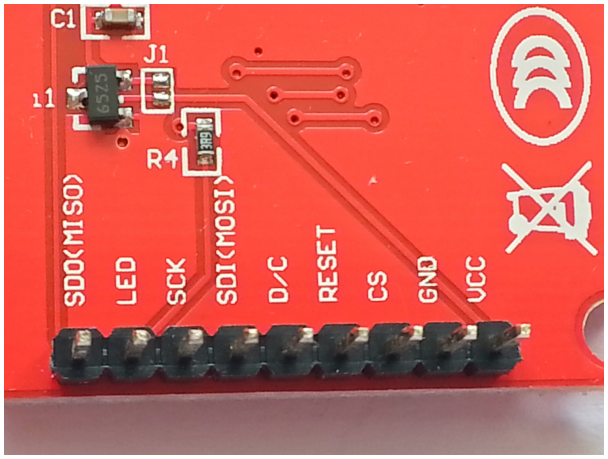
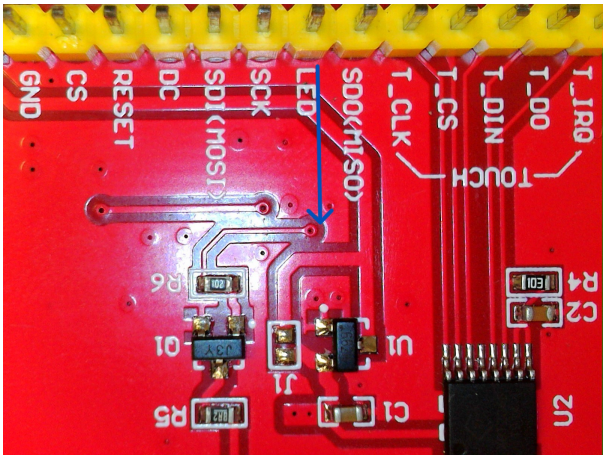


Figura 3.23 – *Enccendido y apagado simulando el consumo del dispositivo mediante un Power Analyzer N6705A (Ver sección de instrumentación).*



(a) *ILI9341 sin pin LED ni transistor de regulación del brillo.*

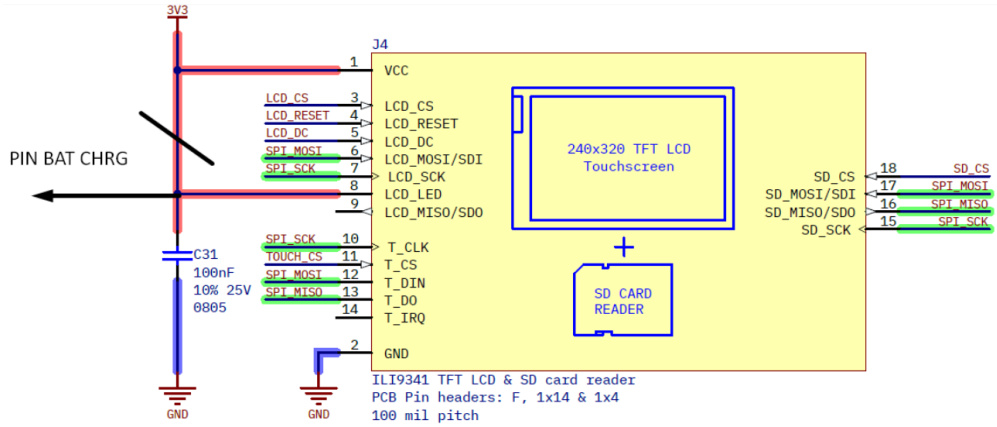


(b) *Ili9341 de el laboratorio de GranaSAT.*

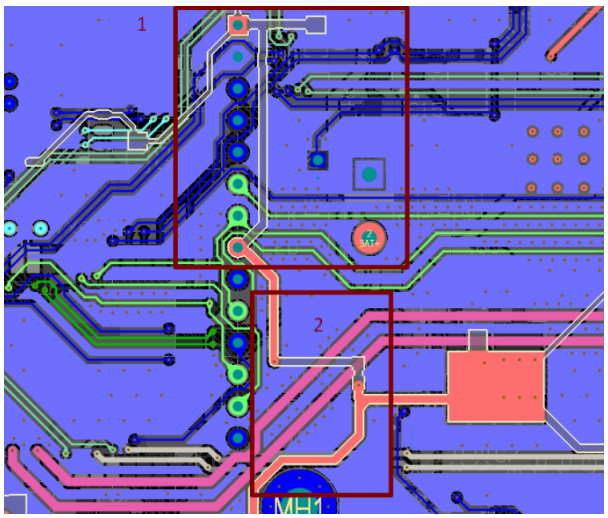
Figura 3.24 – *Parte trasera de módulo ILI9341.*

de instrumentación E. En las dos horas correspondientes a los C/2 podemos ver la capacidad de la batería así como las fases en las que una batería es cargada:

- Fase de precarga: Esta fase es esencial para proteger la batería de daños y asegurar que el proceso de carga se realice de manera segura y eficiente. La intensidad que entra es controlada y tiene como objetivo activar las celdas electroquímicas más apagadas. Una vez que la batería ha alcanzado un voltaje mínimo seguro, la carga continúa con la fase de corriente constante.



(a) *Esquemático de la versión anterior con el corte y conexión correspondiente para controlar el brillo.*



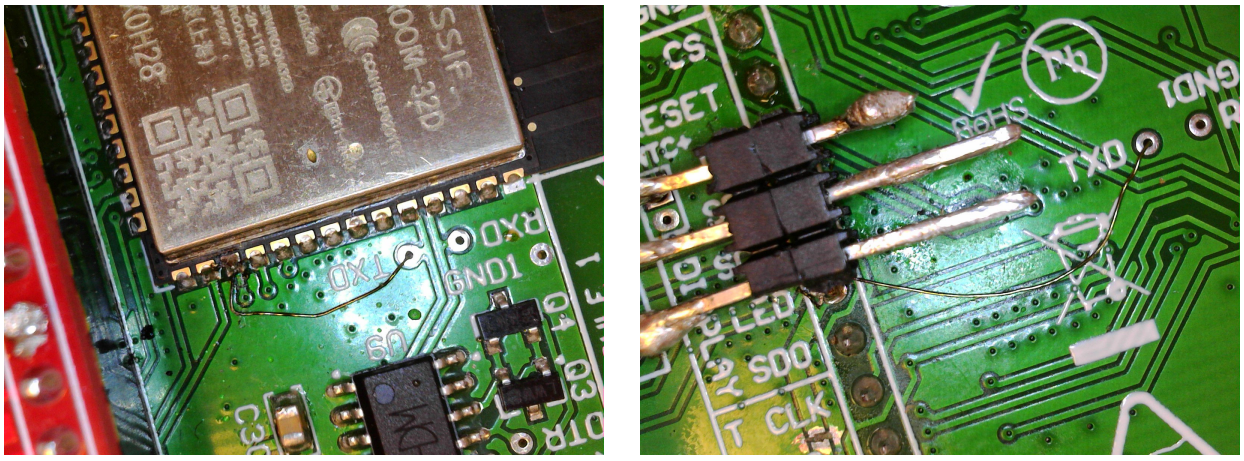
(b) *Pistas de PCB a cortar.*

Figura 3.25 – *Esquemático y pistas entre el pin LED y el pin VCC.*

- Fase corriente constante: La corriente se mantiene constante en esta fase (escogidos los C/2 con una batería de 1500 mAh, la intensidad que entra es de 750 mAh), donde la tensión aumenta gradualmente. Vemos como la intensidad se mantiene estable y por lo tanto es buen indicativo de que la carga se realiza correctamente.

- Fase tensión constante La tensión se mantiene constante una vez que la batería ha ancanzado su máximo. La intensidad va disminuyendo y asegurando la carga completa sin sobrecarga. Una vez que la intensidad disminuye considerablemente, el cargador desconecta la batería automáticamente. Otros cargadores mantienen el valor muy bajo alcanzando la energía máxima de la batería.

Como vemos el proceso de carga se realiza sin problema y sin niguna irregularidad. En otra batería sin marca y con capacidad menor he comprobado que hay picos de intensidad inesperados en los que el cargador ve que alguna celda está desbalanceada 3.30. La resistencia interna como se ha comentado varía con el tiempo y al empezar a realizar la carga esa resistencia cambia y produce ese cambio debido a un aumento repentino de la misma.



(a) *Conexión de hilo aislante conectado a un pin del microcontrolador en la cara top.*

(b) *Conexión de hilo al pin LED pasando a través de una vía de la PCB en la cara bottom.*

Figura 3.26 – *Modificación de pista para controlar el brillo de la pantalla.*

Figura 3.27 – *Batería MIKROE-4473 MLP604060 en el TIK Modelo 1.*

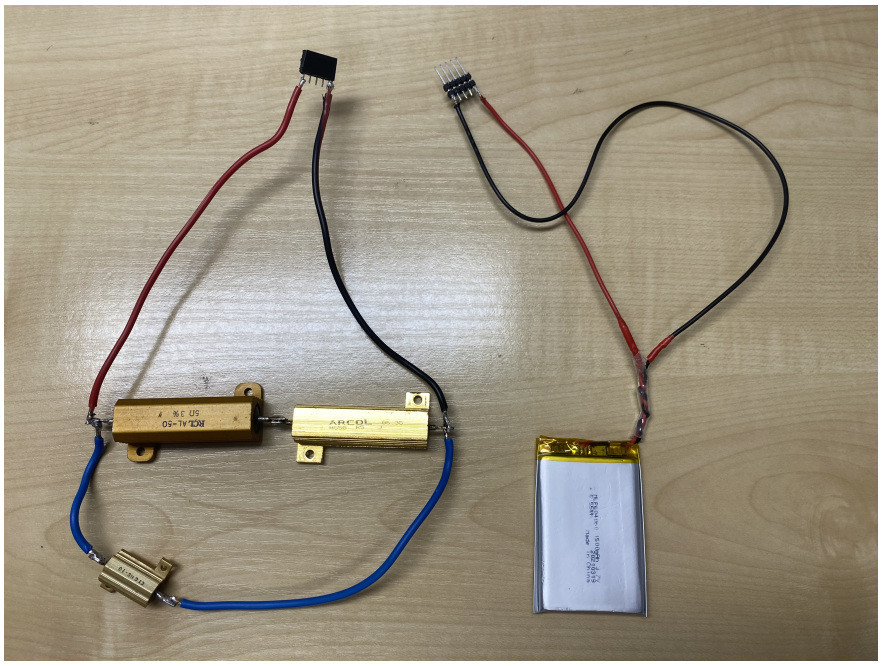
3.6. Comunicación SPI

La comunicación que usamos en este equipo para transmitir y recibir datos de los periféricos es el SPI. Este protocolo síncrono que funciona como Maestro-Esclavo siendo el microcontrolador el maestro y uno o varios periféricos los esclavos. Los componentes del protocolo son:

- MOSI (Master Out Slave In): Esta es la línea a través de la cual el maestro envía datos al esclavo. La comunicación es siempre iniciada por el maestro, controlando la dirección y el flujo de datos. Cuando el maestro tiene que enviar información al esclavo, lo hace a través de la línea MOSI. Esta línea es unidireccional, lo que significa que los datos solo fluyen del maestro al esclavo. Ejemplos típicos de datos enviados por MOSI pueden incluir comandos, direcciones de memoria, o información a ser procesada por el esclavo.
- MISO (Master In Slave Out): Contraria a la línea MOSI, la línea MISO es utilizada por el esclavo envíando

```
juanm@Juanma-msiPC MINGW64 ~/Desktop/TFG Juanma TIK/20241101_Agilent_N6705A
$ C:/python310/python.exe "c:/Users/juanm/Desktop/TFG Juanma TIK/20241101_Agilent_N6705A/main.py"
Connected Resources: ('USB0::0x0957::0x0F07::MY47000853::INSTR',)
Agilent Technologies,N6705A,MY47000853,D.01.09

        Measure0     Measure1     Measure2     Measure3     Measure4     Measure5     Measure6     Measure7     Measure8     Measure9
[V]    [3.302886]   [3.283366]   [3.281221]   [3.266121]   [3.26416]    [3.244427]   [3.228402]   [3.213822]   [3.186701]   [3.161433]
[I]  [-0.1329848] [-0.1330993] [-0.1329268] [-0.1330885] [-0.1329337] [-0.1328115] [-0.1329154] [-0.1329049] [-0.1330105] [-0.133055]
```

(a) *Medida de tensión e intensidad de consumo a medición por minuto.*



(b) *Asociación de potencias de 2.8 Ω.*

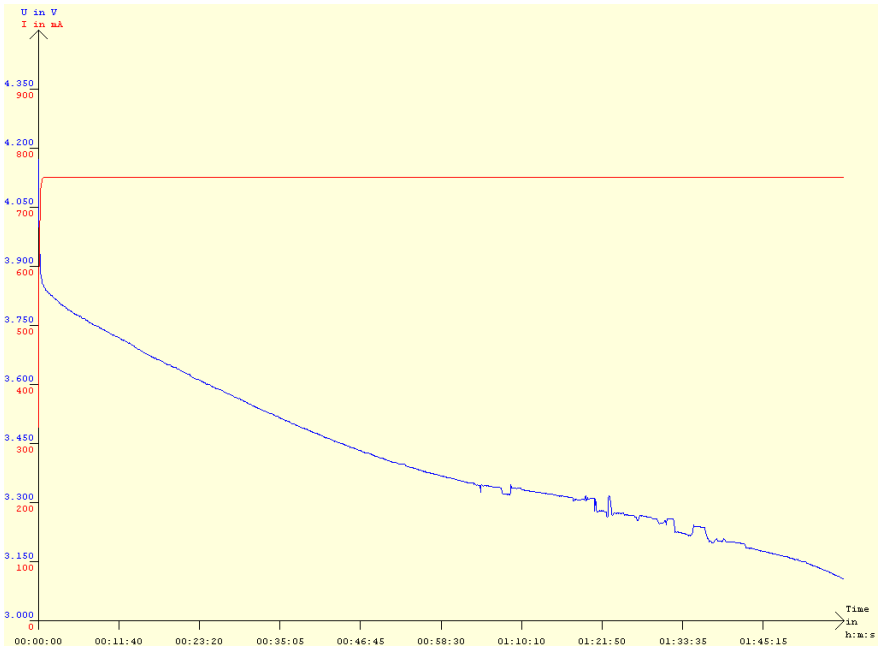
Figura 3.28 – *Medida de descarga de batería mediante Power Analyzer N6705A y asociación de resistencias de 2.8 Ω.*

datos de vuelta al maestro. Esta línea también es unidireccional, permitiendo que los datos circulen únicamente desde el esclavo hacia el maestro. Después de recibir un comando a través de MOSI, el esclavo puede responder con datos o resultados procesados a través de la línea MISO. Es importante destacar que solo un esclavo activo puede utilizar esta línea a la vez para evitar conflicto.

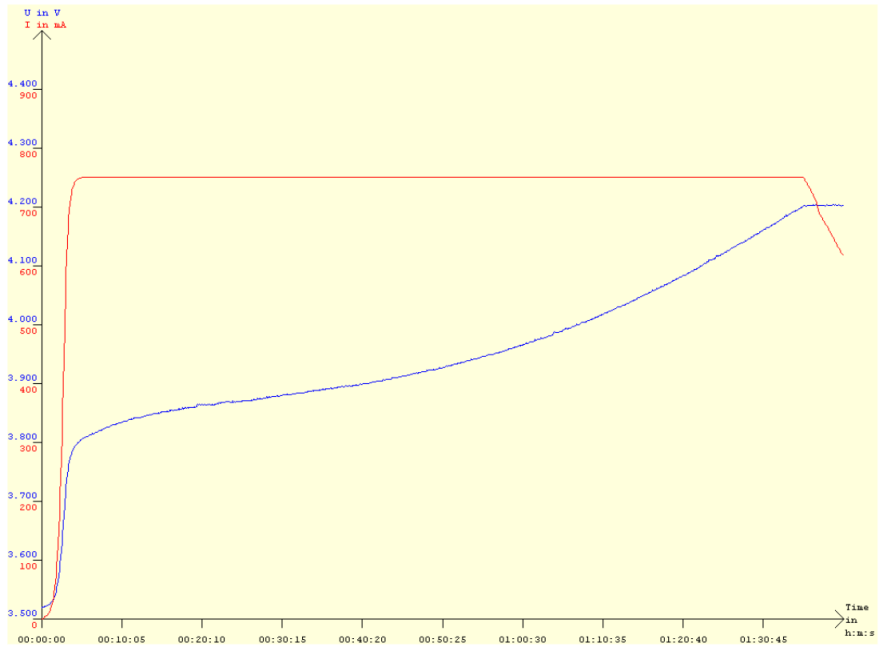
- SCLK (Serial Clock): La señal de reloj asegura que el maestro y el esclavo están sincronizados durante la transferencia de datos. El maestro genera la señal de reloj, y cada ciclo de reloj permite que un bit de datos se transfiera entre el maestro y el esclavo. Dependiendo de la configuración, los datos pueden ser capturados en el flanco ascendente o descendente de la señal de reloj. La frecuencia del reloj determina la velocidad de la comunicación, y debe ser suficientemente baja como para permitir que ambos dispositivos procesen los datos, pero lo necesaria para ser eficiente.

- SS (Slave Select): La señal de selección de esclavo es utilizada por el maestro para activar el esclavo con el cual desea comunicarse. En sistemas con múltiples esclavos, el maestro puede tener varias líneas SS y una para cada esclavo. Cuando la señal SS de un esclavo está en bajo, ese esclavo está activo y responde a las señales MOSI, MISO y SCLK. Los esclavos no seleccionados ignoran las señales del bus SPI, lo que permite que múltiples dispositivos compartan las líneas de datos sin interferencias.

En cada ciclo de reloj, se transmite un bit por las líneas MOSI y MISO además de ser líneas full duplex en la que los datos pueden viajar en las dos direcciones. Nosotros nos decantamos por este tipo de conmutación por su velocidad de comunicación mayor que la de I2C. Ver tabla de diferencias entre los protocolos de comunicación 3.2.



(a) *Descarga completa a C/2.*



(b) *Carga completa a C/2.*

Figura 3.29 – *Carga y descarga de la batería mediante el equipo ALC8500.*

En este protocolo hay hasta 4 formas de transferencia de datos donde se selecciona el modo de recoger la información. Depende de la polaridad y de la fase del reloj para iniciar la comunicación de datos 3.3 donde CPOL es (Clock Polarity) y CPHA (Clock Phase), polaridad y fase, respectivamente. El pin CS (Chip Select) es el que controla que periférico inicia la comunicación. Ver la figura 3.31.

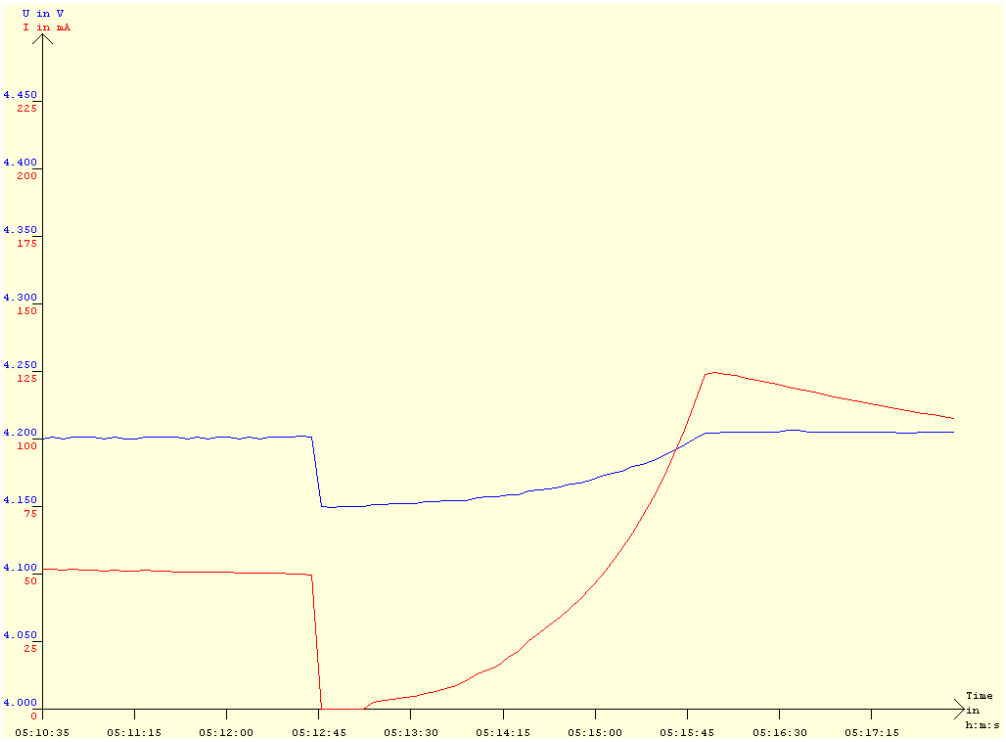


Figura 3.30 – *Cambio abrupto de intensidad y de tensiñon para balancear las celdas internas.*

Características	I2C	SPI
Líneas de comunicación	2 (SDA, SCL)	4 (MOSI, MISO, SCLK, SS)
Velocidad	Hasta 3.4 Mbps	Hasta 50 Mbps
Complejidad	Menor (requiere menos pines)	Mayor (más líneas de comunicación)
Número de dispositivos	Soporta múltiples dispositivos con direcciones únicas	Limitado por el número de líneas SS
Consumo de energía	Menor	Mayor
Rango de distancias	Mayor (debido a menor velocidad)	Menor (ideal para cortas distancias)
Sincronización	Basada en el reloj (SCL)	Basada en el reloj (SCLK)
Acknowledge (ACK)	Sí, para cada byte transferido	No
Full-duplex	No	Sí
Costo de implementación	Bajo	Medio

Tabla 3.2 – *Comparación entre protocolos I2C y SPI*

Tabla 3.3 – *Modos SPI con CPOL y CPHA*

Modo SPI	CPOL	CPHA	Polaridad del Reloj en Estado Inactivo	Fase del Reloj Utilizada para Muestrear y/o Desplazar los Datos
0	0	0	Nivel bajo	Datos muestreados en el flanco de subida y desplazados en el flanco de bajada
1	0	1	Nivel bajo	Datos muestreados en el flanco de bajada y desplazados en el flanco de subida
2	1	0	Nivel alto	Datos muestreados en el flanco de bajada y desplazados en el flanco de subida
3	1	1	Nivel alto	Datos muestreados en el flanco de subida y desplazados en el flanco de bajada

3.7. Velocidad de comunicación

Para nuestro dispositivo es crucial la frecuencia de transmisión de datos ya que necesitamos representar las señales al momento en la pantalla para que el usuario pueda visualizar las medidas que acaba de realizar. Por ello la decisión

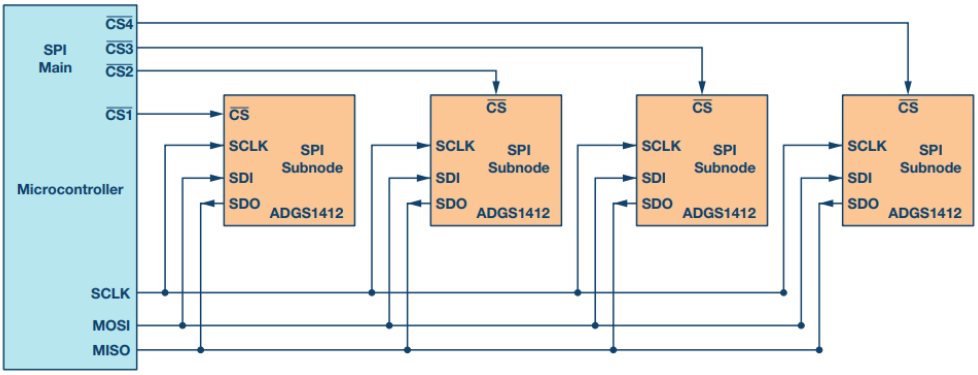


Figura 3.31 – *Conexión SPI con un maestro y tres periféricos [8].*

de usar la rapidez del SPI ya que nuestra transmisión recorre cortas distancias, de pocos centímetros.

Desde el equipo se demandó comprobar que la frecuencia de transmisión es suficiente comparada con la frecuencia de medida y lectura del ADC interno (revisar la memoria de Antonio Morales para más información). Desde Espressif nos comentan que podemos ajustar la frecuencia de reloj de la línea SCL como división entera de una frecuancia denominada APB donde SPI_CLKCNT_N y SPI_CLKDIV_PRE son bits de registro. La frecuencia APB es la frecuencia máxima que puede tomar el reloj del SPI y es de 80 MHz. La frecuencia serán valores divisores de números enteros entre la frecuencia nominal APB.

$$f_{\text{spi}} = \frac{f_{\text{apb}}}{(\text{SPI\_CLKCNT\_N} + 1)(\text{SPI\_CLKDIV\_PRE} + 1)} \tag{3.7.1}$$

El valor f_{SPI} es configurado fácilamente con *SPI.setFrequency ()*.Para comprobar el análisis de las señales digitales en un primer momento usé el osciloscopio HAMEG HMO1024 de 4 canales (revisar apéndice de instrumentacion E) donde coloqué cada canal en una de las líneas del bus SPI y así poder medir en forma Mode ROLL los cambios de las señales a tiempo real. Sin embargo, no fuí capaz de visualizar los cambios ya que, el mencionado osciloscopio no cuenta con $\frac{Div}{Segundo}$ sufientemente grande como para visualizar las señales con cambios a una frecuencia de MHz como se ve ne la figura 3.32.

Debemos usar otra alternativa para visualizar y asegurar que la frecuencia de reloj del SPI es la suficiente. En los laboratorios de GranaSAT cuentan con un analizador digital de señales digitales que tiene un máximo de 24MHz de frecuencia en señales a medir y 8 canales. Este dispositivo 3.34 se conecta a mediante un puerto USB A a nuestro ordenador y mediante un software específico 3.33a podemos visualizar las señales del bus.

La conexión la hice al bus SPI que controla la comunicación del módulo ILI9341 con la tarjeta SD. Después, soldé estos cables con termoretráctil a una cabecera de 4 pines hembra que conecté al analizador digital. Puesto que está conectado al bus que controla la comunicación con los datos de la tarjeta SD, cada vez que guarde la información, se iniciará el protocolo de comunicación pasando la señal del pin CS Chip Select de alto a bajo como se puede ver 3.32.Antes debemos de programar algunos parámetros mencionados anteriormente, como son los modos de comunicación, el número de bits de transferencia que hay en cada paquete. Estos ajustes lo hacemos desde el software de Saleae para indicarle que tipo de programación estamos utilizando en nuestro analizador 3.35.

Una vez configurado el modo de trabajo del protocolo SPI, podemos visualizar en la imagen 3.37 como se inicia el envio de información desde el maestro al esclavo en cuestión 3.25a donde a la derecha podemos ver los 4 pines correspondientes a la conexión del bus.

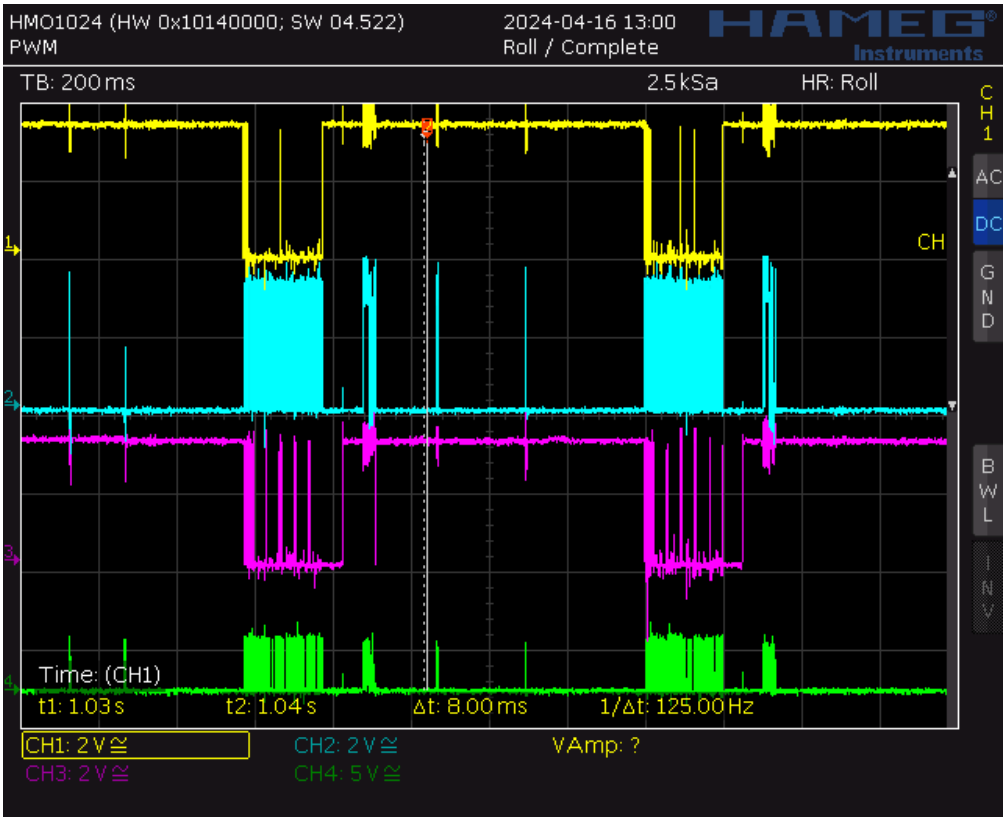


Figura 3.32 – *Visualización del bus SPI cuando se realizan dos guardados de datos en la SD. Por orden de visualización de arriba a abajo: CS, SCLK, MOSI y MISO.*

(a) *Logo del software Saleae Logic 2,4,14 OpenSource.*

(b) *Analizador digital de señales de 24 MHz y 8 canales.*

Figura 3.33 – *Software e instrumentación usada para analizar las señales del bus SPI.*

Finalmente, el control de frecuencia es complicado ya que el reloj no ajusta a la frecuencia indicada y está fija a un valor de 4 MHz. Buscando foros oficiales no comentan nada de este error asi que puede ser porque el ajuste no se realiza automático cuando indicamos con el comando *SPI.setFrequency ()* y se deban de ajustar los registros internos como se visualiza en la ecuación 3.7.1. En nuestro caso solo buscamos asegurar que la frecuencia de transmisión es igual o superior a la del ADC para poder representar los datos en pantalla de las mediciones sin retardos. Como vemos en la figura 3.37, esta frecuencia es de 4 MHz y suficiente para nuestro objetivo comentado.

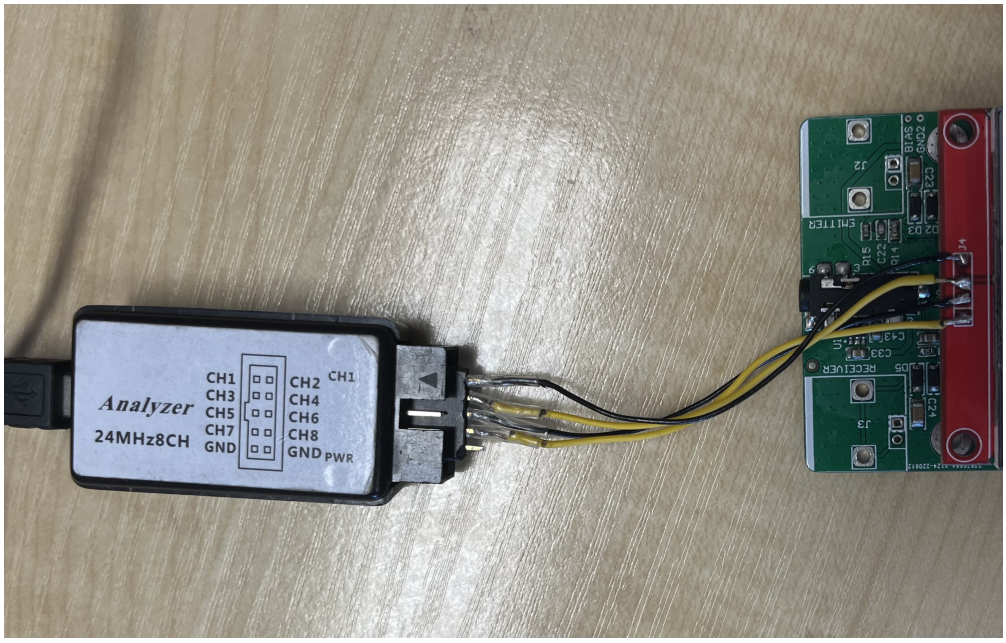


Figura 3.34 – *Conexión al analizador digital mediante la soldadura de 4 cables a un cabezal hembra y sus correspondientes termoretráctiles.*

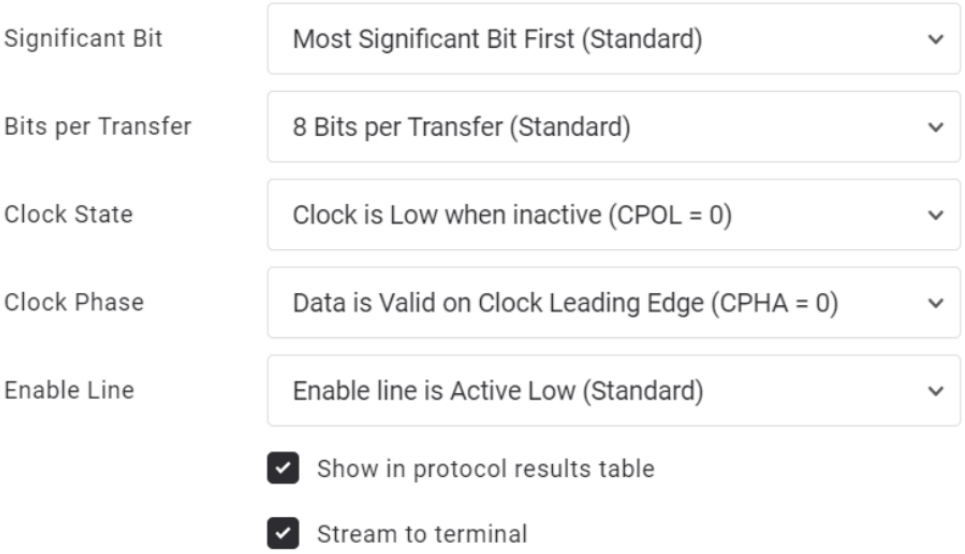


Figura 3.35 – *Ajustes para el modo de trabajo de nuestro bus SPI.*

Si no realizamos esta comprobacción podría suceder que leeríamos los datos más rápido de lo que somos capaces de representarlos, el usuario vería la pantalla sin resultados hasta pasado un tiempo que dibujaría poco a poco la señal adquirida. Esto no puede suceder en nuestro dispositivo ya que necesitamos representación en tiempo real y para ello está diseñado el software (RTOS).

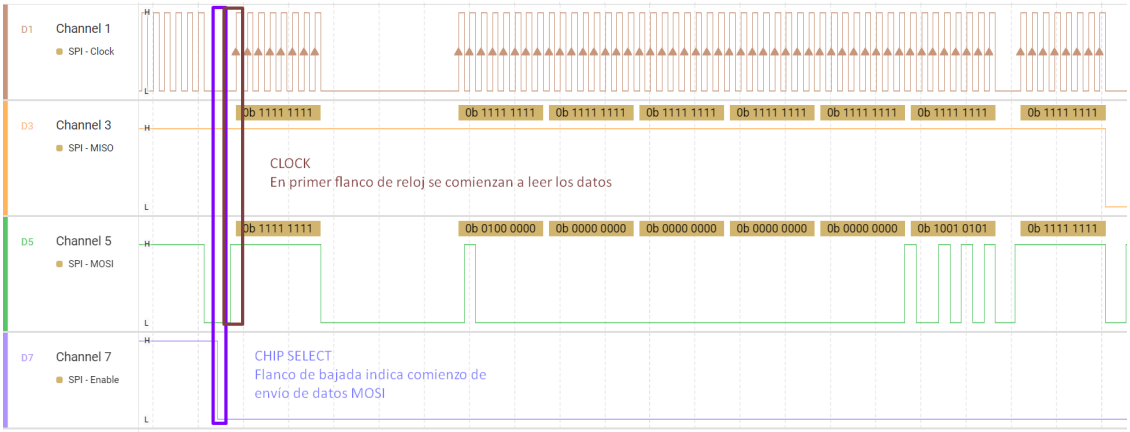


Figura 3.36 – *Inicio de comunicación del protocolo SPI.*

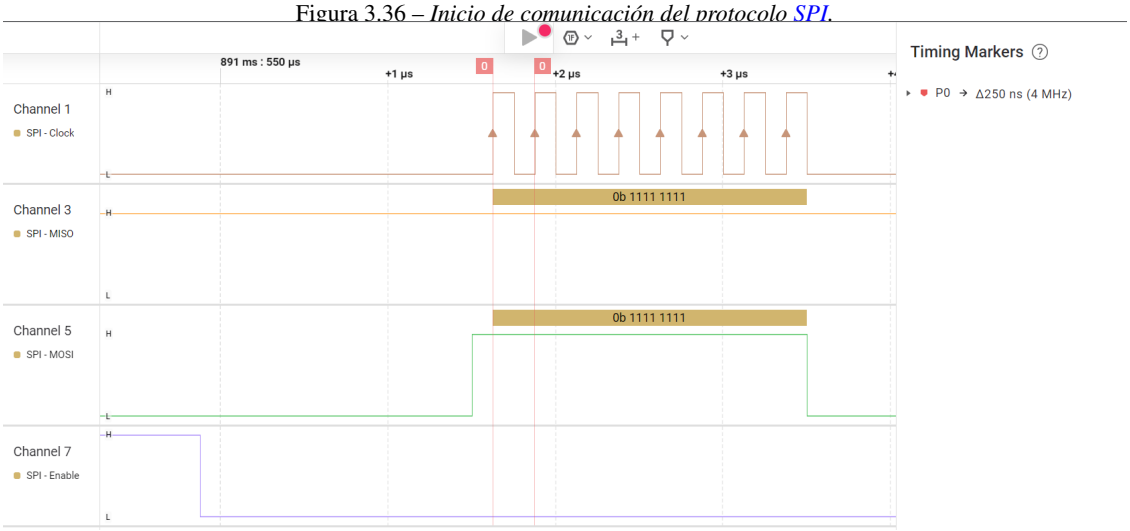


Figura 3.37 – *Frecuencia del reloj del bus SPI.*

3.8. Gestión de la batería

En esta sección detallamos como se ha desarrollado e implentado un sistema de monitorización tanto de la temperatura como de la tensión de la batería del equipo. En la anterior version [6](sección 5.1.2.3.3) se añadió un integrado como es el INA219, IC desarrollado por *National Instruments* el cual es conectado directamente en paralelo con la resistencia de shunt y detalla la corriente y potencia máxima disipada a través de la R_{shunt}, además de la resolución del integrado para nuestra aplicación que por defecto funciona con 12 bits.

Para obtener medidas precisas necesitamos calibrar el integrado y asegurarnos de que las lecturas son correctas. Para ello necesitamos asegurarnos de que la librería usada Adafruit INA219 está calibrada para nuestra resistencia de shunt. Sin embargo, no es el caso ya que está pensada para usarla con 100Ω, y nuestro circuito utiliza 200Ω. Por lo tanto la he manipulado para obtener los valores más precisos (más detalle en el capítulo 4).

Para llevar a cabo la calibración he necesitado dos equipos(consultar apéndice de instrumentación ??):SDM3065X multímetro digital de 6 1/2 dígitos conectada a la resistencia de shunt, cuyo valor conozco con exactitud ya que ha sido medida también con el mencionado multímetro 220 fuente de corriente programable de Keithley, con este equipo podemos limitar la tensión máxima que puede alcanzar el dispositivo (al igual que una fuente de tensión con la corriente si hay un cortocircuito).

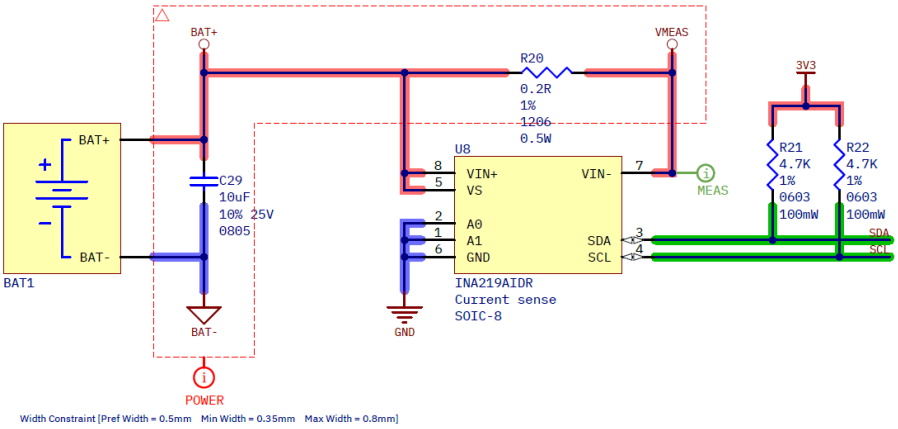


Figure 5.7 – *INA219 current sense circuit for battery current monitoring.*

3

Figura 3.38 – *INA219 Current monitoring circuit version 0.6*

El objetivo por tanto, es conectar la fuente de corriente en paralelo con la resistencia de shunt y con el multímetro de 6 dígitos también conectado buscamos hacer que los valores de lectura del INA por la terminal sean prácticamente los mismos. También se han de alimentar los buses de 3.3V que alimentan los integrados necesarios para hacer tal operación y ello es realizado con otra fuente externa. En la figura3.39 podemos ver el circuito montado.

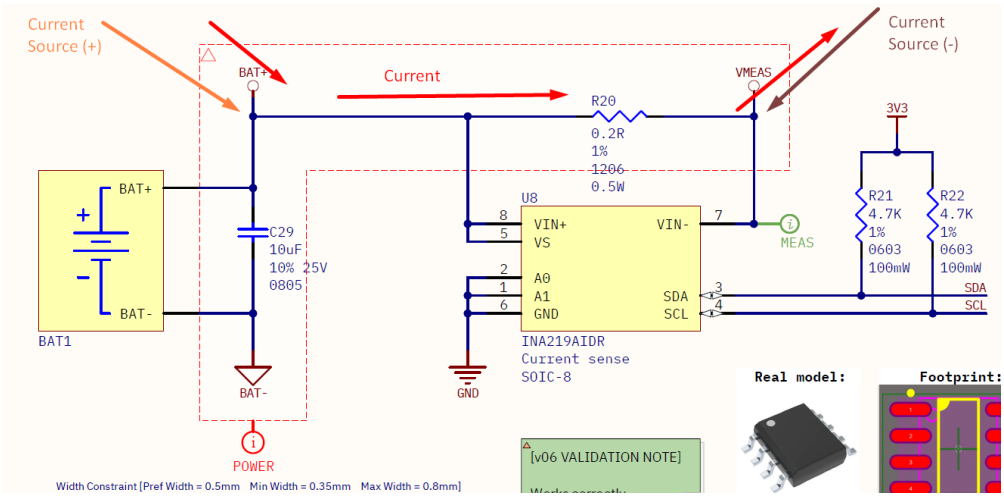


Figura 3.39 – *Conexión de la fuente de corriente*

Uno de los problemas que surgió al hacer este conexionado fué que, por un lado al estar conectado el USB que transfiere los datos al microccontrolador y alimenta el dispositivo y por otro lado la fuente de alimentación, se estaba provocando una incongruencia circuital al estar el cargador de la batería recibiendo alimentación por la entrada con el USB e intentado alimentar la fuente de alimentación que está conectada a la salida 3.40.

Puesto que la alimentación Vmeas corresponde a la alimentación necesaria para el microcontrolador y demás integrados (entre ellos el INA219) debemos de cortar la alimentación de el puerto USB. La forma más sencilla que he encontrado para cortar la pista de alimentación es quitando la perla de ferrita (ver figura 3.41b) que incluye el puerto de entrada de la imagen 3.41a.

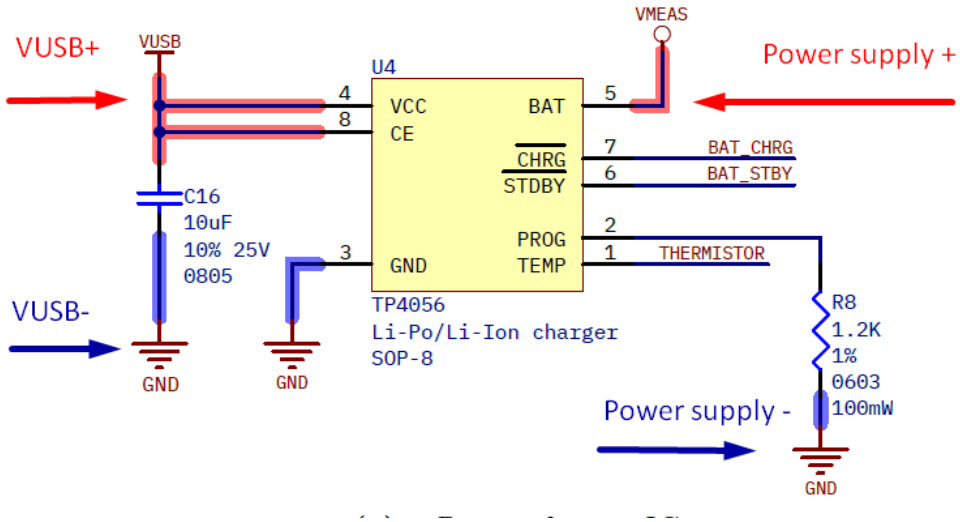
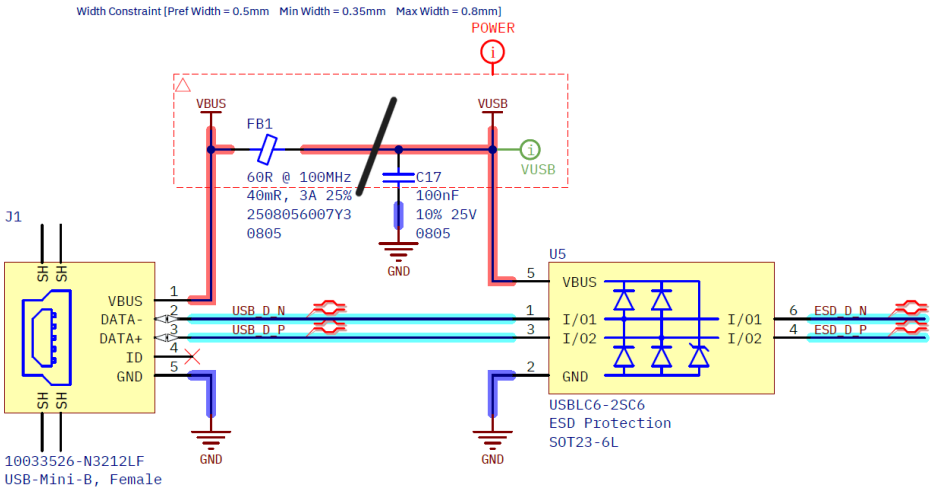


Figura 3.40 – *Cargador de batería de LiPo recibiendo alimentación por la salida Vmeas*

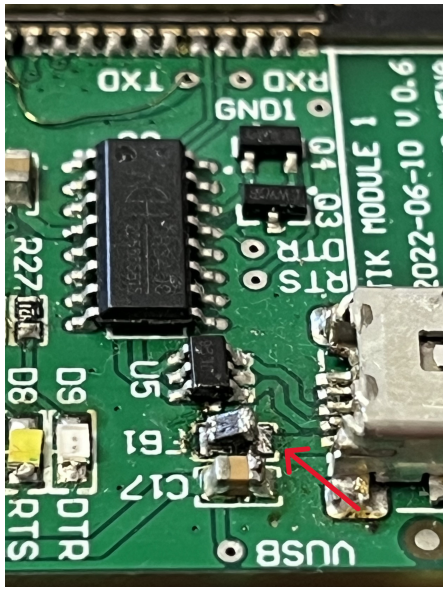
Tabla 3.4 – *Comparación de mediciones con y sin fuente de corriente.*

	Con Fuente de Corriente	Sin Fuente de Corriente
Multímetro Digital Vshunt	9,9180 mV	29,8359 mV
I INA219	57,2 mA hasta 59,10 mA	157,8 mA hasta 160,05 mA
I Real	49,59 mA	149,59 mA

Como vemos en la tabla 3.4, los valores recibidos del INA219 frente a los reales obtenidos con el multímetro se diferencian con un error de entre el 7 % y el 5.2 %. Para ajustarlo aún más, necesitamos conocer también el valor de la resistencia de shunt con un error del 1 %. Tenemos $0,2283\,\Omega$, $0,2302\,\Omega$ y $0,2297\,\Omega$, un error de $0{,}6\,\%$ entre el valor más alto y más bajo, por lo que nos quedamos con el valor medio de $0,2303\,\Omega$ y lo incluimos en el ajuste de la librería INA usada, que será detallada más adelante en el capítulo 4 sección 5.2.2.



(a) *Esquemático del puesto USB de la versión anterior [6].*



(b) *Corte de pista mediante la perla de ferrita.*

Figura 3.41 – *Esquemático del puertos USB de entrada y corte de alimentación en la perrla de ferrita.*

Capítulo 4

Especificación del sistema

Las necesidades para desarrollar los objetivos del mencionado proyecto están bien recogidas en el capítulo 4 System specification [6], donde se relatan todas las necesidades como sistema en cuanto a requerimientos al ser un producto de ingeniería, y en concreto las necesidades de este producto electrónico como sistema embebido a tiempo real. Se definen y explican conceptos básicos en cuanto a la definición de PCB, tecnología de soldadura, número de capas conductivas y aislamiento entre sí.

Todos los circuitos no testeados e implementados en la versión anterior han sido probados en esta versión y no necesitan la mayoría modificaciones. Además toda la discusión de componentes a elegir y nomenclatura de los mismos están detallados en la sección 4.2 y en este documento se explicarán todas las modificaciones y componentes añadidos que no se incluyen en el TIK Model 1, así como el por qué no realizar algunas mejoras.

4.1. Arquitectura electrónica

4.1.1. Microprocesador

El microprocesador es la piedra angular de todo sistema embebido y en especial en un equipo portable como este donde la eficiencia y el número de puertos I/O son esenciales. Puesto que el desarrollo de este dispositivo está realizado por estudiantes de úlltimo año y no por profesionales en sistemas embebidos necesitamos un microprocesador SoC que cuente con todo tipo de posibilidades de conectividad: Wi-Fi y Bluetooth. Además, varios núcleos para trabajar en paralelo con tareas RTOS, memorias RAM y ROM así como protocolos de comunicación I2C y SPI.

Tabla 4.1 – *Comparativa de Módulos SoC*

Característica	ESP32-WROOM-32D	ESP32-WROOM-32E	ESP32-S3	STM32F411	ESP8266	NRF52840
Procesador	Dual-core Xtensa LX6	Dual-core Xtensa LX6	Dual-core Xtensa LX7	ARM Cortex-M4	Tensilica L106	ARM Cortex-M4
Frecuencia de CPU	Hasta 240 MHz	Hasta 240 MHz	Hasta 240 MHz	Hasta 100 MHz	80 MHz	64 MHz
Memoria Flash	4MB	4MB	8MB	512KB	4MB	1MB
RAM	520KB SRAM	520KB SRAM	512KB SRAM	128KB SRAM	160KB	256KB
Conectividad Wi-Fi	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n	No Wi-Fi	802.11 b/g/n	No Wi-Fi
Bluetooth	v4.2 BR/EDR y BLE	v4.2 BR/EDR y BLE	v5.0 BLE	No	No	v5.0 BLE
GPIO	36 pines	36 pines	44 pines	37 pines	17 pines	48 pines
Voltaje de operación	2.3V - 3.6V	2.3V - 3.6V	2.3V - 3.6V	1.7V - 3.6V	2.5V - 3.6V	1.7V - 3.6V
Consumo (W)	0.8W (Típico)	0.8W (Típico)	0.9W (Típico)	0.1W (Típico)	0.3W (Típico)	0.2W (Típico)
Interfaz SPI/I2C/UART	Sí	Sí	Sí	Sí	Sí	Sí
Modos de ahorro de energía	Light Sleep, Deep Sleep	Light Sleep, Deep Sleep	Light Sleep, Deep Sleep	Low-power Sleep	Modem Sleep	Various
Encapsulado	38-Pin QFN, 49-Pin QFN, 48-Pin QFN, 64-Pin QFN	38-Pin QFN, 49-Pin QFN, 48-Pin QFN, 64-Pin QFN	38-Pin QFN, 49-Pin QFN, 48-Pin QFN, 64-Pin QFN	LQFP-48, LQFP-64, WLCSP-49, QFN-32	32-Pin QFN, 16-Pin SOIC-8, QFN-24	48-Pin QFN QFN-48, QFN-64
Diferencias clave	Mayor disponibilidad en ciertas aplicaciones	Más reciente, mejoras en la estabilidad	Mayor capacidad de procesamiento, BLE 5.0	Soporte de baja energía, más pines	Más económico, menos potente	Mayor seguridad y conectividad BLE

En la tabla 4.1 comparamos las principales alternativas barajadas. Los SoC que tengan menos de dos núcleos son desechados ya que no cumplen los requerimientos para hacer un sistema RTOS donde necesitamos varias tareas ejecutándose a la vez. Finalmente, nos quedamos con las opciones que tengan más memoria RAM debido a que las tareas a manejar son pesadas para la pila de memoria, cuanto más espacio tengamos más funcionalidades podemos añadir a las tareas en curso.

Con todo ello, nos quedamos con las opciones ESP32 WROOM 32D O ESP32 WROOM 32E. El primer caso, es el ya implementado en la primera versión de TIK y funciona sin problemas ya que es amplimente usado por Espressif en placas de desarrollo. Sin embargo, puesto que hay que actualizar el equipo al rápido avance del mercado de componentes, hemos optado por el segundo caso que es practicamente el mismo microprocesador pero con mayor estabilidad y mejoras internas de robustez.

Figura 4.1 – *ESP32-WROOM-32D, 32E Y S3 con encapsulado QFN*

Figura 4.2 – *STM32F411 con encapsulado LQFP*

Figura 4.3 – *ESP8266 con encapsulado QFN*

Figura 4.4 – *NRF82840 con encapsulado QFN*

4.1.2. Batería LiPo

La batería usada en TIK Modelo 1 es la MIKROE-4473 de 3.7V y 1500 mAh figura 3.27 [38]. El aumento de autonomía es posible ya que las dimensiones de la PCB necesita de un diseño mecánico que puede albergar a una batería mayor que la actual. Por otro lado, las baterias planas de LiPo tienen la posibilidad de contener 3 cables: 2 para la polaridad de la batería y otro conectado a un divisor de tensión interno para el control de la batería.

La nueva batería escogida EASYLANDER 803450 4.5 [39] tiene por muy poco mayor ancho, lo que nos proporciona 300 mAh de más respecto a la anterior versión como se puede ver en la tabla 4.2.

Figura 4.5 – *EASYLANDER 803450 en TIK Modelo 2*

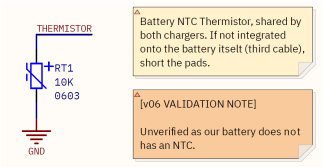


Figura 4.6 – *Conexión NTC de batería. Creditos [6] (Esquematicos 8 de 21)*

Tabla 4.2 – *Comparación de Especificaciones de Baterías*

Especificaciones	Modelo 803450	MIKROE-4473
Tipo	Batería de litio	Li-Polymer
Capacidad	1800 mAh	1500 mAh
Recargable	Sí	Sí
Voltaje de carga	3.7 V 4.2 V	3.7 V 4.2 V
Voltaje nominal	3.7 V	3.7 V
Temperatura de trabajo	-10 °C 50 °C	-20 °C 60 °C
Dimensiones (L * W * H)	50x34x8 mm	50x34x6.5 mm
Peso	No especificado	25 g

4.1.3. Puerto USB

El puerto USB nos proporciona alimentación a la batería y además, transfiere los datos al equipo de forma diferencial. Tenemos que tener en cuanta que es un componte que sufre de tensión mecánica cuando se conecta y desconecta el puerto.

Desde la UE se demanda unificación en el puerto USB para hacer evitar el gran número de cargadores que son extraviados por los usuarios y/o obsoletos por los fabricantes de sus equipos. La solución a este dilema de contaminación electrónica medioambiental se puso fin con la regulación del 28 de diciembre de 2022 donde se ordenó a las compañias tecnológicas que tenían que hacer efectivo el cambio de tipo de puerto a tipo C con fecha máxima en el fin de 2024. Esta unificación también conlleva que para abril de 2026 todos los portátiles también tienen que tener este tipo C [40].

Los USB tipo C no dependen de orientación entre hembra y macho lo que los hace muy prácticos. Esto se consigue haciendo que cada pin del USB macho sea el espejo del hembra como se puede ver en la imagen 4.9. En total el número de pines es de 24 si buscamos usar todos los recursos que nos ofrece.

- VBUS (Pines A4, A9, B4, B9): Proporciona alimentación al dispositivo conectado, normalmente a 5V, pero puede variar hasta 20V. Para nuestro caso la lalimentación será de 5V a 1A pudiendo llegar a 3A y soportar 15W.
- GND (Pines A1, A12, B1, B12): Pines de masa para las corrientes de retornos de la PCB.
- D+ / D- (Pines A6, A7, B6, B7): Líneas de datos USB 2.0. Estas líneas diferenciales son las que fundamentan

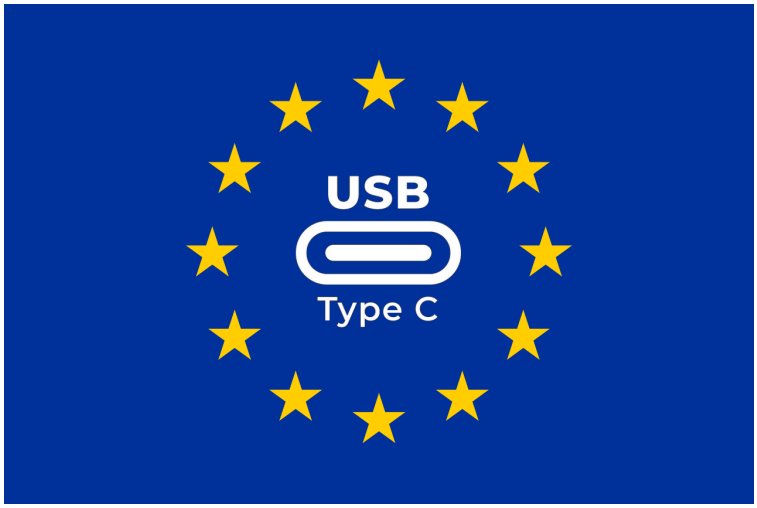
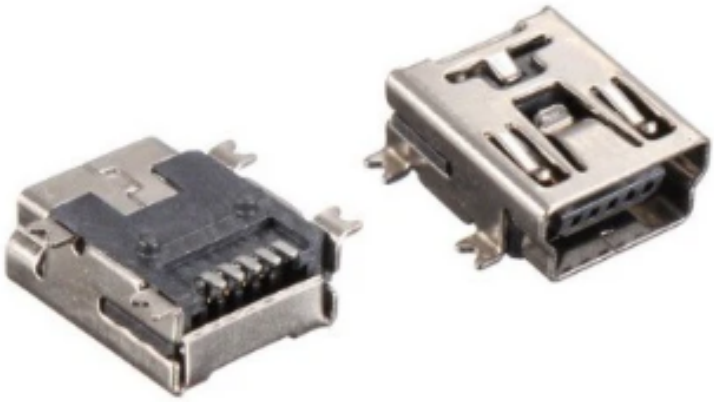


Figura 4.7 – *Propuesta de la UE por el USB tipo C.*



(a) *USB mini B usado en la anterior versión.*

(b) *USB tipo C 4085 GCT usado en la actual versión.*

Figura 4.8 – *USB usados en TIK modelo 1 y 2.*

el paso de información.

- TX1+ / TX1- (Pines A2, A3) y RX1+ / RX1- (Pines B10, B11): Pares diferenciales para la transmisión de datos SuperSpeed en USB 3.0.

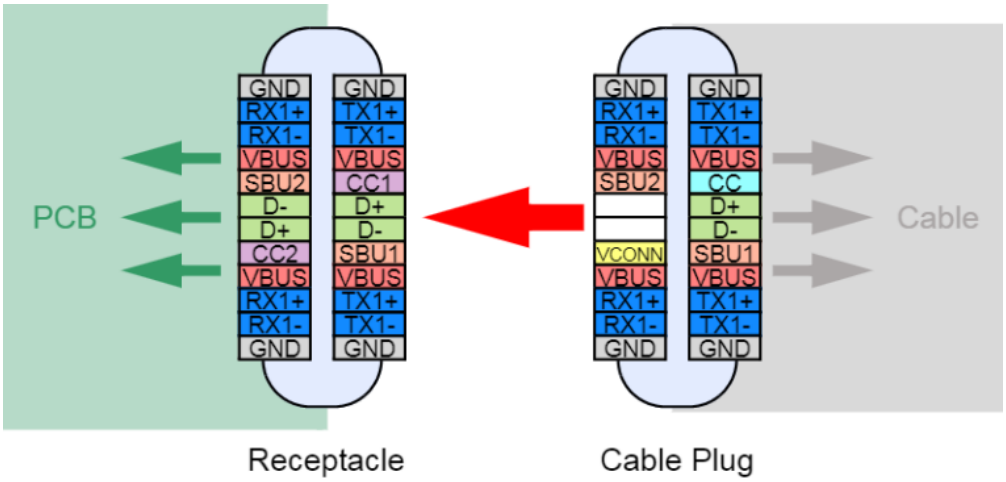


Figura 4.9 – *Macho y hembra tipo C. [9]*

- TX2+ / TX2- (Pines B3, B2) y RX2+ / RX2- (Pines A11, A10): Pares diferenciales adicionales para datos SuperSpeed, compatibles con la operación de doble canal.
- CC1 / CC2 (Pines A5, B5): Pines del Canal de Configuración utilizados para determinar la orientación del cable.
- SBU1 / SBU2 (Pines A8, B8): Pines de uso lateral para modos alternativos como DisplayPort o Thunderbolt.

La tecnología superspeed corresponde a USB 3.0 en adelante y permite alta velocidad de rtnasmisión de datos llegando al orden de los gigabytes por segundo. Para ello necesitamos electrónica extra que no es necesaria. Con la operación de doble canal podemos enviar datos simultáneamente en dos pares de canales TX1+/TX1- y RX1+/RX1- como los pares TX2+/TX2- y RX2+/RX2. Esta tecnología de doble canal es la que consigue la función de DisplayPort (transmisión de video y audio) y Thunderbolt (transferencia de datos a alta velocidad). Ya que no necesitamos todo ello, nos basta con USB 2.0, que conlleva 16 pines (si buscasemos solo alimentación nos bastaría con 6 pines). Por otro lado, pensando en el proceso de soldadura, 16 pines a mano sería complicado en tecnología SMD y como existe la opción THT, pues esta última es la opción aceptada con el USB GCT 4085 que se ve en la imagen 4.8b.

4.1.4. RTC (Reloj a tiempo real RTC)

El dispositivo no contaba con la fecha y hora, el integrado ideal es el denominado RTC. Este integrado nos permite tener un control horario aunque el dispositivo permanezca apagado ya que no consume intensidad de la batería principal. Además esto nos aporta un recurso extra a la hora del firmware ya que podemos programar alguna funcionalidad en base a los datos que nos puede aportar. Por ejemplo, una alarma a cierta hora o un recordatorio en cualquier día del año.

En el mercado ya hay integrados de este tipo integrados en módulos como por ejemplo DS1307, uno de los RTC más populares. Este módulo utiliza un cristal de cuarzo para mantener la hora y la fecha, y es compatible con el protocolo I2C para la comunicación con microcontroladores. Incluye una batería (pila de moneda), permitiendo mantener el tiempo incluso cuando el dispositivo principal está apagado. Como podemos ver en 4.10, este tipo de módulos son muy voluminosos para nuestro dispositivo donde el tamaño es un compromiso para un dispositivo manejable y móvil. Los módulos están también desfasados ya que son enfocados para proyectos de arduino simples.

Por lo tanto, se decide integrarlo en la PCB, que necesita del integrado, un cristal de cuarzo que oscila y permite asociar un número de ooscilaciones a un segundo, minuto ... etc. Además se necesita una batería la cual permita ser reemplezada fácilmente una vez que sea agotada.

Los RTC del laboratorio de GranaSAT están algo desfasados (M41T00, MCP7940N y PCF8563T) por lo que se

Figura 4.10 – *Módulo RTC DS1302.*

necesita comprar uno que esté acorde con la eficiencia actual del mercado.NXP vende gran variedad de ellos y son los más eficientes. La duda está entre PCF8523 que es el que menor potencia consume y por otro lado el 85363A o 85363B (depende de el protocolo de comunicación I2C acaba en A y SPI acaba en B). Una buena opción es el 85363A que es el que más funcionalidades tiene (2 alarmas independientes además del control horario), la potencia disipada es la misma y tiene menor consumo. El amplio rango de voltaje de operación del PCF85363A acompañado de una gran precisión y bajo consumo lo hace ideal para dispositivos móviles y es por tanto, la opción escogida (ver 4.3).

Características	PCF85363A	DS3231	MCP7940N	PCF8563T
Fabricante	NXP	Maxim Integrated	Microchip	NXP
Comunicación	I2C	I2C	I2C	I2C
Voltaje de Operación	0.9V - 5.5V	2.3V - 5.5V	1.8V - 5.5V	1.8V - 5.5V
Consumo en Standby	0.28 µA @ 3.0V	3.5 µA @ 3.3V	1.2 µA @ 3.3V	0.25 µA @ 3.0V
Precisión	±3ppm	±2ppm	±5ppm	±10ppm
Batería de Respaldo	Sí	Sí	Sí	Sí
Alarmas Programables	2	2	2	2
Registro de Tiempo de Encendido/Apagado	No	Sí	Sí	No
Interrupciones	Sí	Sí	Sí	Sí
Rango de Temperatura	-40°C a 85°C	-40°C a 85°C	-40°C a 85°C	-40°C a 85°C
Tamaño del Encapsulado	MSOP - 8	SOIC-16, TSSOP-16	SOIC-8, MSOP-8	SOIC-8, TSSOP-8
Aplicaciones Principales	Wearables, IoT	Electrónica de Precisión	Monitoreo de Energía	Electrónica general

Tabla 4.3 – *Comparación de RTCs: PCF85363A, DS3231, MCP7940N, y PCF8563T*

El RTC va acompañado de un oscilador como se ha comentado y de una batería, en este caso una batería primaria desechable. Hay distintas opciones de portapilas:

- Portapilas: Esta opción permite colocar y quitar la pila para reemplazarla. Esta es la opción más cómoda y se puede acceder a la pila de botón desde el exterior con la tapa pertinente en la carcasa como se ve en 4.11 y 4.12.
- Pila soldada: Por otro lado, la pila soldada a la PCB no permite cambiar la pila fácilmente si nos encontramos en el exterior sin los medios necesarios para ello, por lo tanto desechamos esta opción 4.13.

La opción escogida es la que podemos ver en la figura 4.12 por ser de menos tamaño que la que se ve en la figura 4.11. Pero, este menor tamaño conlleva una pila de botón con menor capacidad. La pila tiene 3V y 40 mAh y podemos

Figura 4.11 – *Portapilas CR2032.*

Figura 4.12 – *Portapilas CR1220.*

Figura 4.13 – *Pila CR1220 con pads de soldadura en cada cara.*

ver el tiempo de funcionamiento del RTC escogido en la ecuación 4.1.4. Con valores típicos de consumo vemos que el reemplazo de pila de botón se preveé a largo plazo.

$$\text{Autonomía} = \frac{\text{Capacidad}}{\text{Consumo típico}} \tag{4.1.1}$$

$$\text{Autonomía} = \frac{40\,\text{mAh}}{0{,}28\,\mu\text{A}} = 142857{,}1\,\text{Horas} \tag{4.1.2}$$

$$\text{Años de autonomía} = \frac{\text{Autonomía}}{\text{Horas de un año}} = \frac{142857{,}1}{24 \times 365} = 16{,}3\,\text{años} \tag{4.1.3}$$

$$\tag{4.1.4}$$

4.1.5. Protección ESD

Actualmente, el equipo cuenta con protección ESD en el puerto USB hembra para proteger las lineas de datos diferenciales que comunican al equipo con el exterior como se ve en la figura 4.14.

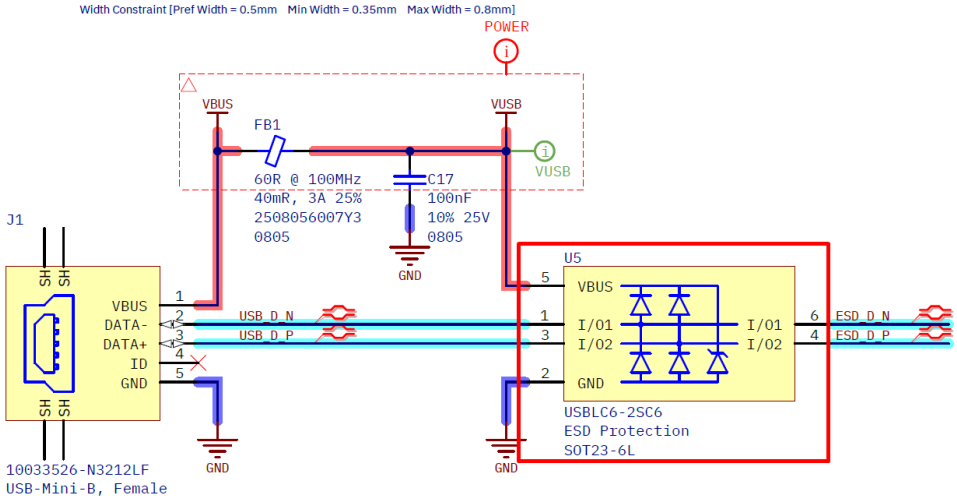


Figura 4.14 – *Protección ESD en la entrada USB. Créditos [6]*

Hay otras líneas de transmisión de datos que también necesitan protección. Las líneas del bus SPI son líneas de

alta velocidad de transmisión del orden de los MHz y son puntos críticos que deben de ser también protegidos. En concreto la transmisión SD pero esta se encuentra integrada en el módulo de pantalla ILI9341 como se puede ver en la figura 4.15.

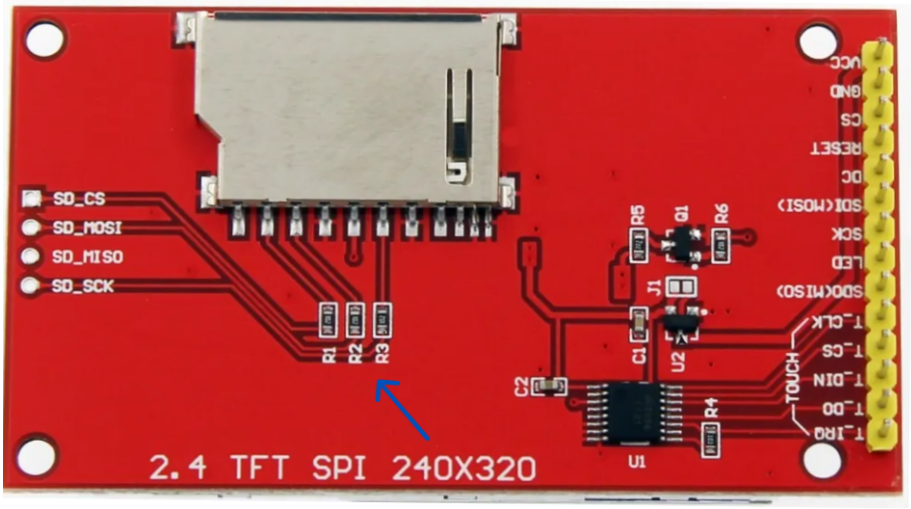


Figura 4.15 – *Parte trasera del módulo ILI9341*

La protección ESD debe de ser ubicada lo más cerca posible al punto donde se puede producir la descarga. En el caso de que el puerto de entrada SD se ubicase independientemente al módulo si podríamos colocar la protección pero en este caso no y en TIK Modelo 2 no se usará un módulo distinto. Una opción de protección, es la recomendada por Texas Instruments utiliza el TPD1E6B06 dispositivo de protección de un solo canal que utiliza un diodo de ESD en configuración bidireccional para ofrecer protección contra sobrevoltajes y se usa en líneas independientes para proteger en mayor medida líneas más críticas. Para proteger varias líneas a la vez, se usa el TPD4E101 (ver la imagen 4.16).

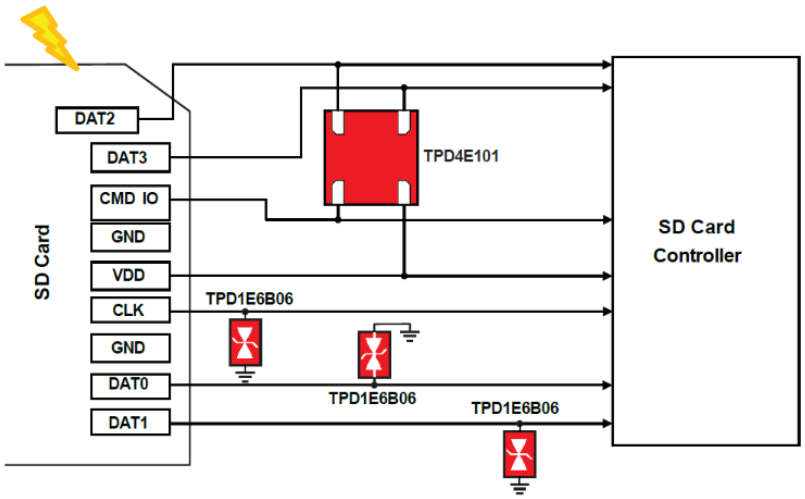


Figura 4.16 – *Protección SD frente a ESD [10].*

4.1.6. Buzzer

Para añadirle más personalidad al producto, se ha integrado un zumbador que haga de altavoz. Este zumbador nos permite añadir sonidos cada vez que interactuamos con el equipo, por ejemplo cuando tocamos el thumb button o cuando navegamos a través de la interfaz gráfica. También puede ser usado para indicar el inicio de una medición y

el final de la misma o avisar mediante una serie de pitidos cuando el usuario no interactúe con el equipo después de cierto tiempo, como lo hacen los multímetros digitales, para recordarnos que nos hemos dejado el equipo encendido y no lo estamos usando. Además, con la programación de alarma que incluye el RTC podemos programar avisos y recordatorios que tendrán un efecto sonoro gracias al buzzer.

El mercado nos ofrece poca gama de zumbadores, tan solo he encontrado dos tipos que nos sirvan para nuestro sistema. El primero es el típico buzzer pasivo THT muy usado en proyectos de arduino 4.17a y el segundo es un buzzer piezoeléctrico SMD de mayor tamaño con mayor potencia acústica y por tanto mayor consumo 4.17b.

(a) *Buzzer pasivo.*

(b) *Zumbador piezoelectrico.*

Figura 4.17 – *USB usados en TIK modelo 1 y 2.*

Como podemos observar en la tabla 4.4, el zumbador piezoeléctrico consume más y el propósito no es reproducir canciones o por el estilo, simplemente buscamos pitidos y con el buzzer pasivo nos basta.

Característica	Buzzer Pasivo	Buzzer Piezoeléctrico
Voltaje	3VDC	3 a 24VDC
Rango de Voltaje	2.7-3.5VDC	3 a 24VDC
Corriente (mA)	<=32mA	20mA
Nivel de Sonido	>=85dB	95dB
Frecuencia de Resonancia	2300Hz $\pm$ 300	3900Hz $\pm$ 500
Rango de Temperatura	-20°C a 45°C	-20°C a 60°C
Diámetro	30mm	22.5mm
Altura	10mm	11mm
Distancia entre Agujeros de Montaje	15mm	30mm

Tabla 4.4 – *Comparación de características entre un buzzer pasivo y un buzzer piezoeléctrico.*

4.1.7. Bombillas tubo de LED

Para conocer el estado de carga del dispositivo debemos de ver las luces LEDs que controla el cargador LT4056. Tenemos que ser conscientes que estas luces tienen cierta distancia desde la PCB hasta la carcasa superior donde se ubica la pantalla. La idea más simple es dejar dos orificios desde los que se verían estos LEDs, pero no es la mejor opción ya que no se verían nítidimante y uno reflejaría la luz sobre el otro haciendo que se difuminen los colores. Para ello en el mercado existen unos tubos tanto para LEDS SMD como para THT los cuales redireccionan esta luz y la conducen hasta un punto exacto.

Como se ve en la imagen 4.18b esta opción es más útil ya que el tronco negro que aparece se puede cortar y ajustar a la altura necesaria para después poder ubicar la bombilla del extremo final, mientras que la segunda opción no integra esta opción y no pueden ser ajustados. Por lo que, nos decantamos por esta segunda opción. En nuestro caso, se han hecho dos orificios para introducirlo fijarlo (ver imagen 5.34).

4.2. Arquitectura de software

Para desarrollar el Firmware necesitamos de un editor de código que nos permita agrupar los distintos archivos. El escogido ha sido Visual Studio Code, desarrollado por Microsoft además de gratuito, altamente conocido por los desarrolladores gracias a su rendimiento y características avanzadas de personalización. Su extensibilidad ha permitido instalar extensiones como la de Plataform IO que permite programar sistemas emebebidos actuando de entorno de desarrollo integrado IDE.

En cuanto al lenguaje de programación usado, ha sido C++ que es ampliamente usado en este tipo de proyectos ya que permite un control preciso del hardware acerca de la memoria, CPU y bajo nivel a la hora de tener mayor capacidad de manipular registros. Como primogénito de C, este permite utilizar las librerías y código escrito en C pudiendo aprovechar proyectos existentes. Ademas cuenta con un gran ecosistema en internet y una gran comunidad que nos ayuda a introducirnos en la programación de sistemas embebidos.

Puesto que nuestro dispositivo necesita responder a tiempo real y contamos con un microcontrador de más de un núcleo, freeRTOS como sistema operativo es ideal al ser de código abierto y ser amplimiante utilizado. Con el se busca la eficiencia haciendo que pequeños microcontroladores trabajen con tareas simultáneamente y consigan una gestión precisa del tiempo. Para enteneder bien como funciona, he aprendido con los vídeos de Digikey [41] y he ralizado un notebook sobre su funcionmiento además de una presentación para los compañeros del laboratorio de GranaSAT(consultar apéndice F para conocer más).

(a) *Tubo de LED transparente y no modificable.*

(b) *Tubo de LED con conductor ajustable en altura. Opción en SMD y THT.*

Figura 4.18 – *Tipos de tubos LEDs.*

Figura 4.19 – *Visual Studio Code.*

Figura 4.20 – *Lenguaje de programación C++.*

Figura 4.21 – *Extensión en Visual Studio Code para programar microcontroladores.*

Figura 4.22 – *Sistema operativo en tiempo real freeRTOS.*

4.2.1. Sistema operativo

Para poder manejar en nuestro dispositivo varias tareas al mismo tiempo y en tiempo real necesitamos utilizar un Firmware que se adecúe a nuestro hardware. El microocontrolador ESP32-WROOM-32D que tiene integrado el TIK modelo 1 maneja dos núcleos que nos permiten por ejemplo, manejar la pantalla a la misma vez que el dispositivo adquiere y lee señales por los BNCs. Este sistema operativo da prioridad a como se organizan las tareas, pudiendo controlar su estado gracias a los llamados mutex integrados en este equipo para coordinar los estados de las tareas. Un mutex usa una variable global que cambia su valor mientras que una tarea se está ejecutando en el núcleo, para el resto de tareas, al no encontrar esta variable en su valor de disponibilidad hace que las tareas se mantengan en espera a ser ejecutadas cuando la ultima tarea haya cabado de ejecutarse.

Observando la figura 4.23:

- Running: La tarea está usando uno de los núcleos. Solo una tarea puede estar en este estado por cada núcleo de CPU.
- Ready: La tarea está lista para ejecutarse pero no está ejecutándose en ese momento porque otra tarea de igual o mayor prioridad está en ejecución. En cuanto el núcleo deje de ejecutar una tarea entra ella.
- Blocked: La tarea está esperando una indicación para continuar, como por ejemplo la finalización de un temporizador. Mientras la tarea está en este estado, no consume recursos de la CPU.
- Suspended: La tarea está explicitamente suspendida. No está lista para ejecutarse y no pasará al estado listo.

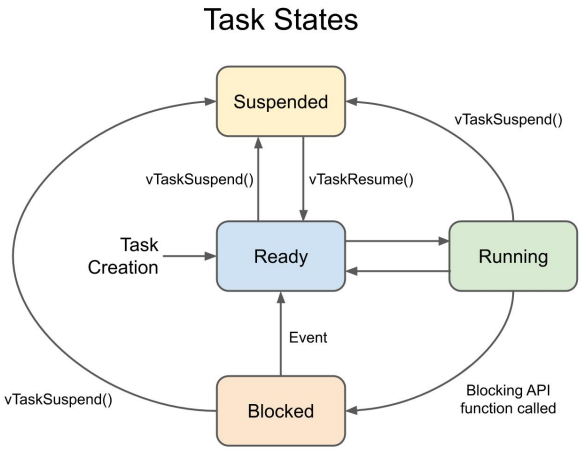


Figura 4.23 – *Estados y como pasar entre unos y otros estados de tareas.*??

Los clásicos Arduinos UNO, integran solo un núcleo ATmega328P de la marca Atmel y son con los que cualquiera se inicia en el mundo de la programación de microcontroladores. Estos tan solo permiten concatenar tareas una tras otra y no permiten realizar dos de ellas a la vez, además, cuando el núcleo recibe una ISR, puesto que todas las tareas tienen la misma prioridad, se ejecutará cualquier tarea. Esto con RTOS no pasará ya que el sistema operativo reconoce cuales son más urgentes y las ejecutará gestionando de mejor forma las tareas. Podemos afirmar por lo tanto, que un sistema operativo super loop no nos permite control sobre las tareas y para nuestro equipo será determinante hacerlo, por ejemplo, si recibimos una orden de apagado del equipo las tareas de guardado de datos en la SD o de terminar con la recogido de señales END 4.24. Para conocer más,

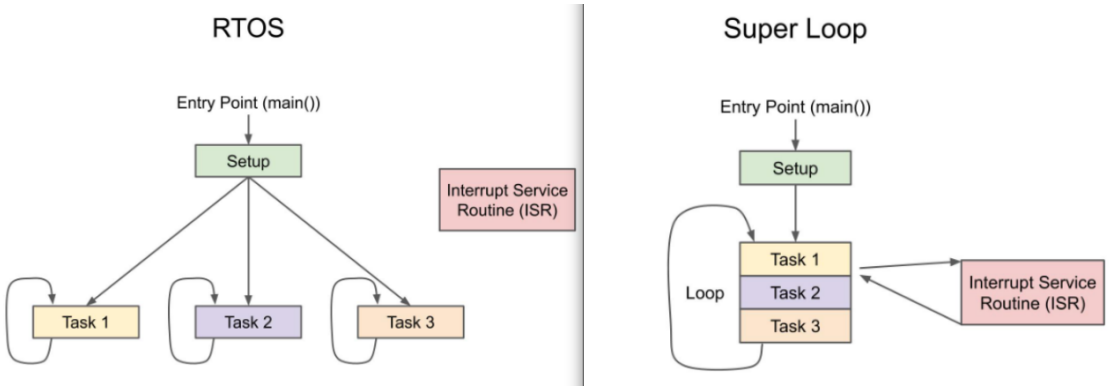


Figura 4.24 – *Diferencias entre utilizar tres núcleos o un núcleo con varias tareas.*??

4.3. Arquitectura mecánica

El equipo se diseñó en un primer instante teniendo en cuenta que es un producto portable y manejable por lo tanto el tamaño y el peso son cruciales. El equipo ha de sentirse robusto (pero no pesado) porque puede recibir caidas además de encontrarse en ambientes de sol, polvo y/o humedad. Para más restricciones impuestas revisar sección 4.4.1 y 5.3 [6].

Figura 4.25 – *Renderizados del equipo TIK Modelo 1. Figura 5.52 [6].*

Este dispositivo está pensado para sujetarse durante un largo tiempo. Para ello se ha realizado un estudio con material plástico con las dimensiones y formas esperadas en el equipo, después se ha sujetado firmemente con ambas manos (diestra y zurda) para dejar marcadas las posiciones naturales de los dedos al coger un bloque similar. Puesto que los dedos de mi mano no son universales, se han hecho más grandes los huecos dando así cierta tolerancia.

En un principio se ideó hacer un dispositivo para diestros ya que el 89,4 % de la población lo es [42]. Sin embargo, los dispositivos móviles intentan ser ambidiestros y por ello, se ha buscado incluir las hendiduras de ambas manos. En la figura 4.26a podemos ver en la versión 1 los dos modelos tanto para mano derecha como para izquierda, en la tercera imagen vemos el modelo B definitivo para ambidiestros.

4.4. Softwares implicados en el dise

Para hacer el diseño, he usado el programa SOLIDWORKS 2021 4.26a con el cual me he instruido bastante durante el curso realizado este año por la UGR: CURSO DE FORMACIÓN PERMANENTE EN DISEÑO MECÁNICO Y

SIMULACIÓN TÉRMICA Y ESTRUCTURAL CON SOLIDWORKS (9ª Edición) [43]. Este software de diseño asistido es muy usado en el ámbito de la ingeniería industrial permitiendo crear modelos en 2D y 3D muy precisos. La interfaz es muy intuitiva y facilita un gran número de herramientas para el modelado paramétrico y así poder crear piezas, ensamblajes y planos. Otra parte muy interesante del software es la funcionalidad de simulación de estrés mecánico, análisis de movimiento y cálculo de mecánica de fluidos pudiendo simular las zonas de mayor probabilidad de rotura así como las zonas donde el equipoa alcanza mayor temperatura.

Una vez que hemos realizado el diseño usamos el software Ultimaker Cura 4.26b que convierte los modelos 3D exportados desde Solidworks en formato .stl, en datos G-code que consiste en modelos digitales en instrucciones donde se indica en los tres ejes los puntos exactos donde la impresora debe de imprimir. El programa nos permite elegir nuestro modelo de impresora 3D con sus limitaciones dimensionales en cuanto a capacidad de tamaño en la impresión. El programa nos permite ajustar la velocidad de impresión, temperatura, tipo de soportes disponibles .. etc.

(a) *Versión 1 para zurdos y diestros distintivamente, junto al modelo final.*

(b) *Proceso de modelación del prototipo final.*

(c) *Modelo final acabado.*

(a) *Ubicacion del lapiz para pantalla táctil.*

(b) *Orificio para la ubicación de tornillo de sujección.*

Figura 4.26 – *Bobina de filamento PLA del color usado.*

Capítulo 5

Diseño de sistema

5.1. Diseño de Hardware

Para realizar el diseño de Hardware se ha utilizado el programa Altium Designer. Este software de diseño de PCB es referencia en la industria de la electrónica. Ofrece herramientas para el diseño de esquemáticos, la creación de layouts y nos permite generar la documentación necesaria para la fabricación. Altium es la plataforma de referencia usada entre los compañeros de GranaSAT por lo que iniciarme en ella es un acierto ya que si tengo cualquier duda algún compañero del laboratorio me puede ayudar.

Hay alternativas de Open Source como KiCad u Eagle (esta última usada en la asignatura de Prototipado y test electrónicos del grado de IEI [44]).

Figura 5.1 – *Altium designer.*

Figura 5.2 – *KiCad EDA.*

Figura 5.3 – *Eagle designer.*

5.1.1. Esquemáticos añadidos y/o modificados

A continuación se detallarán los esquemáticos añadidos y la selección de componentes necesaria para el correcto funcionamiento, así como la justificación de los valores de los componentes pasivos.

5.1.1.1. Microprocesador

En la figura 5.6 podemos ver el diseño de pines. Aunque el nombre inferior del componente indique la letra D, el montado en TIK Modelo 2 es el E, y se ha mantenido el diseño del componente ya que el footprint y el pinout es el

mismo para uno u otro modelo de microprocesador de Espressif.

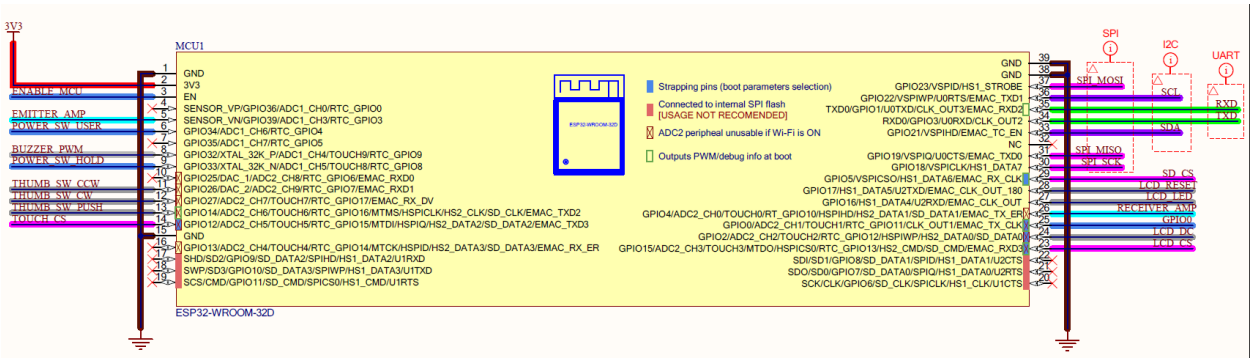


Figura 5.4 – *ESP32-WROOM-32E.*

En el apartado de esquemáticos (A) podemos ver los condensadores de desacoplo recomendados por el fabricante para asegurar una alimentación estable.

- Pines internos para SPI: En la versión anterior se usaron los pines 20 y 22 para los puertos Jacks con el objetivo de conectar el equipo TIK a otros equipos, esto ya no es necesario y estos pines dejan de ser usados ya que el protocolo SPI los necesita y tuvieron que ser cortadas las pistas en la anterior versión.
- Emisor y receptor de señal de END: El compañero de equipo Antonio Morales me indicó que en la anterior versión se usaba el pin 4 como emisora de señal y el 5 como receptora pero que no era la mejor opción porque el pin 4 (canal 0 del adc1) siempre manda paquetes de 12 bits sin información del i2s, mientras el que el pin 5 (canal 3 del adc 1) envía paquetes de datos de salida del ADC con 14 bits, siendo los dos bits más significativos información de la configuración i2s, por lo que el cambio ejecutado es el 5 como emisora y el 26 como receptora.
- Cambio pin problemático para el circuito de encendido: El pin que debe mantener en alto la señal del DC/DC ha sido cambiado del 11 al 9 y asegurar que eliminamos el comportamiento inestable del anterior pin.

El resto de pines han sido configurados como indica la datasheet en cuanto a pines de entrada y/o salida. Para más información consultar la tabla 3 y 4 de la datasheet [45] donde se describen los pines de strapping y todos los demás.

En el rediseño, se ha reubicado este integrado para contar con el puerto USB hembra lo más al centro posible. Para ello la reubicación del microprocesador ha sido necesaria manteniendo las limitaciones de diseño para la antena impuestas por el fabricante(figura 5.5). Estas indican que no puede haber plano de masa ni pista alguna a 15 mm de la antena para asegurar que no hay problemas con la señal de comunicación de radiofrecuencia.

5.1.1.2. Tensión de polarización para los conectores hembra de los BNCs

Para polarizar los conectores y que estos tengan una tensión de referencia estable se necesita un valor distinto de los 3.3V de alimentación que nos aporta el DC/DC por lo que en la anterior versión se usó un LDO NCP562SQ18T1G con 1.8V a la salida. En esta actualización se ha considerado poner un divisor de tensión que nos permita ajustar esa tensión de referencia. Además si es necesario, podemos cambiar la tensión de referencia de uno u otro conector. Para aportar estabilidad de tensión se han usado grandes capacitores electrolíticos de aluminio que actúan como condensadores de desacoplo.

Los condensadores usados son los RVT1000UF10V34RV0081 5.7. del fabricante KNSCHA. Busqué condensadores que no sean demasiados altos para que no sean más altos que el módulo de pantalla y así asegurar que la carcasa contiene sin problemas el condensador. La altura del condensador es de 10mm mientras que la pantalla alcanza los 13.5 mm.

Para analizar el comportamiento de la tensión de salida del resistor he simulado en LTSpice un entorno de ruido

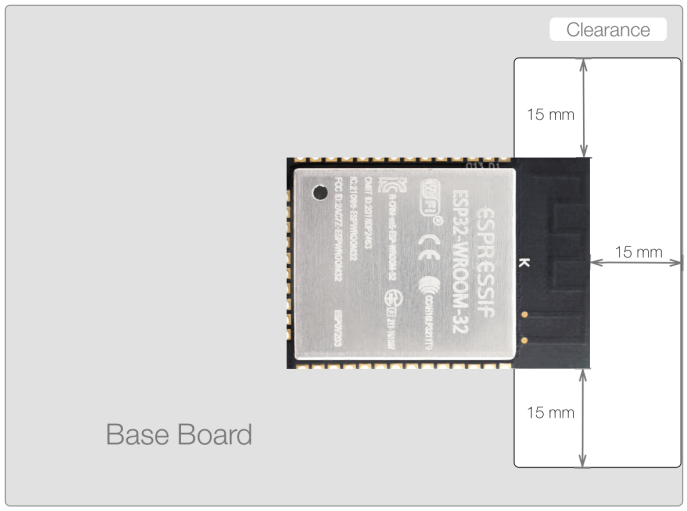


Figura 5.5 – *ESP32-WROOM-32E limitaciones para la antena.*

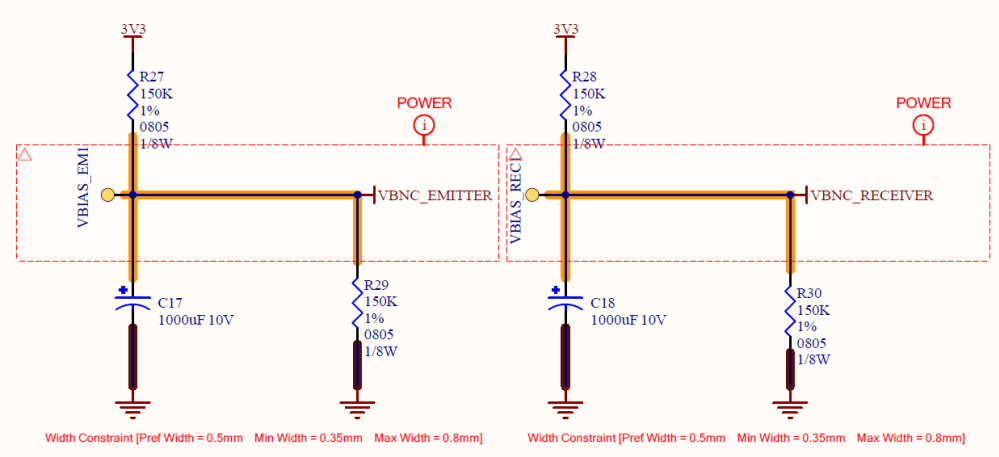


Figura 5.6 – *Tensión de polarización para los conectores BNCs.*

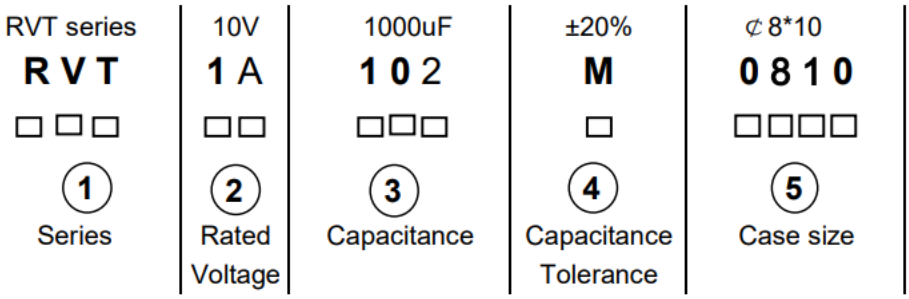


Figura 5.7 – *Código de referencia de los condensadores electrlíticos. [11]*

en la alimentación de 3.3 V, así podremos ver la respuesta en frecuencia de esta tensión crucial que polariza los conectores.

Como podemos ver en el circuito de polarización 5.8 el análisis en frecuencia, cuanto mayor es la frecuencia, mayor desacoplo se produce con el valor de polarización de los conectores y menos ruido alcanza, por ello está justificado incluir un condensador de gran capacidad y asegurar que la tensión tiene el menor suido posible 5.9.

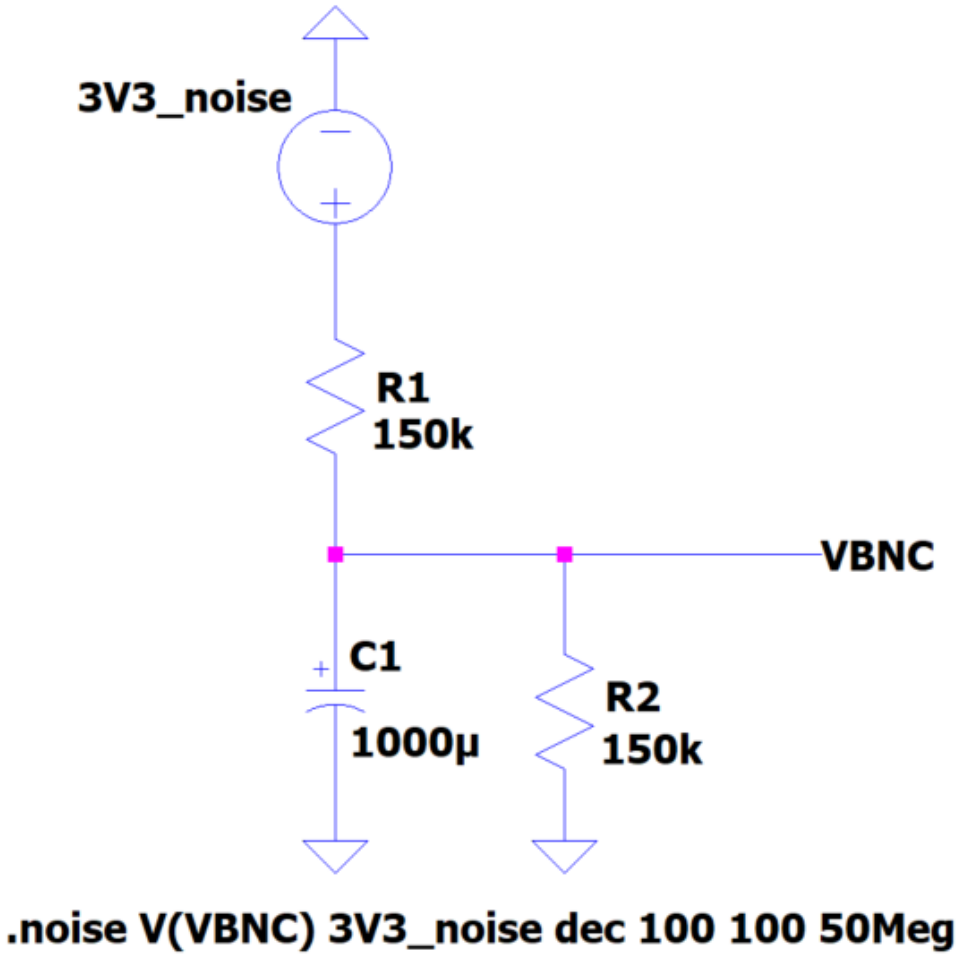


Figura 5.8 – *Divisor de tensión para la polarización de los conectores BNCs en LTSpice.*

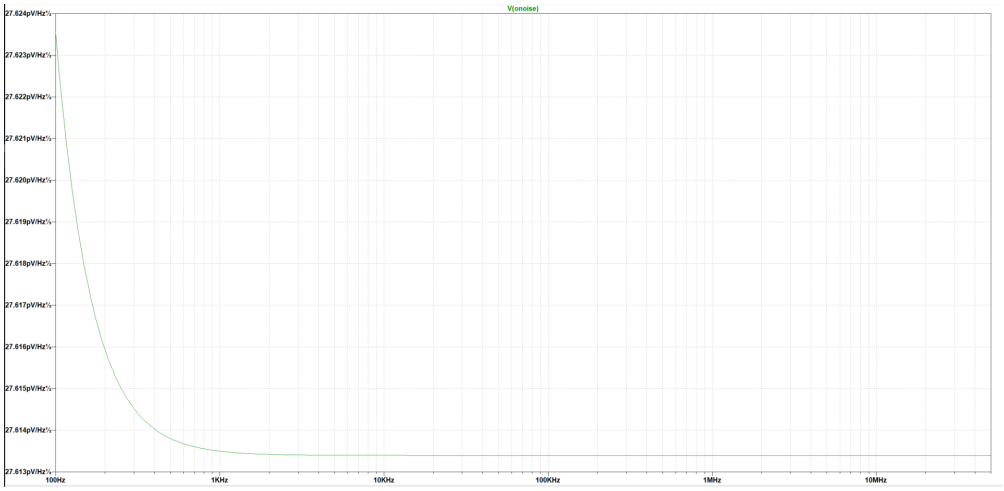


Figura 5.9 – *Respuesta en frecuencia con condensador de 1000uF.*

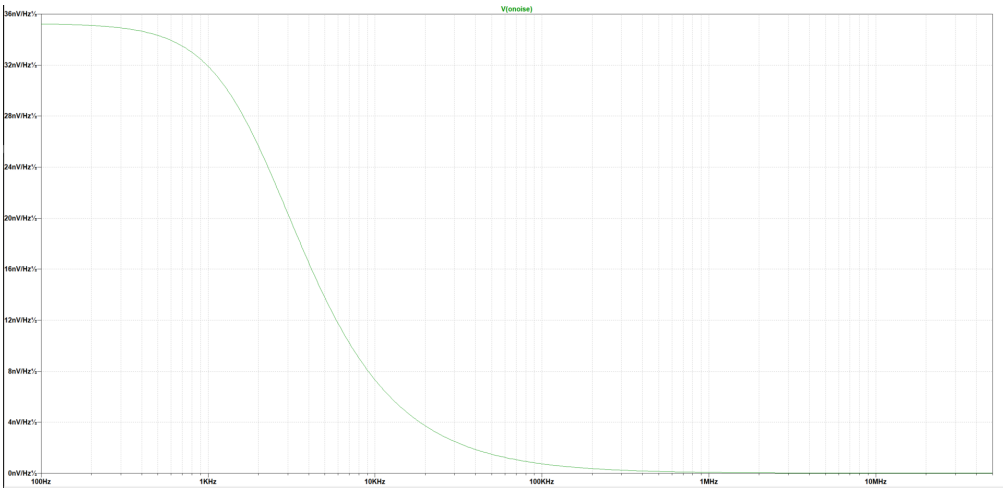


Figura 5.10 – *Respuesta en frecuencia con condensador de 1nF.*

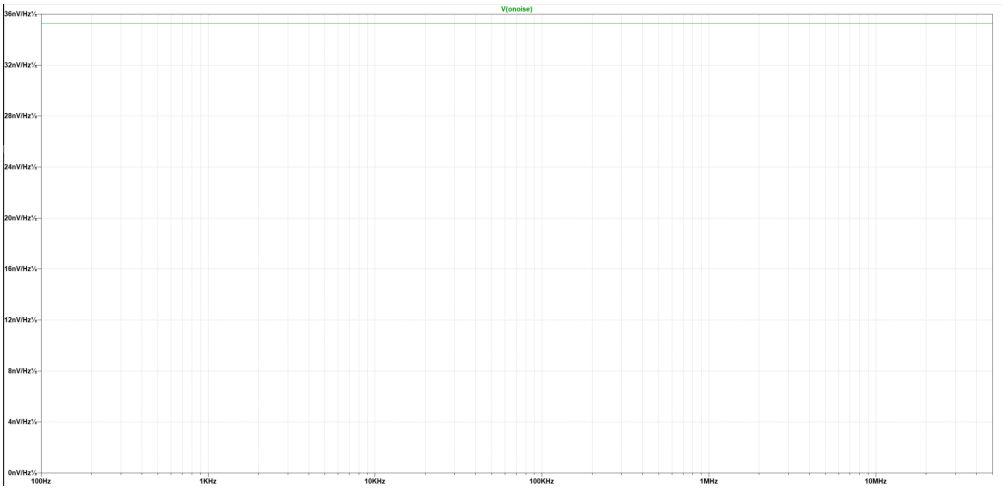


Figura 5.11 – *Respuesta en frecuencia sin condensador.*

5.1.1.3. INA219

El integrado medidor de voltaje, intensidad y por tanto consumo es otra vez escogido por su disponibilidad en el laboratorio de GranaSAT, además hay mucha documentación y es ampliamente usado. Además, su ajuste interno es fácil de realizar y una vez hecho esto he comprobado que es bastante preciso. Otras alternativas como el INA3221 tienen más canales para un mismo ADC y es más complejo e innecesario para nuestro caso.

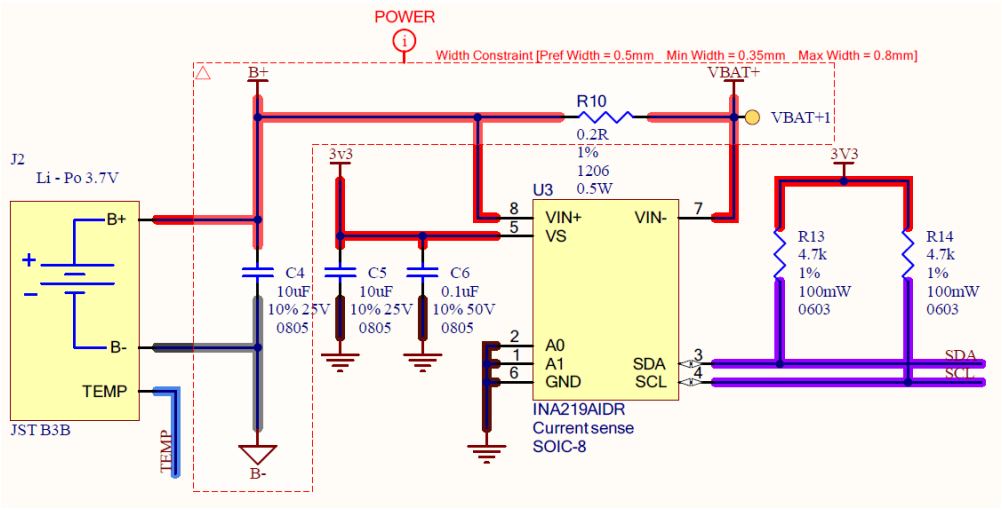


Figura 5.12 – *Esquemático del INA219.*

El pin de alimentación VCC ha sido modificado y conectado a los 3,3V. Anteriormente, estaba conectado al polo positivo de la batería. Esta no es una buena práctica ya que como indica la figura 5.13 el integrado es alimentado con un rango de tensión de 3V a 5.5 V, si la tensión de la batería disminuye por debajo de los 3V (que puede ser el caso ya que las baterías de 3.7V de LiPo pueden bajar de los 3V aunque puede ser crítico y dañarse) el integrado dejará de estar alimentado.

Entre los pines de VBAT+ y VBAT- hay un condensadore bypass de 10 uF que asegura alta impedancia entre los polos de la batería.

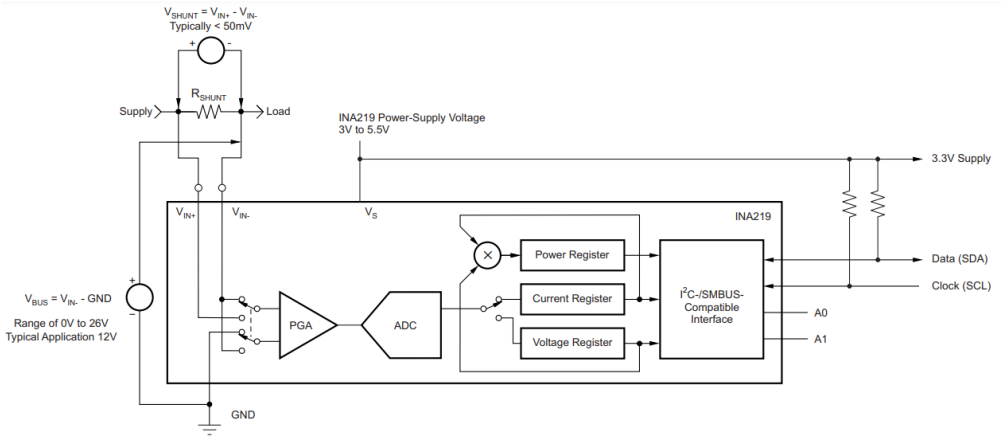


Figura 5.13 – *Esquemático del INA219 con filtro de entrada. Figura 13 [12]*

5.1.1.4. LT4056

Para la protección de carga volvemos a contar con el integrado LT4056 5.15 pero en este caso usamos el pin habilitado para la monitorización de la batería que integra. Esta línea denominada TEMP va desde ese tercer hilo de la batería del conector JST B3B integrado para conectar la batería como se ve en la figura 5.12.

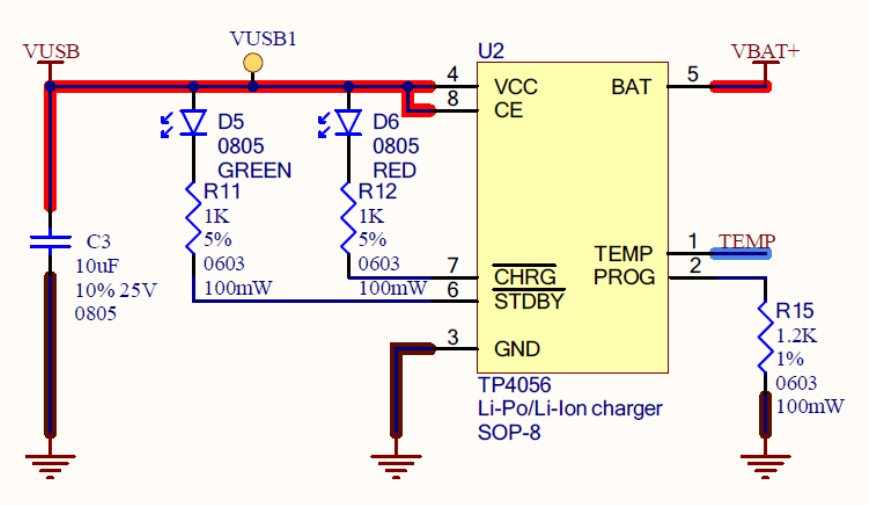


Figura 5.14 – *Integrado LT4056 encargado de la protección de la batería.*

Los LEDs que indican la carga ahora son controlados íntegramente por el LT4056 iluminando el LED STBY cuando la batería ya está cargada y CHRG mientras que la batería se está cargando.

5.1.1.5. TFT LCD ILI9341

El único cambio en este esquemático es el pin LED que ahora con la conexión al pin 27, GPIO16 podemos controlar gracias al transistor de la parte trasera figura 3.24b, una señal PWM que mediante la modulación del ancho indique un valor mayor de tensión y viceversa.

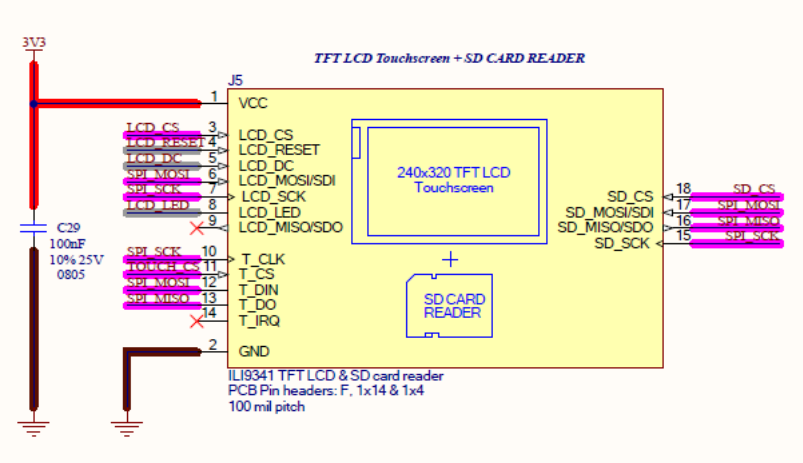


Figura 5.15 – *Esquemático módulo ILI9341.*

5.1.1.6. Buzzer 3VDC

Para integrar un buzzer necesitamos tan solo un transistor MOSFET canal N y una resistencia pull down (ver figura 5.16). La puerta queda conectada al GPIO32 de entrada y salida y con la resistencia Pull down aseguramos que el buzzer no pita ni consume intensidad hasta que la puerta recibe un valor alto correspondiente a 1 (3.3V del microprocesador). Cuando recibe el valor alto el buzzer se ve alimentado por su polo positivo a 3.3V y el negativo a masa (GND).

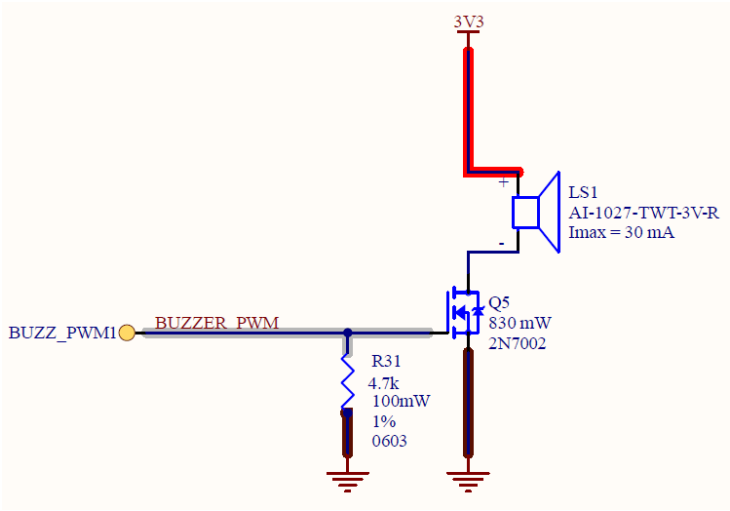


Figura 5.16 – *Buzzer pasivo de 3V.*

5.1.1.7. RTC

EL integrado RTC necesita de una batería externa al aprincipal y un oscilador externo conocido también como cristal de cuarzo. Este oscilador conectado entre los pines OSCI e OSCO se encarga de mantener una referencia temporal exacta. Con la frecuencia de oscilación escogida de 32.768 kHz el integrado es capaz de dividir internamente para generar señales de tiempo necesarias en segundos, minutos y horas. El oscilador genera una señal periódica sin recibir señal de entrada, tan solo con la alimentación se empieza a generar ruido térmico y fluctuaciones de voltaje que permiten al cristal de cuarzo comenzar su oscilación y mantenerse estable por sus propiedades físicas.

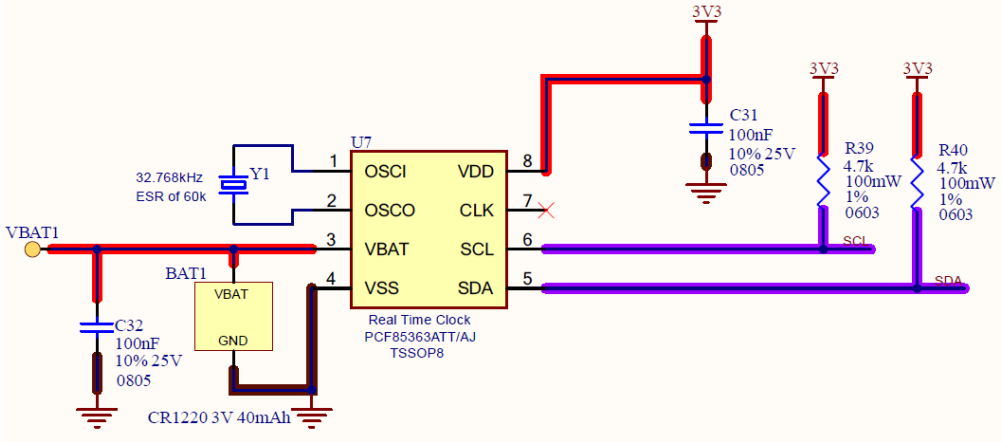


Figura 5.17 – *Esquemático del RTC.*

Como podemos ver en la figura 5.17 tenemos dos pines de alimentación: VDD y VBAT. La alimentación principal viene de la batería LiPo conectada a nuestro equipo TIK, mientras que el equipo esté encendido esta batería alimentará al RTC. Cuando se apague el equipo, el integrado cambiará automáticamente a la alimentación de la batería VBAT , batería de respaldo de 3V y 40mAh ubicada en el portapilas 4.12.

He incluido también dos condensadores de 100 nF en la alimentación del integrado del pin VDD y el pin VBAT. Esto asegura gran impedancia entre 3.3V y VBAT con GND mostrando niveles de tensión de alimentación DC estables.

5.1.2. Enrutados y renderizados finales de la PCB

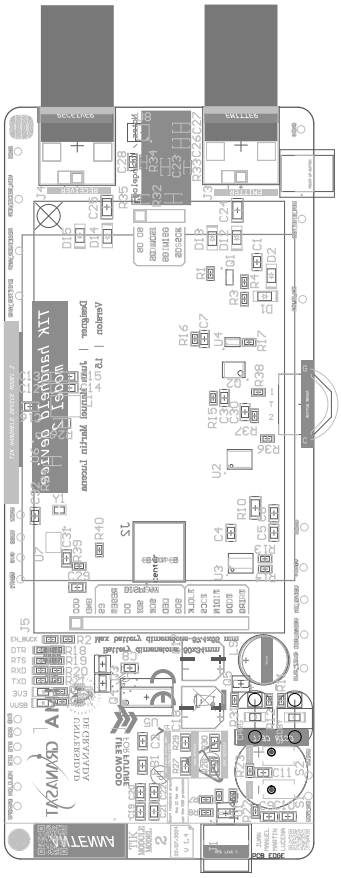
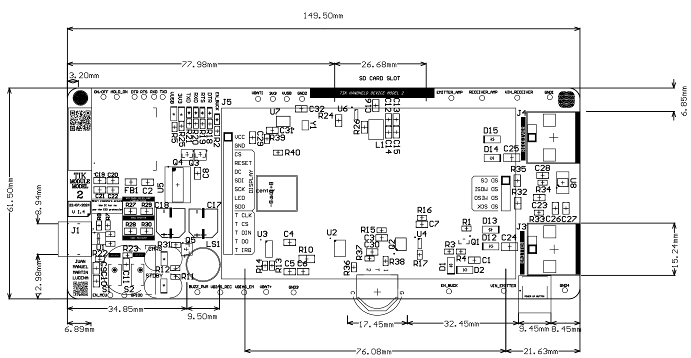


Figura 5.18 – *Escala de grises PCB TIK Modelo 2.*



Fig[redacted]edidas PCB TIK *Modelo 2.*

5

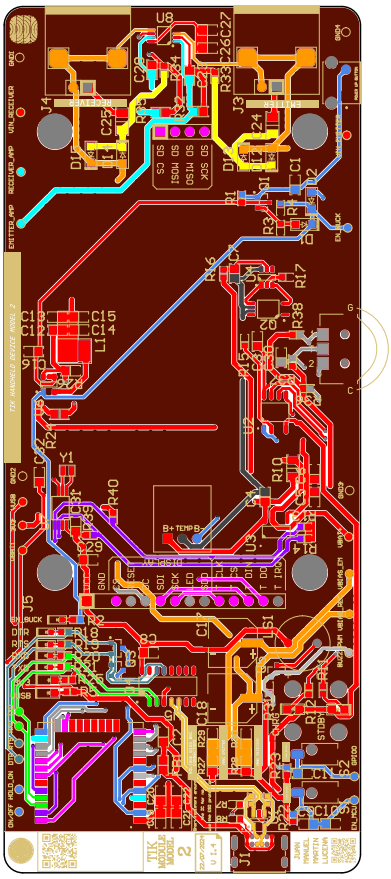


Figura 5.20 – *Pistas superiores de PCB TIK Modelo 2.*

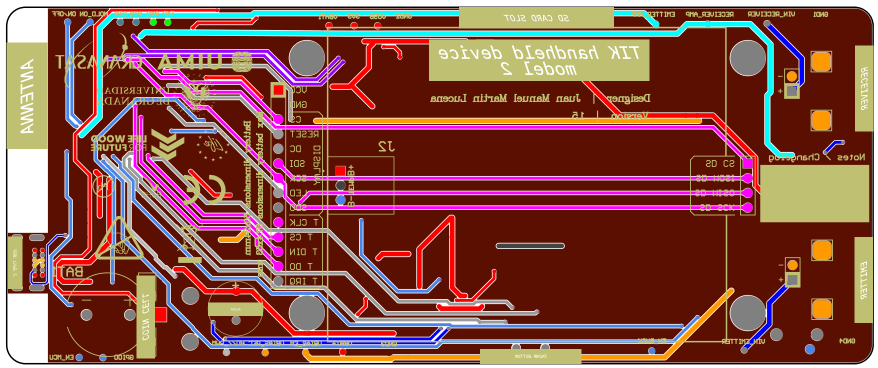


Figura 5[redacted]feriores de PCB TIK *Modelo 2.*

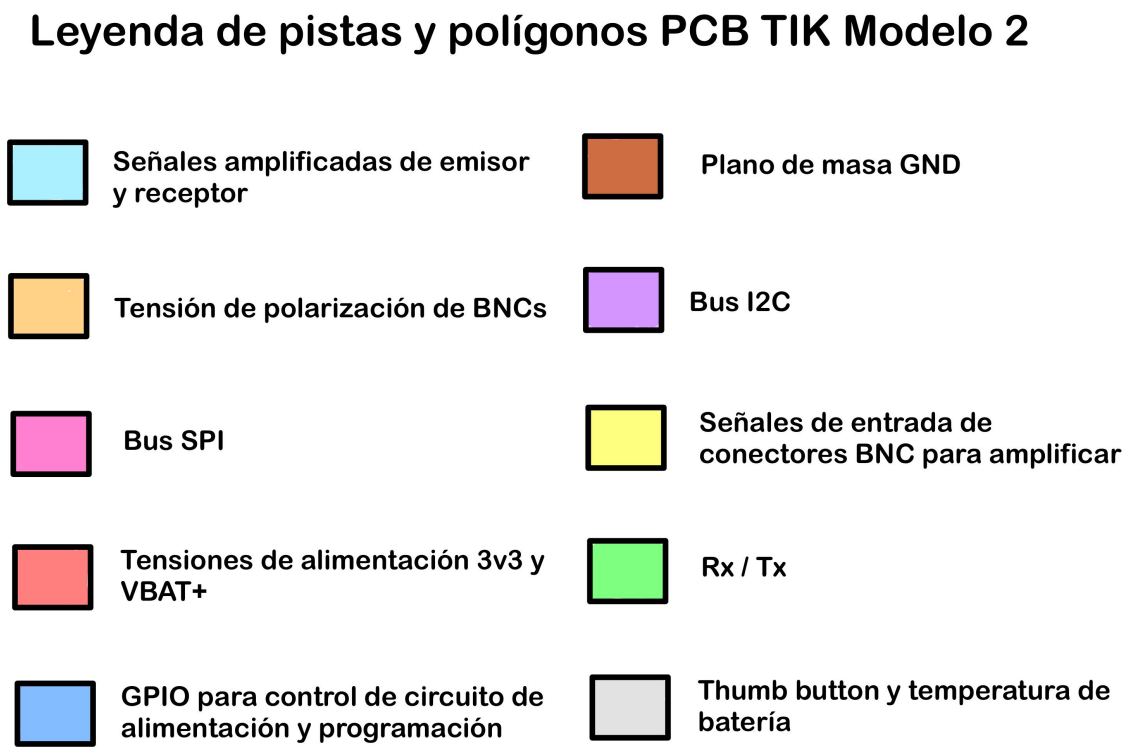


Figura 5.22 – *Leyenda de pistas de PCB TIK Modelo 2.*

5

5.1.3. Manejo de capas

Las PCBs pueden tener distinas capas las cuales con una fina capa de cobre y marcado de pistas producen la conducción eléctrIca. Las más simples como es nuestro caso solo tienen una capa superior y otra inferior, más conocidad como top y bottom layer. Como aislante entre ellas se ubica un material de base que generalmente está hecho de FR-4 (fibra de vídeo epoxy) que además de aiSlar eléctricmente propoRciona rigidez. La capa de soldadura o solder mask es la capa que cubre a la de cobre para protegerlas de la oxidación y evitar que se produzcan soldaduras donde no se debe, esta generalmente es verde como se ve en las imágenes 5.28. Finalmente tenemos la capa de serigrafía donde se imprimen los textos y simbolos de referencia que ayuda en el proceso de enamblaje a ubicar los componentes y en mi caso es de color blanco. En la imagen 5.35 podemos ver la organización de capas usada y en concreto observar que, nuestra placa llevará 1 oz de cobre en la top y otra en la bottom layer. Este ajuste de peso se indicará después para importar las reglas de Altium al fabricante a encargar hacer la PCB.

Las PCBs multicapa son usadas en dispositivos complejos de alta densidad de componentes. Además de las capas top y bottom las cuales llevan soldados los componentes, hay capas internas dedicadas enrutar señales y/o otras capas exclusivas a manejar las tensiones de alimentación y masa. Estas PCBs multicapa permiten un diseño más compacto al permitir más espacio para enrutar psitas y reducen el ruido al tener capas dedicadas a tierra y potencia mejorando así la integridad de señales de alta velocidad. Sin embargo, son PCBs más caras de fabricar y más difíciles de reparar. Por esto último, se ha desestimado hacer una PCB de estas características.

No debemos de olvidar que este trabajo se enmarca aún en la etapa educativa y los alumnos se inician en el diseño de placas por lo que hacer una de estas características no permitiría manipular las pistas cortándolas y soldando hilos de un lugar a otro (como se ha hecho en el presente trabajo) si la pista a modificar se encontrase en una capa interna.

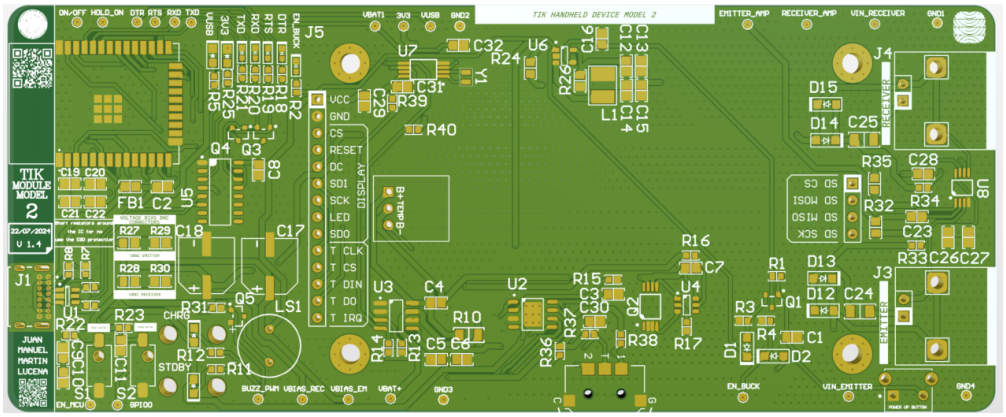


Figura 5.23 – *Visión 3D superior sin componentes de PCB TIK Modelo 2.*

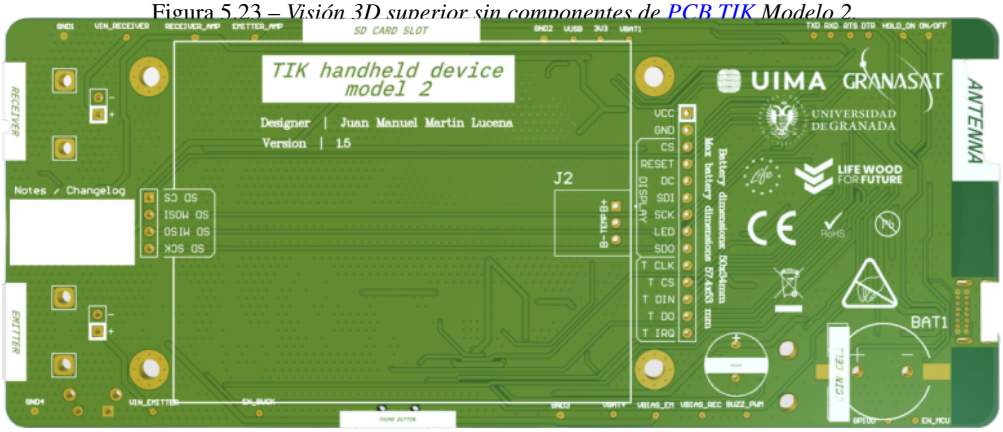


Figura 5.24 – *Visión 3D inferior sin componentes de PCB TIK Modelo 2.*

Por ello, ha sido preferible usar solo dos capas de conducción, superior e inferior.

5.2. Diseño de Firmware

Una prioridad del proyecto tomado de [6] ha sido organizar los directorios así como archivos que finalmente no han sido usados y estabas presentes. Además, una organización limpia de código es quella que desde el main.cpp tenemos la inicializan de los periféricos necesarios y la declaración de las tareas de freeRTOS asignándole las cualidades que deseemos a la tarea como podemos ver en la imagen 5.36.

Como podemos ver en la imagen 5.49 el fichero RTOS_tasks.cpp integra programadas las tareas que posteriormente en el fichero main.cpp son declaradas y se les da las instrucciones de ejecución según el formato de RTOS. Estas tareas son (ver figura 5.38):

- RefreshTouch: Esta trea se encarga de refrescar la pantalla si cada 5 ms (indicado en la variable LOOP_TICKS) y además, indica un punto negro en cada parte de la pantalla que no haya botones o algún tipo de interacción. Finalmente devuelve el mutex para que el resto de tareas puedan realizarse. Esta tarea está asignada al núcleo 0 y con una memoria de 16384 bytes. Esta memoria de pila asignada, como las del resto de tareas deben de optimizarse en un futuro si la complejidad del Firmware va en aumento.
- StatusBarRefresher: Actualiza la barra de pantalla destinada a dar información sobre el estado del WiFi, Bluetooth, batería y pantalla en la que se encuentra el usuario. Tarea destinada también al nucleo 0 y con 16384 bytes.

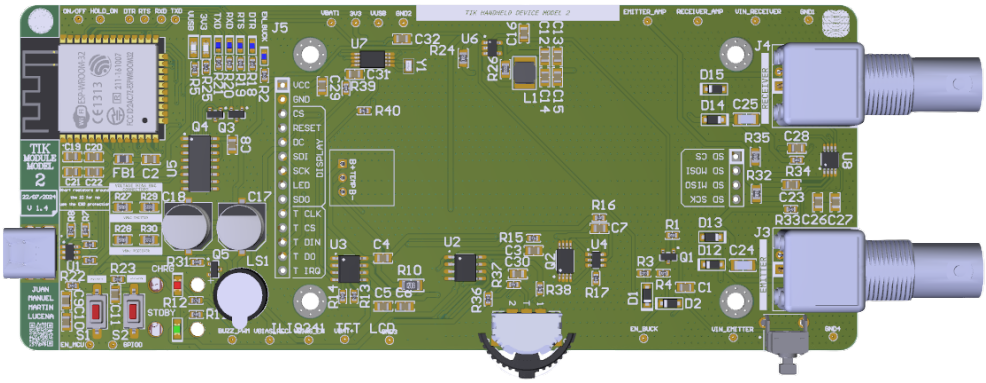


Figura 5.25 – *Visión 3D superior sin módulo ILI9341 de PCB TIK Modelo 2.*

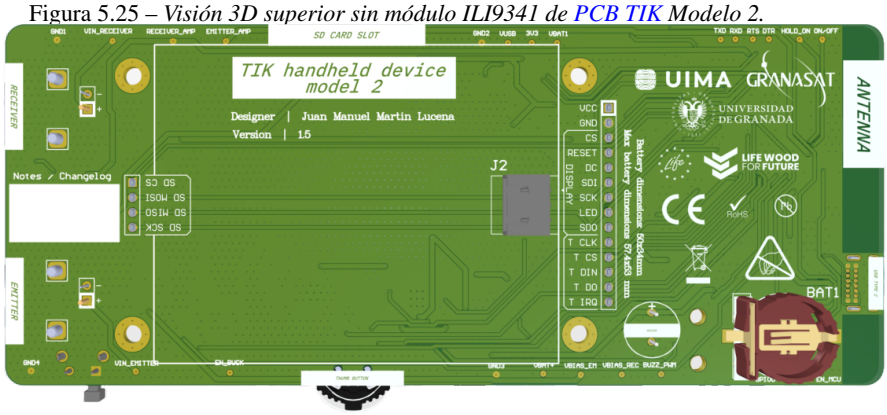


Figura 5.26 – *Visión 3D inferior sin módulo ILI9341 de PCB TIK Modelo 2.*

- ScreenManager: Tarea que se encarga de dibujar en pantalla en cada momento las funciones definidas en la clase gui a petición del usuario gracias a la navegación por el dispositivo. Es actualizada con la misma frecuencia que la tarea refreshtour y tiene las mismas cualidades de memoria y núcleo asignado que las anteriores.
- OnOffManager: Tarea que se encarga de leer la actividad del botón del circuito de encendido y mantener en alto con otro pin de salida la señal de enable del DC/DC que alimenta a 3.3V el equipo.
- MonitorBattery: Tarea que se encarga de monitorizar el estado de la batería por tensión y modificar el icono de la batería. Esta tarea y la anterior, se encuentran en el núcleo 1 con la misma memoria de pila que el resto de tareas.

Figura 5.27 – *Visión 3D superior con módulo ILI9341 PCB TIK Modelo 2.*

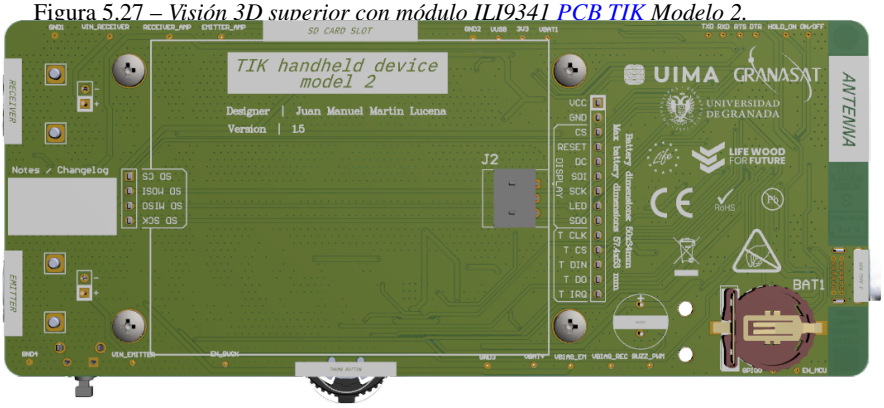


Figura 5.28 – *Visión 3D inferior con módulo ILI9341 de PCB TIK Modelo 2.*

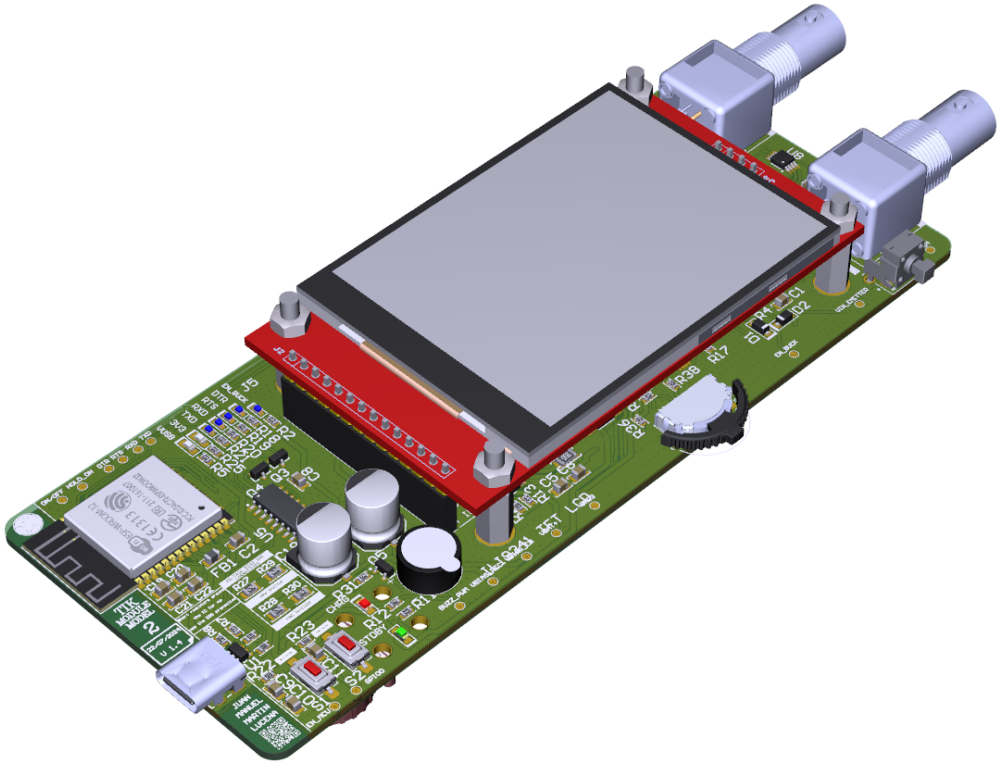


Figura 5.29 – *Visión 3D superior de PCB TIK Modelo 2.*

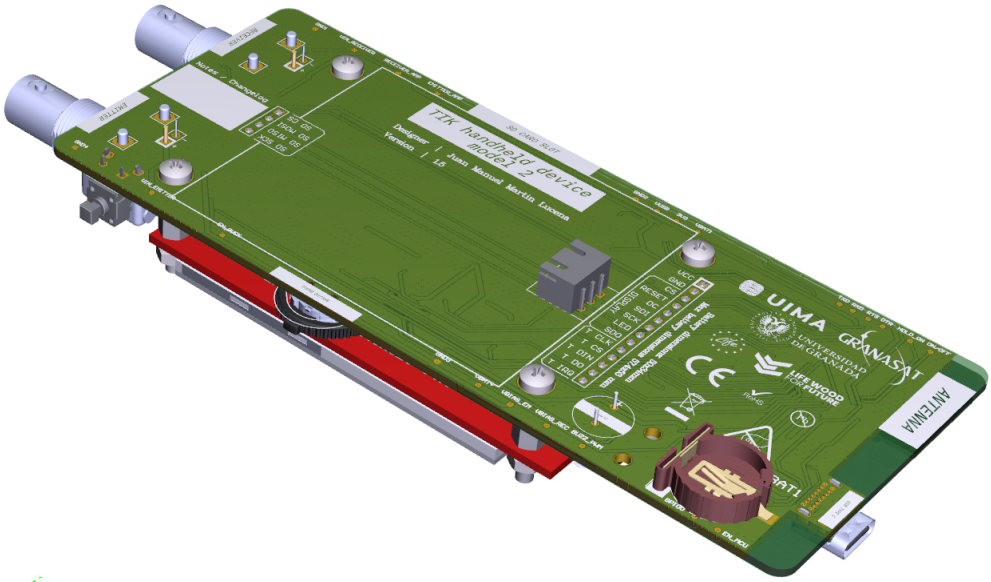


Figura 5.30 – *Visión 3D superior de PCB TIK Modelo 2.*

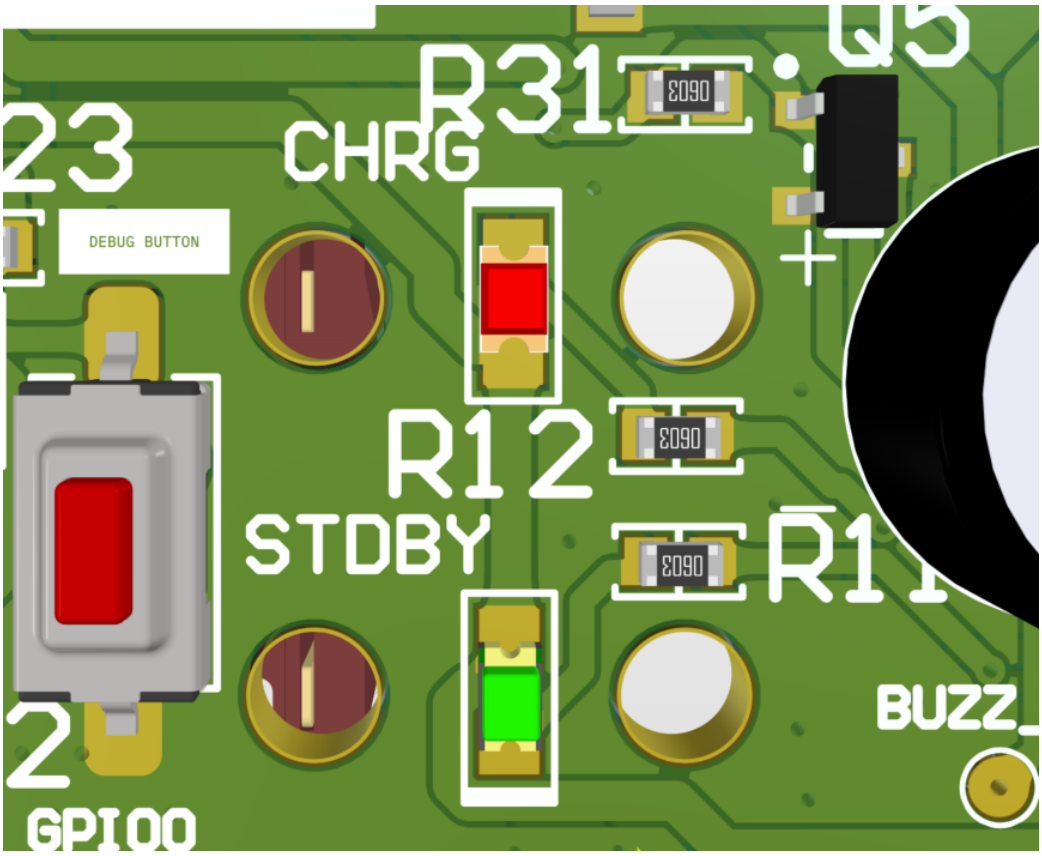


Figura 5.31 – *Detalle de orificios para tubo LED SMD.*

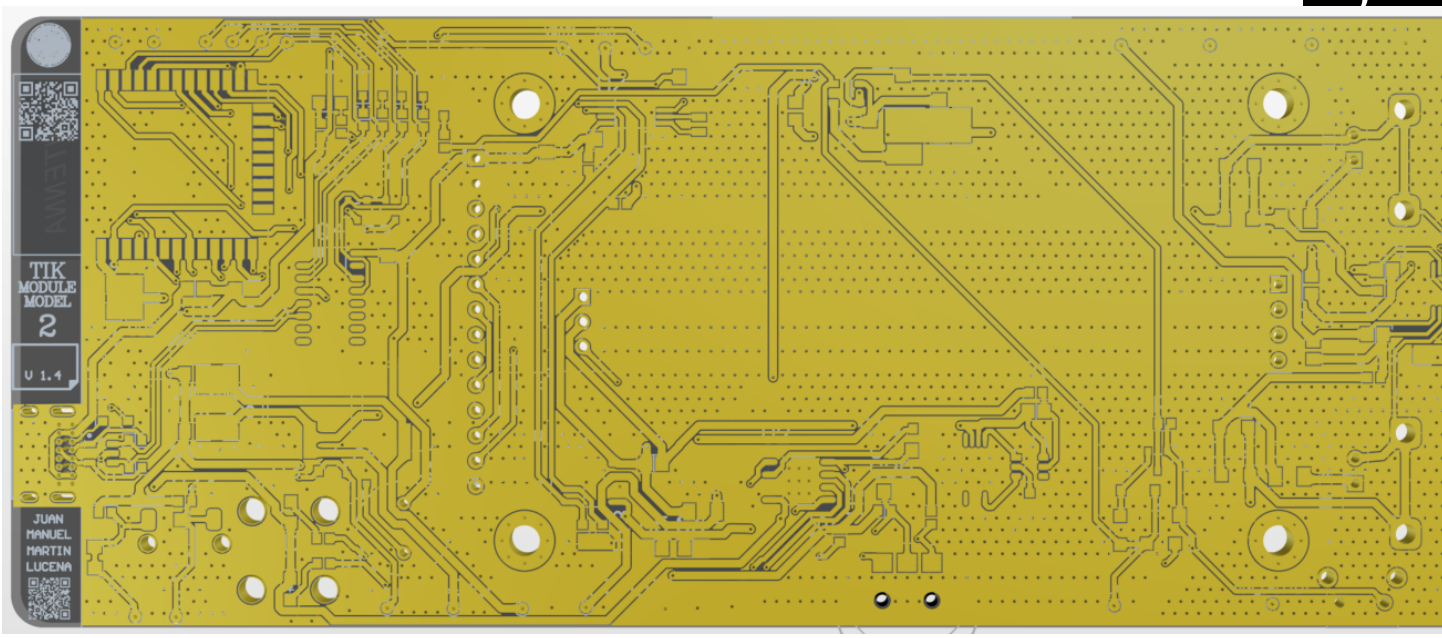


Figura 5.32 – *Detalle de vía stitching para protección electromagnética de la PCB.*

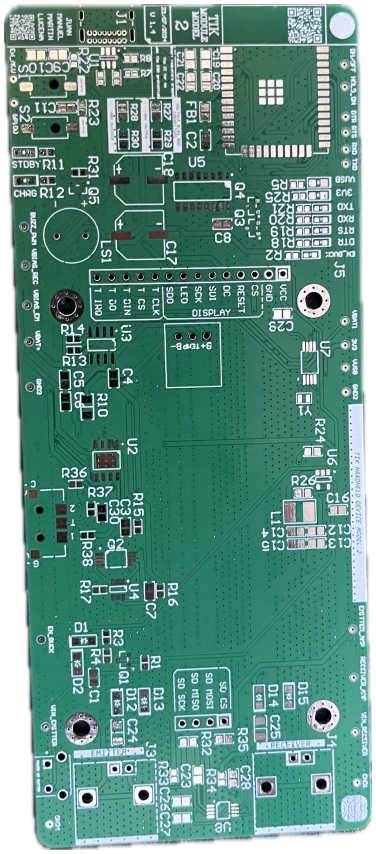


Figura 5.33 – *Modelo PCB real TIK Modelo 2.*

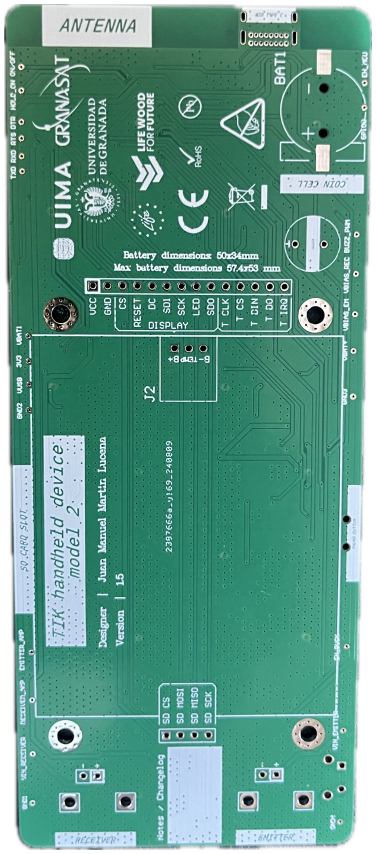


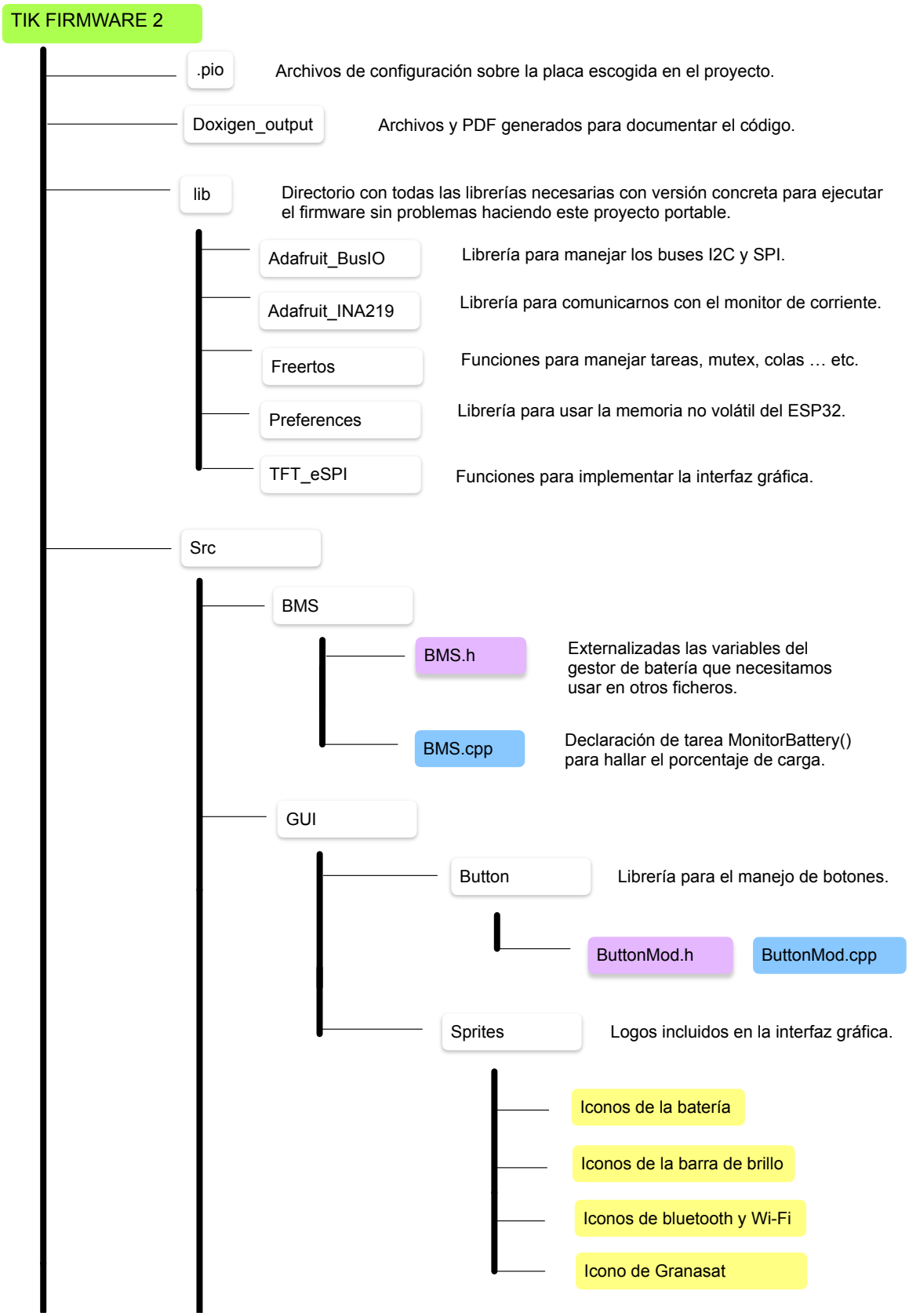
Figura 5.34 – *Modelo PCB real TIK Modelo 2.*

#	Name	Material	Type	Weight	Thickness	Dk	Df
	Top Overlay		Overlay				
	Top Solder	Solder Resist	Solder Mask		0.01mm	3.5	
1	Top Layer		Signal	1oz	0.04mm		
	Dielectric1	FR-4	Dielectric		1.5mm	4.5	
2	Bottom Layer		Signal	1oz	0.04mm		
	Bottom Solder	Solder Resist	Solder Mask		0.01mm	3.5	
	Bottom Overlay		Overlay				

Figura 5.35 – *Capas de PCB TIK Modelo 2.*

```
xTaskCreatePinnedToCore(
                    Task1code,   /* Task function. */
                    "Task1",     /* name of task. */
                    10000,       /* Stack size of task */
                    NULL,        /* parameter of the task */
                    1,           /* priority of the task */
                    &Task1,      /* Task handle to keep track of created task */
                    0);          /* pin task to core 0 */
```

Figura 5.36 – *Función para crear y asignar las cualidades a una tarea ligada a un núcleo freeRTOS*


TIK FIRMWARE 2
.pio
Archivos de configuración sobre la placa escogida en el proyecto.
Doxigen_output
Archivos y PDF generados para documentar el código.
lib
Directorio con todas las librerías necesarias con versión concreta para ejecutar el firmware sin problemas haciendo este proyecto portable.
Adafruit_BusIO
Librería para manejar los buses I2C y SPI.
Adafruit_INA219
Librería para comunicarnos con el monitor de corriente.
Freertos
Funciones para manejar tareas, mutex, colas … etc.
Preferences
Librería para usar la memoria no volátil del ESP32.
TFT_eSPI
Funciones para implementar la interfaz gráfica.
Src
BMS
BMS.h
Externalizadas las variables del gestor de batería que necesitamos usar en otros ficheros.
BMS.cpp
Declaración de tarea MonitorBattery() para hallar el porcentaje de carga.
GUI
Button
Librería para el manejo de botones.
ButtonMod.h
ButtonMod.cpp
Sprites
Logos incluidos en la interfaz gráfica.
Iconos de la batería
Iconos de la barra de brillo
Iconos de bluetooth y Wi-Fi
Icono de Granasat

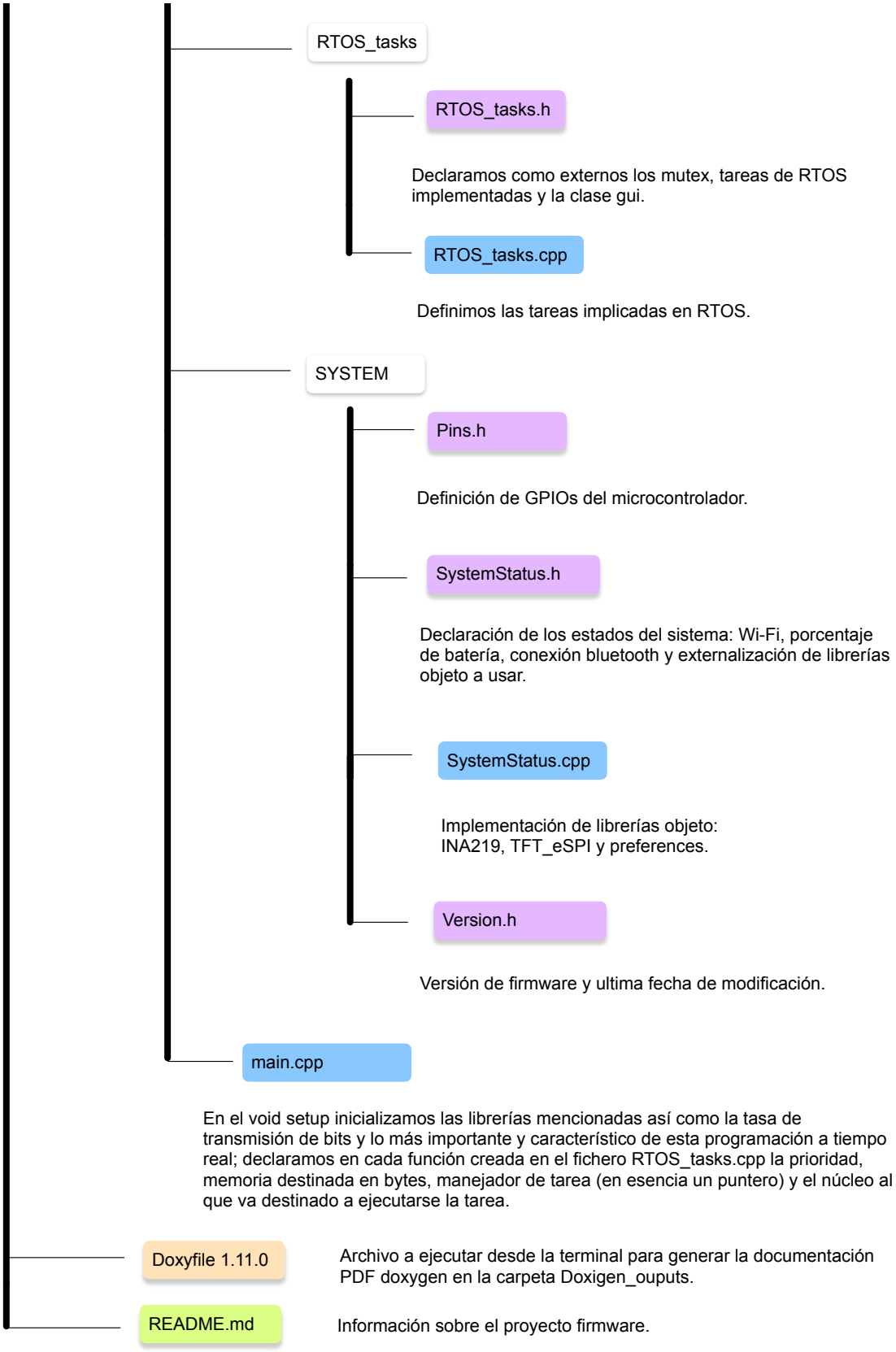


Figura 5.37 – *Directorios y archivos del proyecto firmware TIK Modelo 2.*

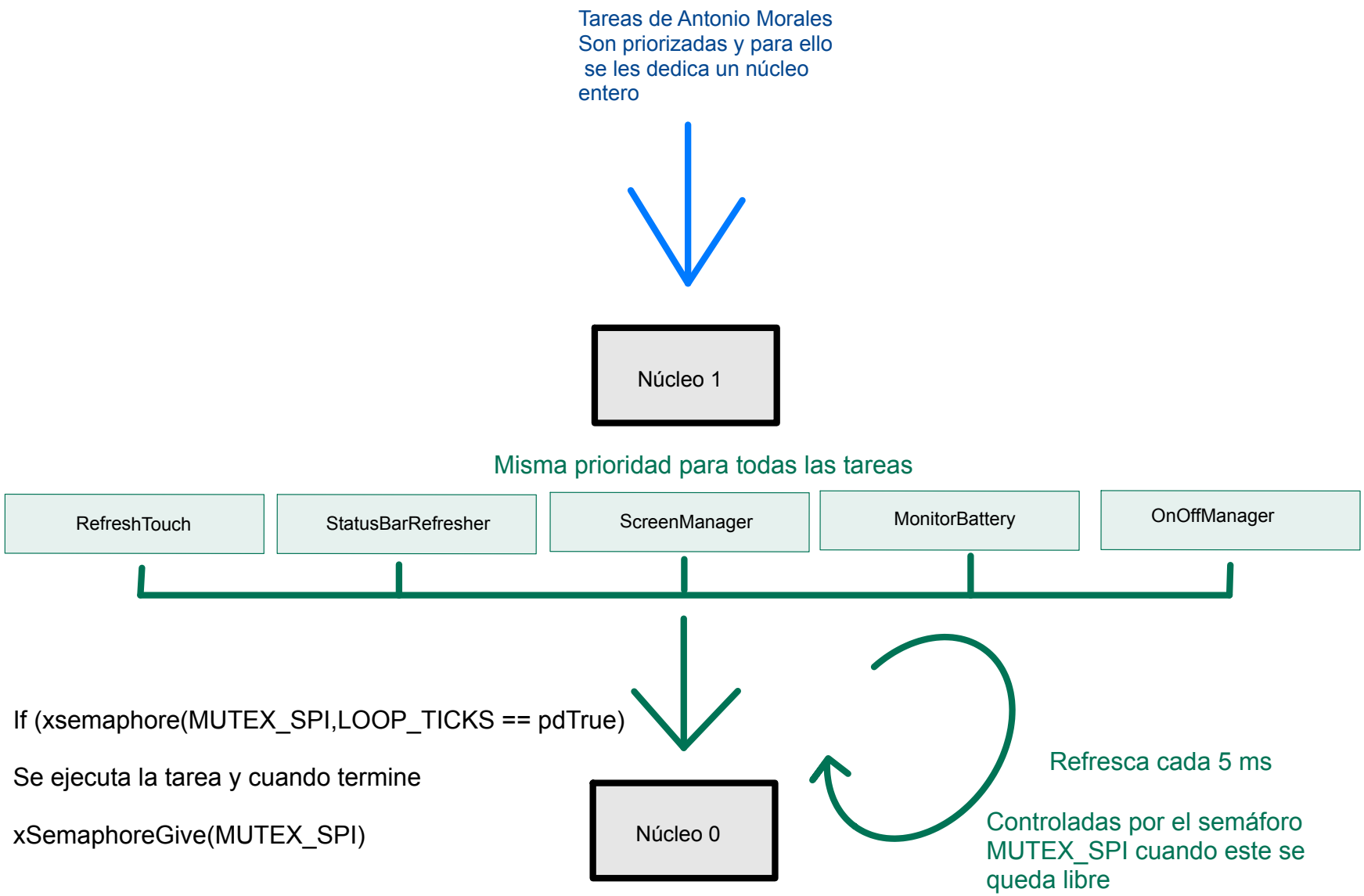


Figura 5.38 – *Manejo de tareas en el Firmware TIK Modelo 2.*

5.2.1. Gestión de pantallas TIK Modelo 2

A continuación, podemos observar las distintas pantallas que se encuentran en el TIK Modelo 1 que serán perfectamente trasladadas a la siguiente versión TIK Modelo 2 al estar el proyecto portable.

5.2.1.1. Pantalla principal

En la pantalla de inicio (ver figura 5.39), se puede ver lo que el usuario percibe cuando enciende el equipo por primera vez. De forma intuitiva se pueden acceder a distintas opciones como:

- Measure: En esta pantalla podemos visualizar las señales recogidas por las sondas, está al principio ya que el objetivo del dispositivo reside fundamentalmente en las señales END. Para más información sobre esta pantalla consultar la memoria de TFG de Antonio Morales.
- Settings: Desde esta pantalla podemos visualizar las funcionalidades del equipo.
- Help y shutdown: Estos apartados se encuentran actualmente vacíos a falta de continuar desarrollando el Firmware.
- Modo de ingeniería: A este modo podemos acceder si pulsamos a la vez el botón S4 de encendido y el botón desplizante central. A el accedemos a información más relevante como tensión de la batería, memoria RAM libre del equipo, versión del equipo así como testear los LEDs del equipo y calibrar la pantalla (imagen 5.40).

Figura 5.39 – *Pantalla de inicio en TIK Modelo 1.*

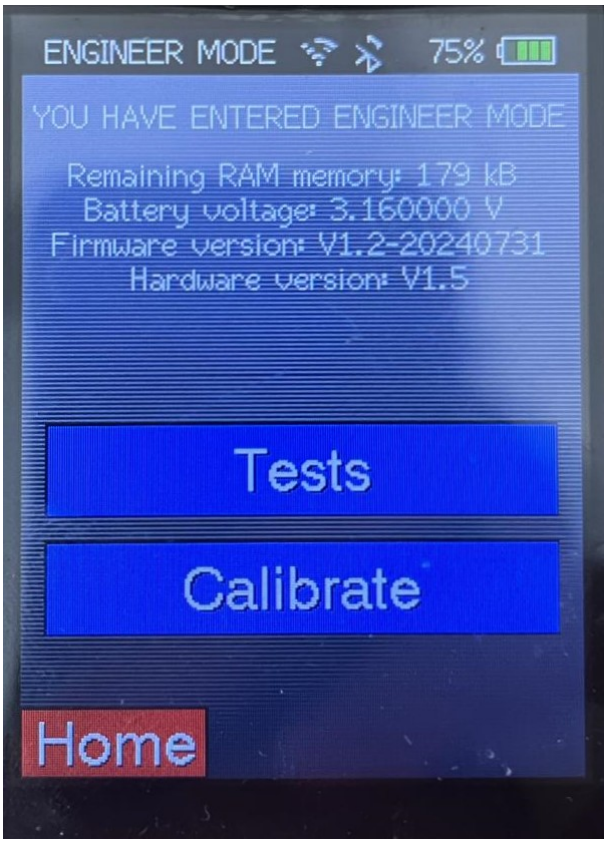


Figura 5.40 – *Pantalla del modo de ingeniería en TIK Modelo 1.*

5.2.1.2. Ajustes

- Connect: Desde aquí accedemos al apartado de conectividad del equipo TIK el cual el SoC integrado soporta Wi-Fi y Bluetooth. Cuando activemos o desactivemos una de ellas el equipo mostrará en pantalla el logo en cuestión (ver figura 5.41).
- Display: Aquí encontramos información sobre la pantalla que integra el módulo ILI9341 así como una barrita que nos permite controlar el brillo del dispositivo (5.42).
- Battery: Estos apartados se encuentran actualmente vacíos a falta de continuar desarrollando el Firmware. La idea es mostrar información más detallada de la batería una vez sea implementado un monitorizador de batería más sofisticado que el actual.
- Memory: En esta sección podemos guardar los datos recogidos de las mediciones para almacenarlos en la memoria introducida la cual indica los megabytes libres, y si no hay memoria conectada indicará 0 megabytes. Se detallan además las instrucciones de medida a seguir para realizar las mediciones.

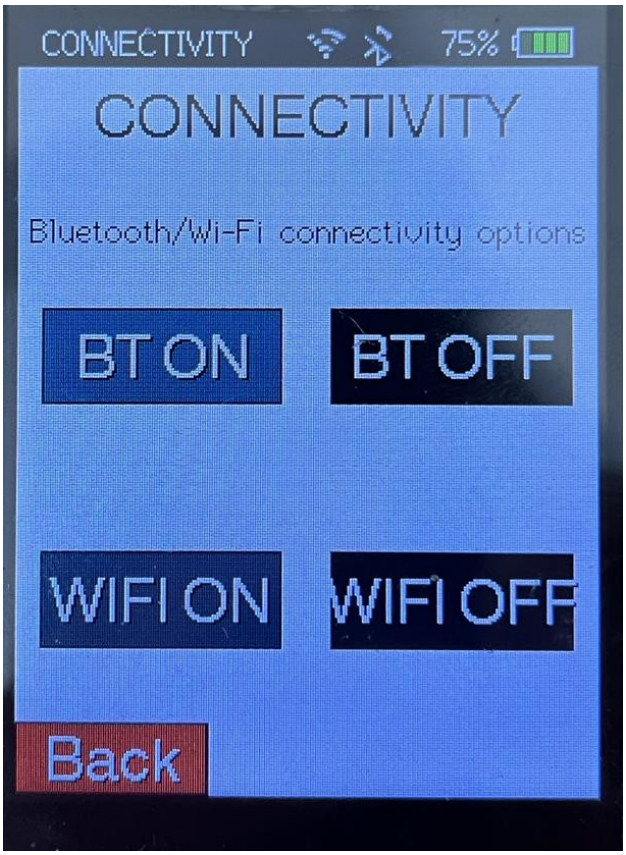


Figura 5.41 – *Pantalla de conectividad en TIK Modelo 1.*

5.2.1.3. Gestión y monitorización de la batería

La forma de mostrar al usuario el System on Chip (SoC) debe de de una forma sutil y no saturar con información que no necesita acerca de la batería de Lithium polymer (LiPo). La opción escogida, es la de mostrar en la barra de estado (como en los dispositivos móviles actuales) un icono de una pila con 4 franjas que indican el 25, 50, 75 y 100 % además de indicar cuando la batería está cargándose como se puede ver en la figura 5.48.

Se planteó dos formas de mostrar los iconos de la figura 5.48:

Figura 5.42 – *Pantalla de display con brillo bajo en TIK Modelo 1.*

Figura 5.43 – *Pantalla de display con brillo emdio en TIK Modelo 1.*

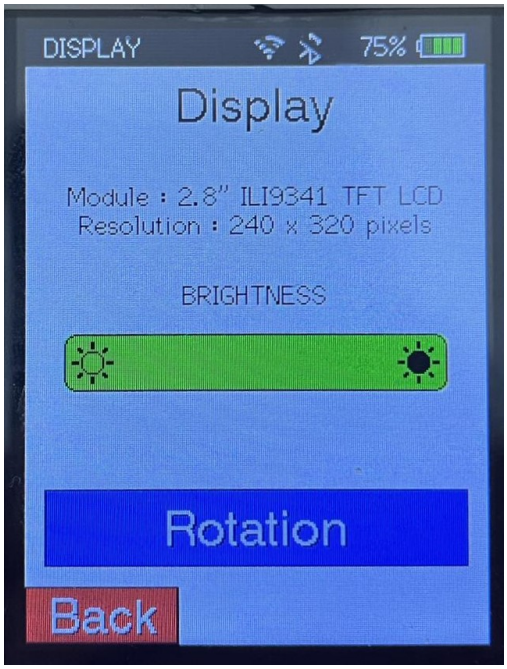


Figura 5.44 – *Pantalla de display con brillo alto en TIK Modelo 1.*

- Monitorización de la tensión: Con el IN219 podemos obtener constantemente la tensión en los polos de la batería y con las curvas obtenidas en la figura 3.29a podemos aproximar un porcentaje de batería. Aquí no tenemos un

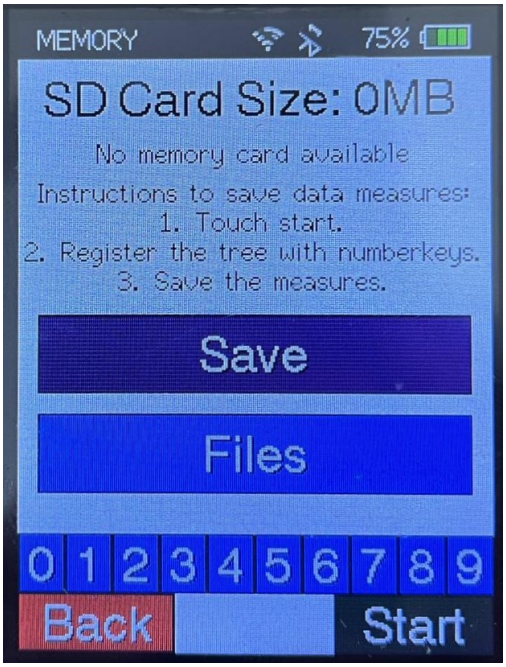


Figura 5.45 – *Pantalla de memoria sin tarjeta de memoria en TIK Modelo 1.*

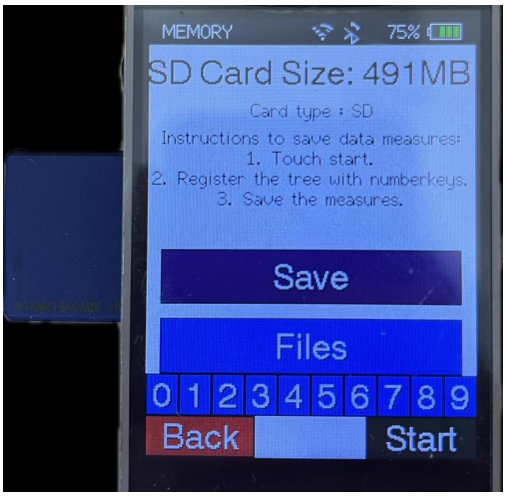


Figura 5.46 – *Pantalla de memoria con tarjeta de memoria en TIK Modelo 1.*

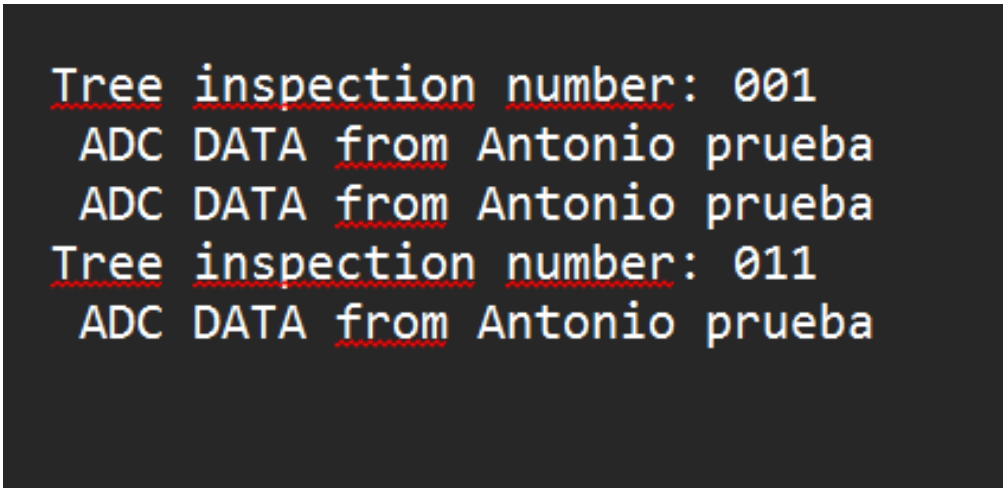


Figura 5.47 – *Guardado de texto desde la pantalla de memoria del TIK Modelo 1.*

(a) *Charging icon*

(b) *High battery level*

(c) *Okay battery level*

(d) *Medium battery level*

(e) *Low battery level*

Figura 5.48 – *Símbolos del estado de la batería.*

gran control del estado de la batería ya que como sabemos las curvas de la batería se degradan con el paso del tiempo, suciedad ... etc y las tensiones de referencia para el estado de la batería pueden variar. Por lo tanto, necesitaríamos cada cierto tiempo obtener las curvas de la batería montada ya ctualizar los valores de referencia.

- Monitorización de la corriente consumida: Aquí debemos de especificar en el Firmware la capacidad actual de nuestra batería, una vez que esta esté cargada al máximo, calculamos la intensidad que se va consumiendo para multiplicarla por el tiempo que el equipo está encendido y finalmente se resta a la capcidad total. Esto requiere de utilizar la memoria no volátil del SoC para almacenar el estado de la capacidad mientras que el equipo está apagado. Esta es la opción ideal.

Puede ocurrir con este método que, como se comentó en la sección de 3.5 en la figura 3.30, la degradación de una batería puede llevar a cambios abruptos en la carga o descarga, si esto sucede y se utiliza la monitorización de la batería por tensión, puede que este cambio esté en el límite superior o inferior de un valor referencia para cambiar el icono de porcentaje y haga que cambie rápidamente entre uno y otro icono.

Hasta la fecha presente, la monitorización de la batería por tensión ya que no ha sido posible por temas de tiempo implementar la opción ideal de monitorización por corriente consumida. Sí se ha añadido portable la librería Preferences que es la necesaria para realizar el almacenamiento en la memoria no volátil.

5.2.2. Modificación de la librería Adafruit INA219

La librería de Adafruit para el integrado INA219 se ajusta en función de tres modos: hasta 16V y 400 mA, 32V y 1A y 32V y 2A. La referencia que debemos de poner en el setup de nuestro main es de 32V y 2A ya que como máximo, por la resistencia de shunt pueden circular 1.6A (revisar sección 5.1.2.3.3 [6]). Esta biblioteca está pensada para usar una resistencia de shunt de 100 mΩ , y en nuestro caso tenemos una de 200 mΩ.

Usar una resistencia mayor nos ayuda a tener mayor caida de tensión (y mayor pérdida energética por disipación) entre los polos de la resistencia que van conectados a los pines Vin+ y Vin- del INA219. Tener mayor caida de tensión nos aporta más rango dinámico para el ADC del integrado haciendo la medición más precisa. Además, un bajo rango dinámico es más susceptible al ruido por lo que nuestro ADC puede tomar valores de tensión de la batería que en realidad es ruido.

Para integrar este nuevo valor en la librería debemos de añadir una variable a la función setCalibration_32V_2A que pasará a su argumento como ina219_calvalue. Esta variable que depende de nuestra resistencia de shunt, modifica un registro del integrado.

```
1     /*!
2     *  @brief  Configures to INA219 to be able to measure up to 32V and 2A
3     *          of current.  Each unit of current corresponds to 100uA, and
4     *          each unit of power corresponds to 2mW. Counter overflow
5     *          occurs at 3.2A.
6     *  @note   These calculations assume a 0.1 ohm resistor is present
7     */
8    void Adafruit_INA219::setCalibration_32V_2A(uint32_t ina219_calValue) {
9      // By default we use a pretty huge range for the input voltage,
10     // which probably isn't the most appropriate choice for system
11     // that don't use a lot of power.  But all of the calculations
12     // are shown below if you want to change the settings.  You will
13     // also need to change any relevant register settings, such as
14     // setting the VBUS_MAX to 16V instead of 32V, etc.
15
16     // VBUS_MAX = 32V             (Assumes 32V, can also be set to 16V)
17     // VSHUNT_MAX = 0.32          (Assumes Gain 8, 320mV, can also be 0.16,
   0.08,
18     // 0.04) RSHUNT = 0.1               (Resistor value in ohms)
19
20     // 1. Determine max possible current
21     // MaxPossible_I = VSHUNT_MAX / RSHUNT
22     // MaxPossible_I = 3.2A
```

```
23
24      // 2. Determine max expected current
25      // MaxExpected_I = 2.0A
26
27      // 3. Calculate possible range of LSBs (Min = 15-bit, Max = 12-bit)
28      // MinimumLSB = MaxExpected_I/32767
29      // MinimumLSB = 0.000061              (61uA per bit)
30      // MaximumLSB = MaxExpected_I/4096
31      // MaximumLSB = 0,000488              (488uA per bit)
32
33      // 4. Choose an LSB between the min and max values
34      //    (Preferrably a roundish number close to MinLSB)
35      // CurrentLSB = 0.0001 (100uA per bit)
36
37      // 5. Compute the calibration register
38      // Cal = trunc (0.04096 / (Current_LSB * RSHUNT))
39      // Cal = 4096 (0x1000)
40
41      //ina219_calValue = 1785.5274629468177855274629468178; // Changed value
    for RHSUNT = 0.2294 V
42
43      // 6. Calculate the power LSB
44      // PowerLSB = 20 * CurrentLSB
45      // PowerLSB = 0.002 (2mW per bit)
46
47      // 7. Compute the maximum current and shunt voltage values before
   overflow
48      //
49      // Max_Current = Current_LSB * 32767
50      // Max_Current = 3.2767A before overflow
51      //
52      // If Max_Current > Max_Possible_I then
53      //    Max_Current_Before_Overflow = MaxPossible_I
54      // Else
55      //    Max_Current_Before_Overflow = Max_Current
56      // End If
57      //
58      // Max_ShuntVoltage = Max_Current_Before_Overflow * RSHUNT
59      // Max_ShuntVoltage = 0.32V
60      //
61      // If Max_ShuntVoltage >= VSHUNT_MAX
62      //    Max_ShuntVoltage_Before_Overflow = VSHUNT_MAX
63      // Else
64      //    Max_ShuntVoltage_Before_Overflow = Max_ShuntVoltage
65      // End If
66
67      // 8. Compute the Maximum Power
68      // MaximumPower = Max_Current_Before_Overflow * VBUS_MAX
69      // MaximumPower = 3.2 * 32V
70      // MaximumPower = 102.4W
71
72      // Set multipliers to convert raw current/power values
73      ina219_currentDivider_mA = 10; // Current LSB = 100uA per bit (1000/100
    = 10)
74      ina219_powerMultiplier_mW = 2; // Power LSB = 1mW per bit (2/1)
75
76      // Set Calibration register to 'Cal' calculated above
```

```
77      Adafruit_BusIO_Register calibration_reg =
78          Adafruit_BusIO_Register(i2c_dev, INA219_REG_CALIBRATION, 2,
    MSBFIRST);
79      calibration_reg.write(ina219_calValue, 2);
80
81      // Set Config register to take into account the settings above
82      uint16_t config = INA219_CONFIG_BVOLTAGERANGE_32V |
83                        INA219_CONFIG_GAIN_8_320MV |
    INA219_CONFIG_BADCRES_12BIT |
84                        INA219_CONFIG_SADCRES_12BIT_1S_532US |
85                        INA219_CONFIG_MODE_SANDBVOLT_CONTINUOUS;
86      Adafruit_BusIO_Register config_reg =
87          Adafruit_BusIO_Register(i2c_dev, INA219_REG_CONFIG, 2, MSBFIRST);
88      _success = config_reg.write(config, 2);
89    }
90
91
92 ///////////////////////////// CLASS DEFINITION
    ////////////////////////////
93
94 /*!
95  *   @brief  Class that stores state and functions for interacting with
    INA219
96  *  current/power monitor IC
97  */
98 class Adafruit_INA219 {
99 public:
100   Adafruit_INA219(uint8_t addr = INA219_ADDRESS);
101   ~Adafruit_INA219();
102   bool begin(TwoWire *theWire = &Wire);
103   void setCalibration_32V_2A(uint32_t ina219_calValue);
104   void setCalibration_32V_1A(uint32_t ina219_calValue);
105   void setCalibration_16V_400mA(uint32_t ina219_calValue);
106   float getBusVoltage_V();
107   float getShuntVoltage_mV();
108   float getCurrent_mA();
109   float getPower_mW();
110   void powerSave(bool on);
111   bool success();
112   uint32_t ina219_calValue;
113
114 private:
115   Adafruit_I2CDevice *i2c_dev = NULL;
116
117   bool _success;
118
119   uint8_t ina219_i2caddr = -1;
120   //uint32_t ina219_calValue;
121   // The following multipliers are used to convert raw current and power
122   // values to mA and mW, taking into account the current config settings
123   uint32_t ina219_currentDivider_mA;
124   float ina219_powerMultiplier_mW;
125
126   void init();
127   int16_t getBusVoltage_raw();
128   int16_t getShuntVoltage_raw();
129   //int16_t getCurrent_raw(uint32_t ina219_calValue);
```

```
130   int16_t getPower_raw();
131   int16_t getCurrent_raw();
132
133 };
```

Listado 5.1 – *Código en C++*

$$ina219\_calvalue = \frac{0{,}04096}{current_{LSB} * R_{SHUNT}} \tag{5.2.1}$$

$$\tag{5.2.2}$$

- 0.04096: Valor de referencia.
- current_LSB: LSB (Least Significant bit) es el mínimo valor de corriente que el ADC de 12 o 15 bits es capaz de leer. Es obtenido al hacer la media de los valores máximas y mínimos esperados al dividir entre más o menos números de bits del ADC como se ve en la ecuación 5.2.5.
- R_SHUNT: Aquí añadimos el valor de ressitencia de shunt que usa nuestro equipo.

Como vemos en la ecuación 5.2.2, el resultado es un registro de calibración que cuanto más ajustado esté el valor, más exacto será el valor que devuelve el INA219.

$$Max\_current\_LSB = \frac{Max_{current}}{2^{12}} \tag{5.2.3}$$

$$Min\_current\_LSB = \frac{Max_{current}}{2^{15}} \tag{5.2.4}$$

$$\tag{5.2.5}$$

5

En el BMS.cpp tenemos este valor como cal_value el cual es el cálculo ya realizado de la 5.2.2.

5.2.3. Documentación de código

En el apéndice B se comenta todo lo relacionado con la documentación de código mediante la herramaienta Doxygen y su extensión Graphviz que nos ayuda a obtener gráficos intuitivos de muchos tipos, para más información consultar el mencionado apéndice D.

5.3. Diseño mecánico

El material usado para la imprresión ha sido PLA que es el más utilizado en la impresión 3D. Este filamento no produce olores durante su fusión en el extrusor (boquilla de salida de filamento fundido) de la impresora y proporciona detalles nítidos y una buena calidad. Tiene menor resistencia mecánica en comparación con el ABS como se puede ver en la tabla 5.1. En un principio, el material escogido para realizar la impresión fué ABS. Sin embargo, este material es muy sensible a cambios de temperatura y se debe de refrigerar lo menos posible el lugar de impresión, si esto sucede, puede despegarse la pieza de la cama aunque las temperaturas de impresión sean las acertadas; 235°C para el extrusor y 90 para la cama (valores correctos como se puede ver en la tabla 5.1). Esto ha hecho que use PLA con el que no he tenido ningún tipo de problema.

Como podemos ver en la imagen 5.56a, el material no era capaz de adherirse a la cama y cuando se despega la impresión posterior es incorrecta por lo que la pieza se debe de tirar y tener las consecuentes pérdidas económicas

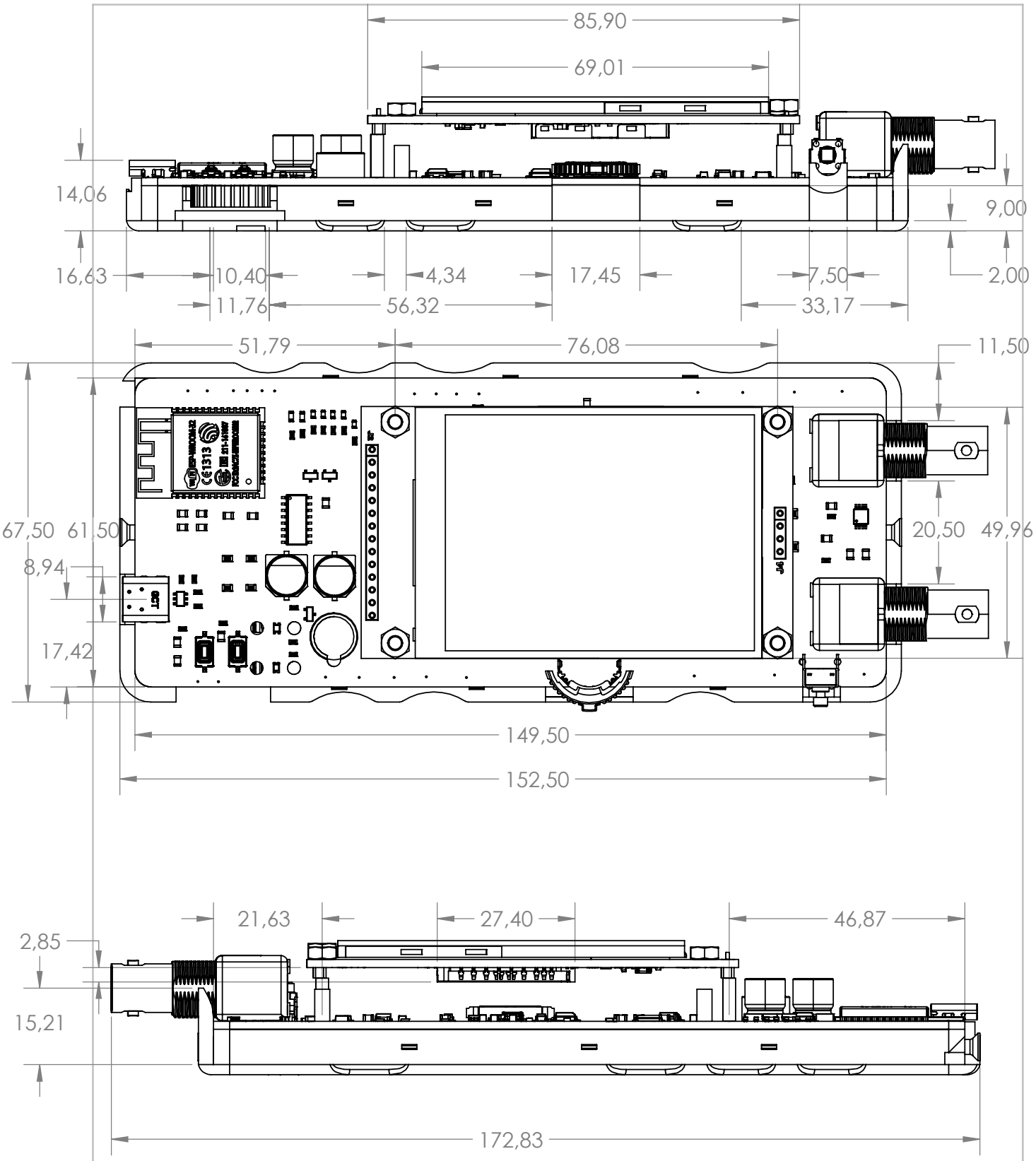
y de tiempo. Posteriormente con el uso de PLA la impresión fué perfecta, la adesión se mantuvo y el resultado es el esperado. En la figura 5.56b, podemos ver con el software Cura como la impresora va a imprimir capa por capa la pieza. Esta fase es principal para ver si nuestra pieza necesita de soporte. Puede que en el diseño hayamos impuesto zonas voladizas las cuales son imposibles de imprimir ya que no hay filamento debajo que soporte la capa superior. Para este caso, en la fase de diseño se procuró evitar usar soportes ya que estos deben de ser muy bien escogidos y ajustados para que al retirar el soporte no arranque filamento de la pieza. En mi experiencia personal, los soportes deben de ser evitados y si son necesarios, se debe ajustar muy bien angulo de voladizo, elegir la forma de arbol y poner tan solo 1 o 2 capas de filament0 entre el final del soporte y la pieza.

Característica	PLA	ABS
Temperatura de Impresión	180°C - 220°C	220°C - 250°C
Temperatura de la Cama	20°C - 60°C (opcional)	80°C - 110°C
Resistencia Térmica	Baja (hasta 60°C)	Alta (hasta 100°C)
Facilidad de Impresión	Alta	Media
Resistencia Mecánica	Media	Alta
Flexibilidad	Baja	Media
Biodegradabilidad	Sí	No
Emisión de Vapores	Baja	Alta (recomendado ventilación)

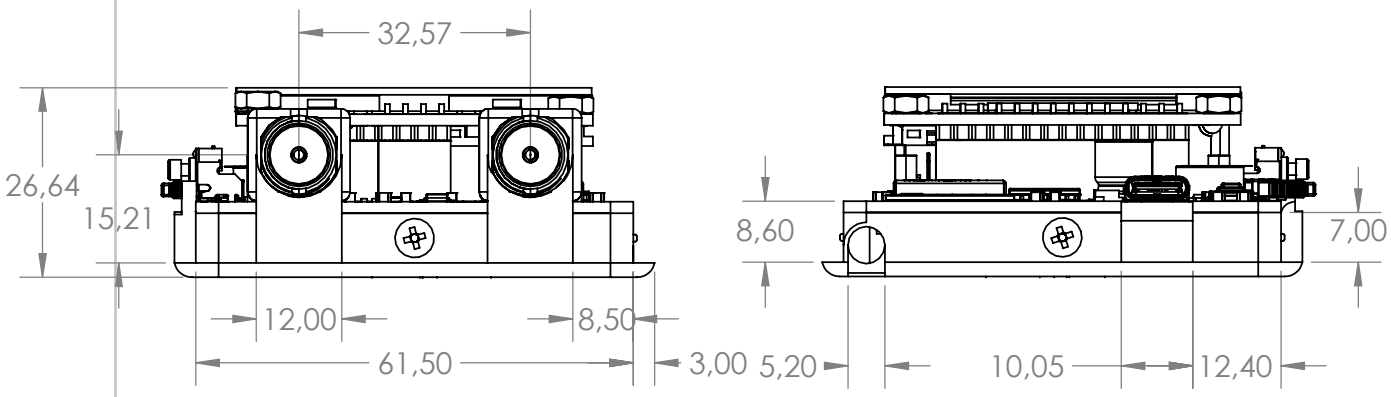
Tabla 5.1 – *Comparación de características entre PLA y ABS.*

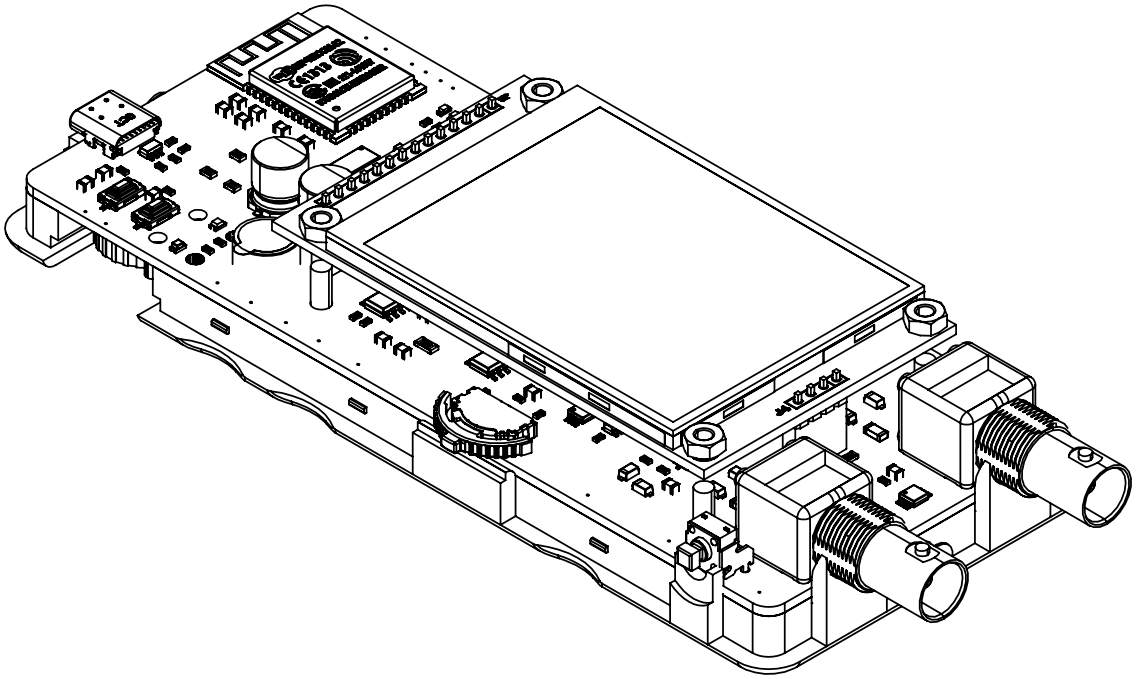
Característica	Kingroon KP3S
Tecnología de Impresión	FDM (Modelado por Deposición Fundida)
Volumen de Construcción	180 x 180 x 180 mm
Diámetro de Filamento	1.75 mm
Tipo de Filamento	PLA, ABS, PETG, TPU
Diámetro de la Boquilla	0.4 mm
Precisión de Impresión	±0.1 mm
Velocidad de Impresión	20 - 100 mm/s
Temperatura Máxima del Hotend	260°C
Temperatura Máxima de la Cama	110°C
Nivelación de la Cama	Manual
Pantalla	Pantalla táctil LCD
Conectividad	Tarjeta SD, USB
Estructura	Marco de aluminio
Extrusor	Direct Drive
Reanudar Impresión	Sí
Fuente de Alimentación	24V 240W
Dimensiones de la Máquina	280 x 285 x 370 mm
Peso	7 kg

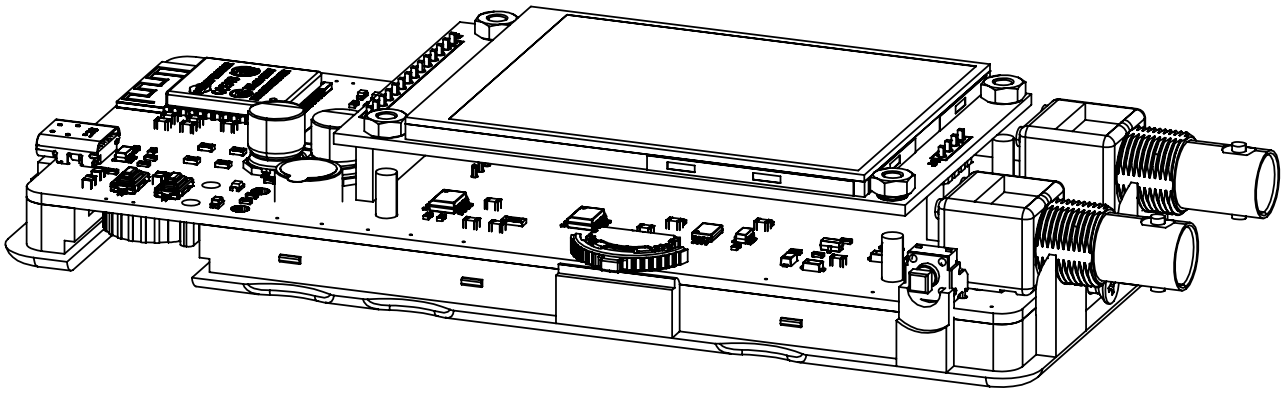
Tabla 5.2 – *Características de la impresora 3D Kingroon KP3S.*



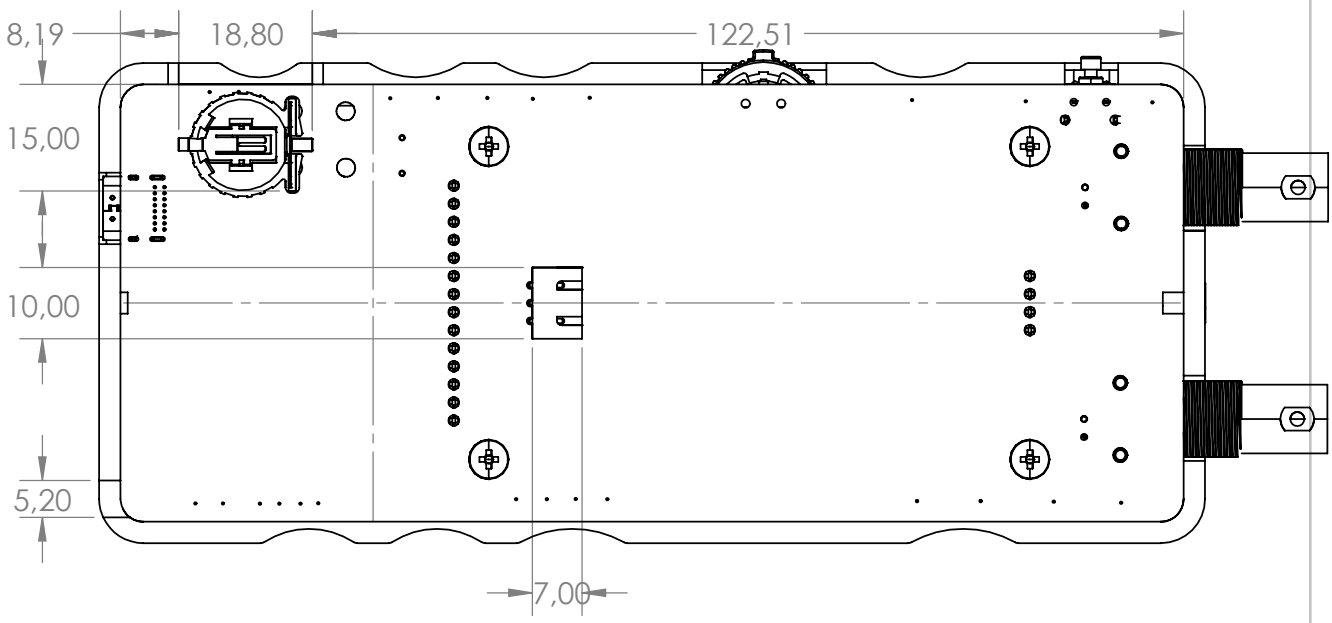
Dibujado: Juan Manuel Martín Lucena	**Grupo:**	**Fecha:** 28/08/2024		
Revisado:		**Material:**		
PLANTA Y PERFILES		**Formato:** DIN A4	**Escala:** 1:1	**Proyección:**

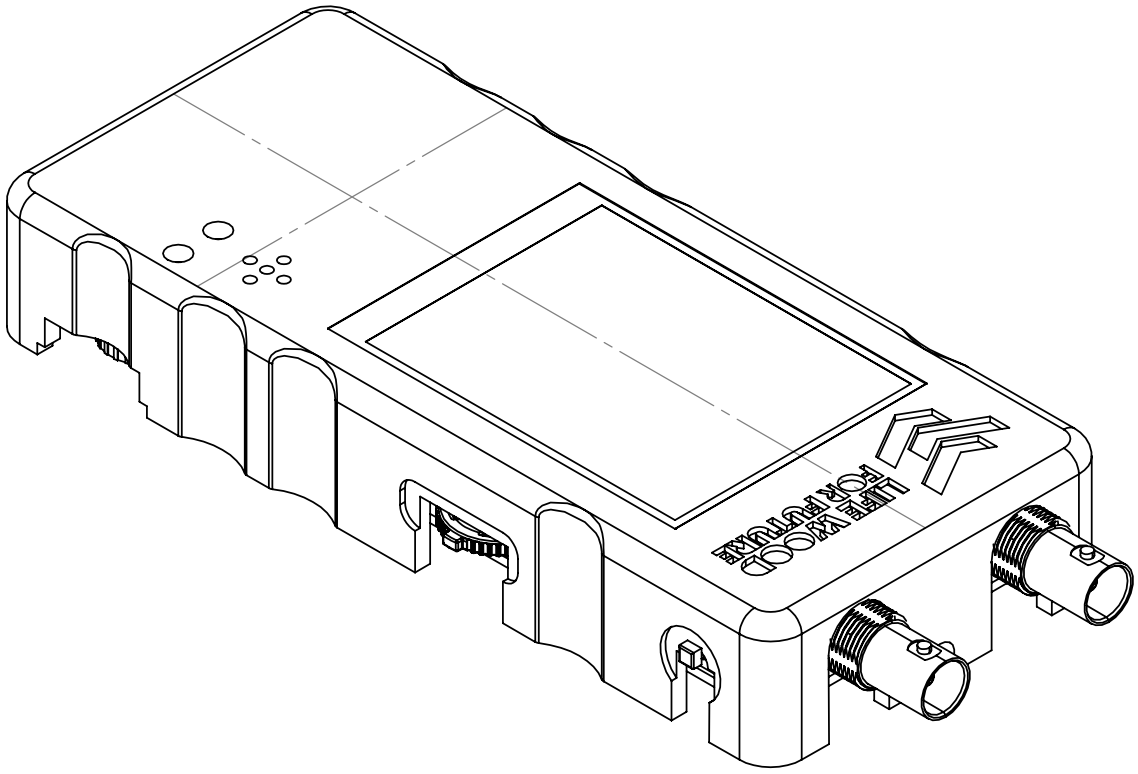




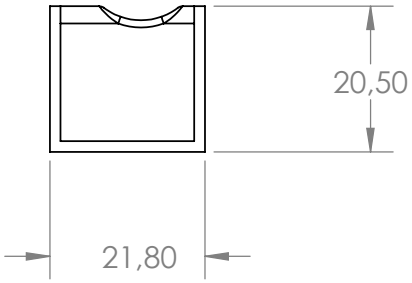


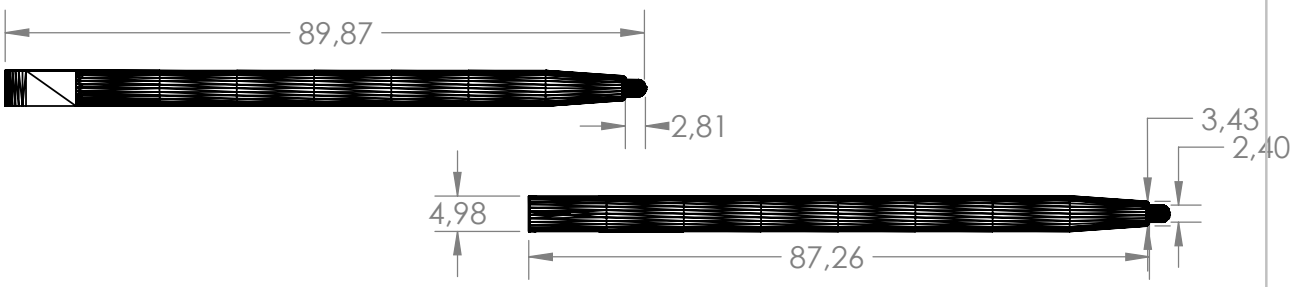
Dibujado: Juan Manuel Martín Lucena	**Grupo:**	**Fecha:** 28/08/2024	
Revisado:		**Material:**	
SUPERIOR E INFERIOR JUNTO A ISOMÉTRICA Y TRIMÉTRICA	**Formato:** DIN A4	**Escala:** 1:1	**Proyección:**

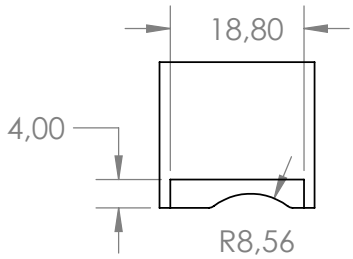


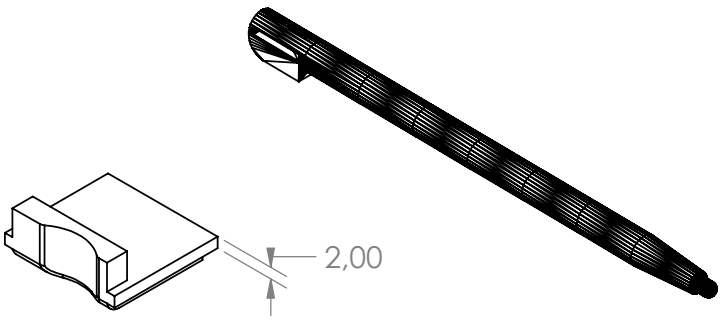


Dibujado: Juan Manuel Martín Lucena	**Grupo:**	**Fecha:** 28/08/2024	
Revisado:		**Material:**	
PLANTA	**Formato:** DIN A4	**Escala:** 1:1	**Proyección:**

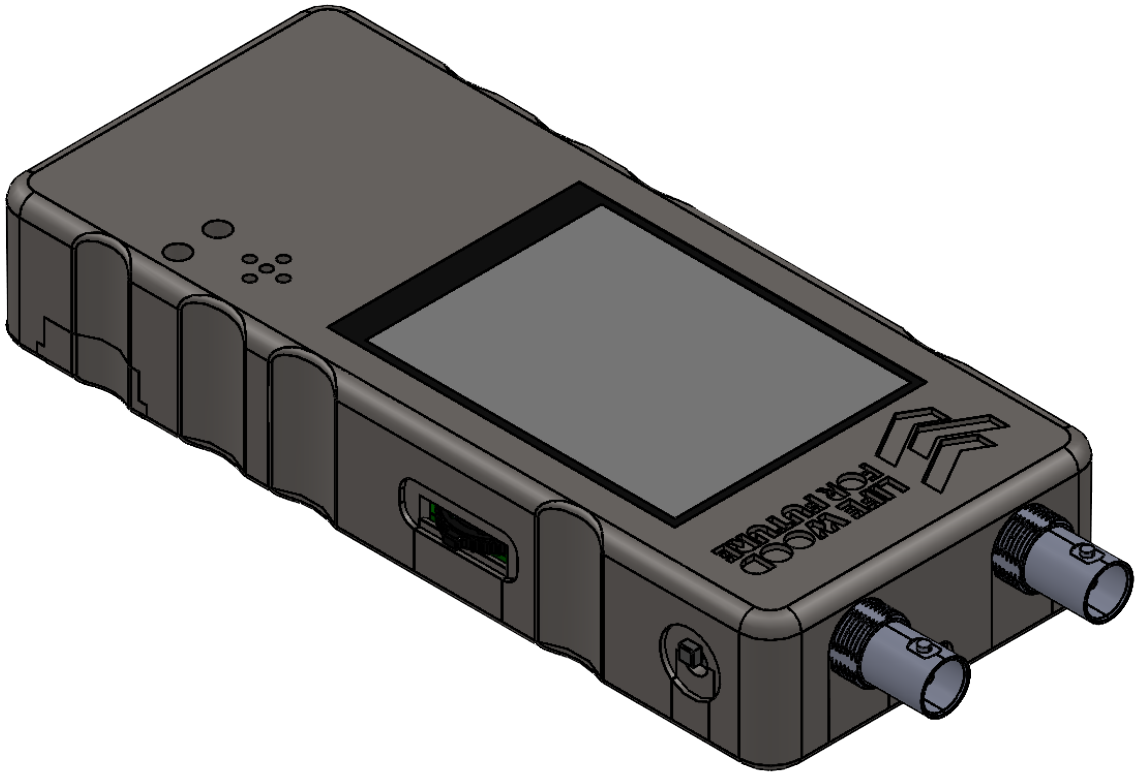


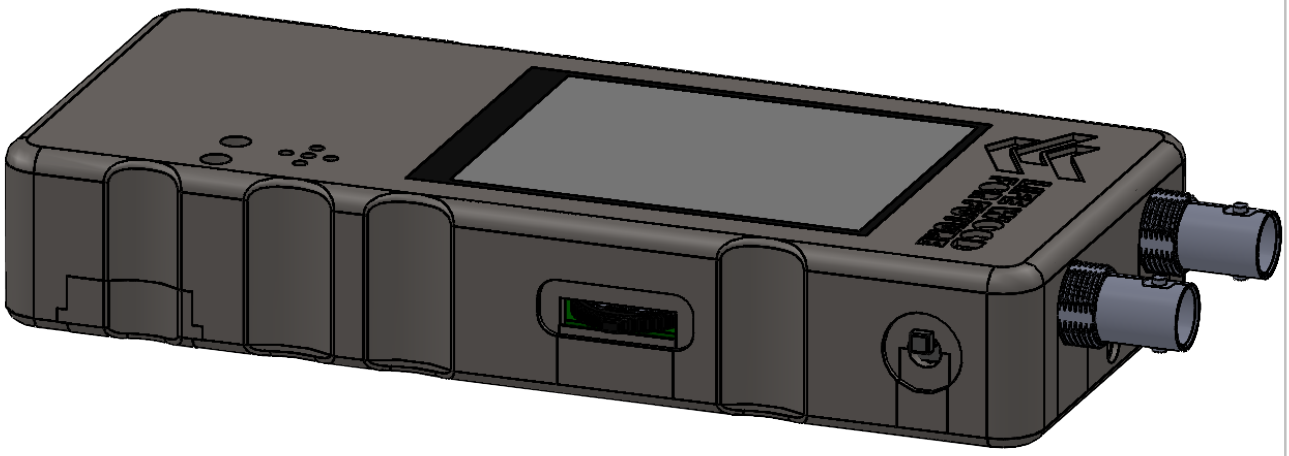






Dibujado: Juan Manuel Martín Lucena	**Grupo:**	**Fecha:** 28/08/2024	
Revisado:		**Material:**	
LÁPIZ Y TAPA DE PILA	**Formato:** DIN A4	**Escala:** 1:1	**Proyección:**

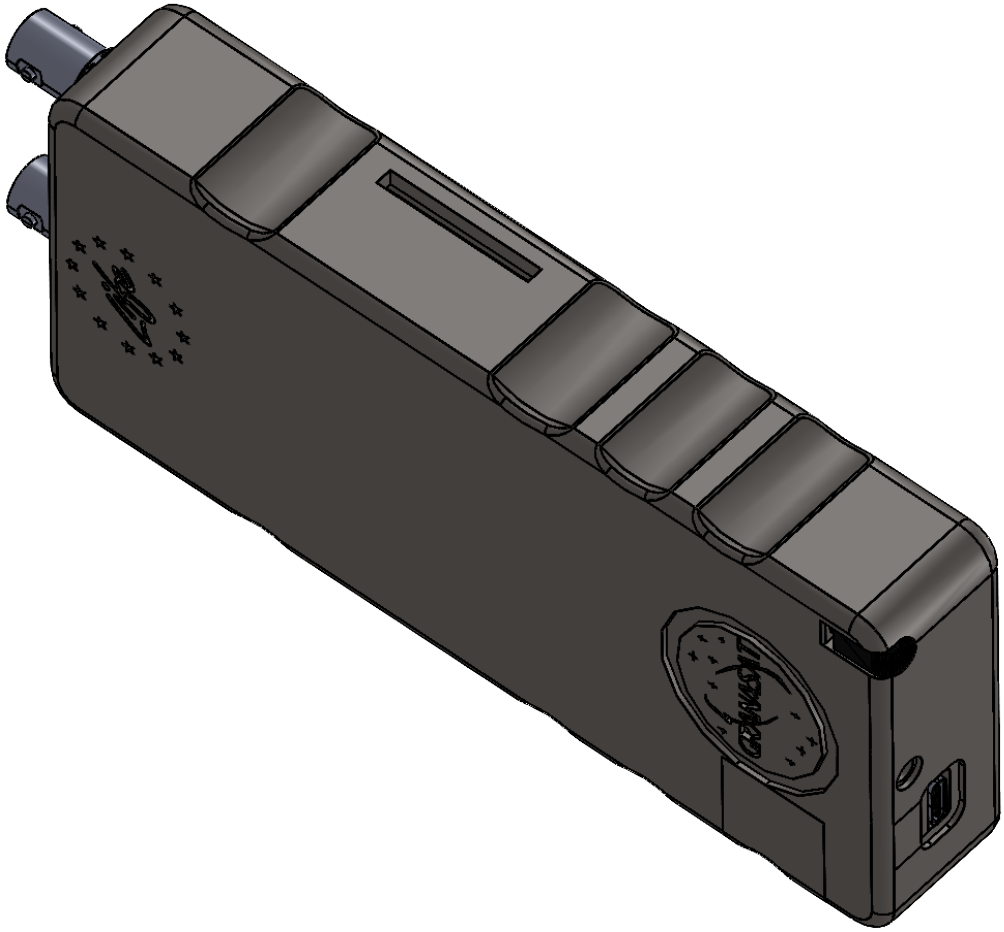




Dibujado:	Grupo:	Fecha:	
Juan Manuel Martín Lucena		28/08/2024	
Revisado:		**Material:**	

MODELO REAL

Formato:	Escala:	Proyección:
DIN A4	1:1	



Dibujado: Juan Manuel Martín Lucena	**Grupo:**	**Fecha:** 28/08/2024	
Revisado:		**Material:**	
MODELO REAL	**Formato:** DIN A4	**Escala:** 1:1	**Proyección:**

5.3.1. Ensamblaje

Para poder realizar el ensamblaje es importante ajustar la expansión horizontal de una de las piezas, este es un ajuste del software que nos permite definir cierta holgadez en piezas que deben de encajar como es nuestro caso. Una expansión horizontal positiva corrige agujeros y detalles grandes, haciendo disminuir la pieza, mientras que, con expansión horizontal negativa corregimos huecos y espacios ajustados agrandando la pieza. Yo le he dado -0.2mm a la carcasa superior y la inferior la he dejado por defecto. Con este ajuste encaja a la perfección una pieza con otra, sin prácticamente necesidad de usar los tornillos métrica 3 de la parte superior e inferior. La carcasa es robusta y firme sin separación entre las partes.

Como ya he mencionado, la carcasa es sólida entre una parte y otra. Para asegurarlo se usan dos tornillos y cabezales que se insertan en la carcasa para fijarlos como se ve en la imagen 5.57a. Para buscar la óptima forma de anclaje he impreso distintas formas más sofisticadas que proporcionen deslizamiento entre las uniones como se ven en las imágenes 5.57a, finalmente se han desechado ya que el resultado no era el esperado.

Los anclajes impresos han consistido en unas pequeñas muecas salientes en cada perfil de la pieza inferior y endiduras en la superior como se ve en ña figura 5.58a. Son más simples que los anteriores y aportan un excelente resultado.

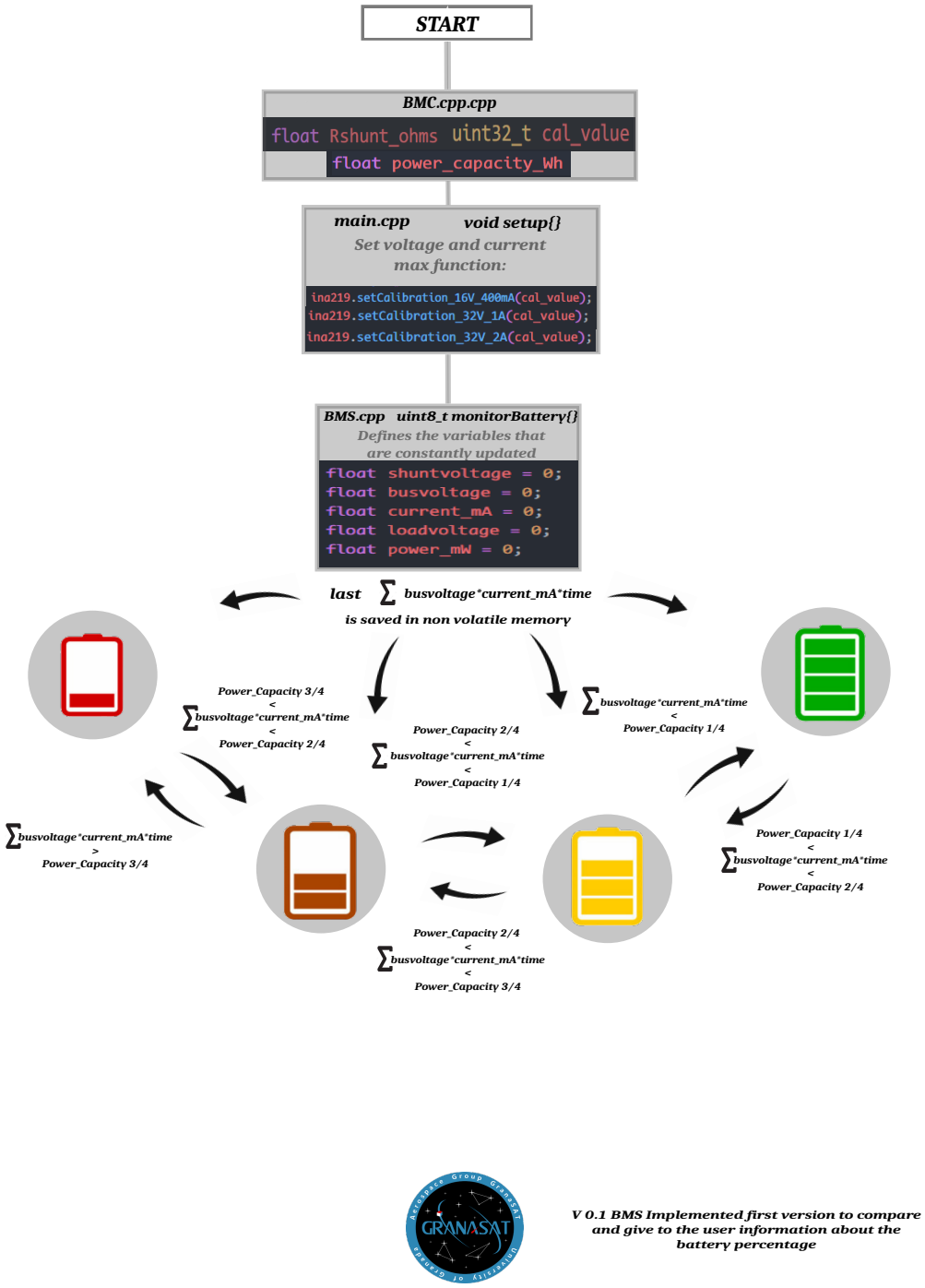


Figura 5.49 – *Forma ideal de monitorizar la batería.*

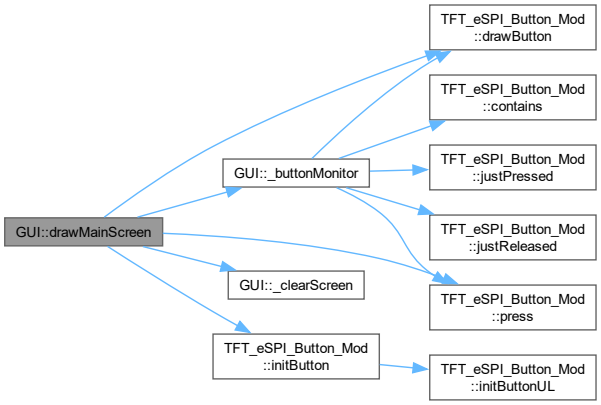


Figura 5.50 – *Accesos necesarios de clases desde la pantalla principal.*

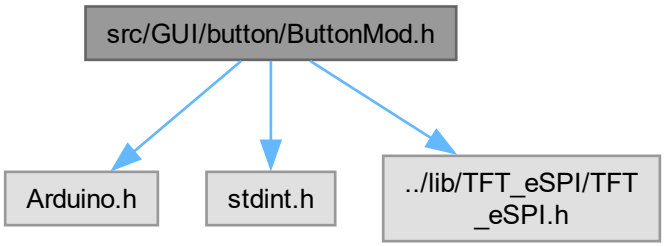


Figura 5.51 – *Librerías de las que depende la función de botón implementada.*

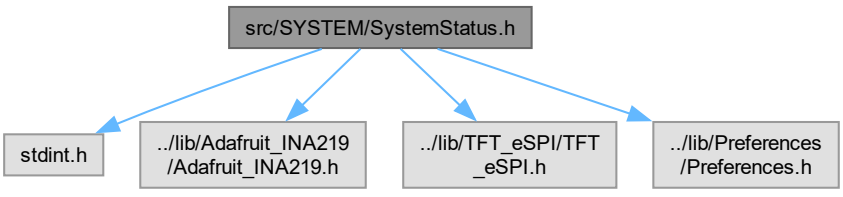


Figura 5.52 – *Accesos necesarios desde las definiciones de sistema.*

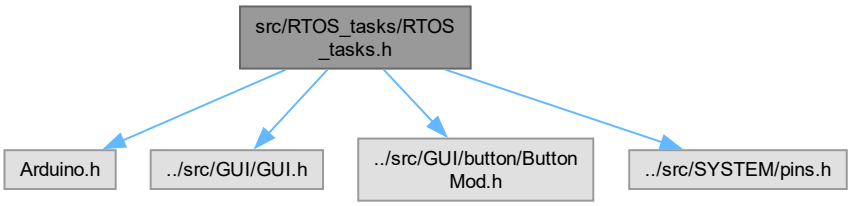


Figura 5.53 – *Accesos necesarios para las tareas de RTOS.*

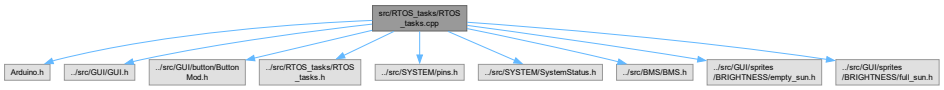


Figura 5.54 – *Accesos necesarios para ejecutar las tareas de RTOS.*

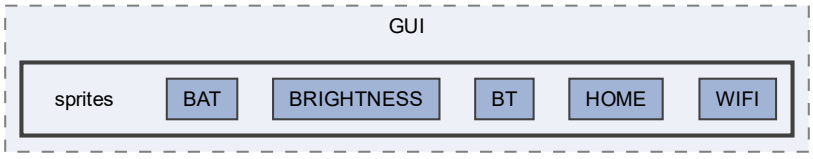
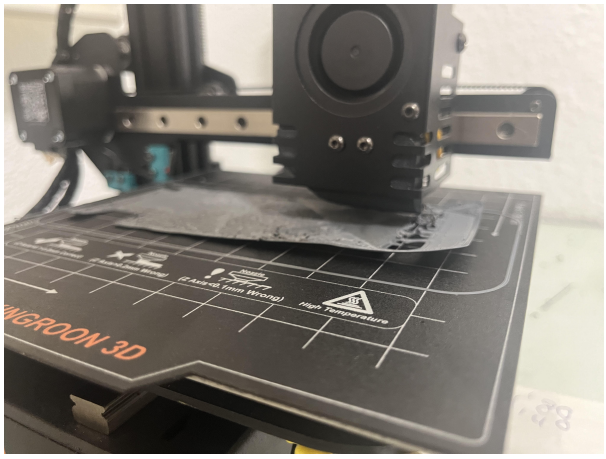
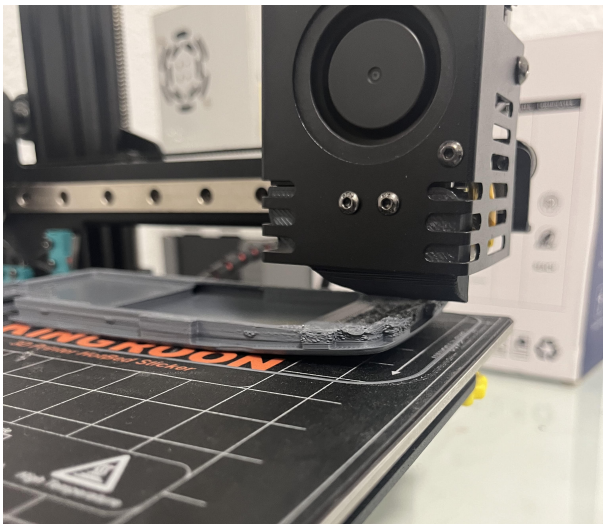


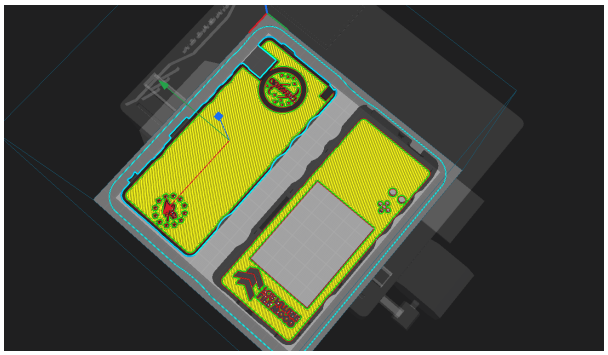
Figura 5.55 – *Logos necesarios del sistema de interfaz gráfica.*



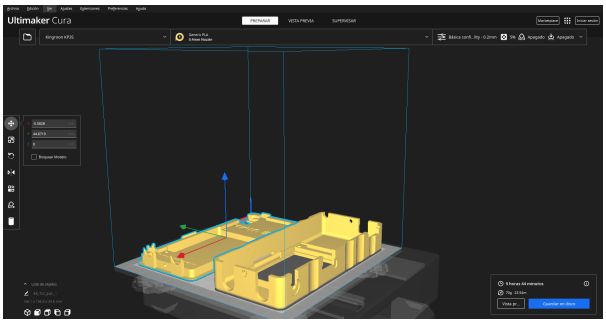
(a) *Primer intento de impresión con 220°C en el extrusor y 80°C en la cama usando ABS.*



(b) *Segundo intento de impresión con 235°C en el extruso y 90°C en la cama usando ABS.*



(a) *Preajuste en Cura para generar el archivo gcode que leerá la impresora 5.56.*



(b) *Avance de capas en la previcualización.*

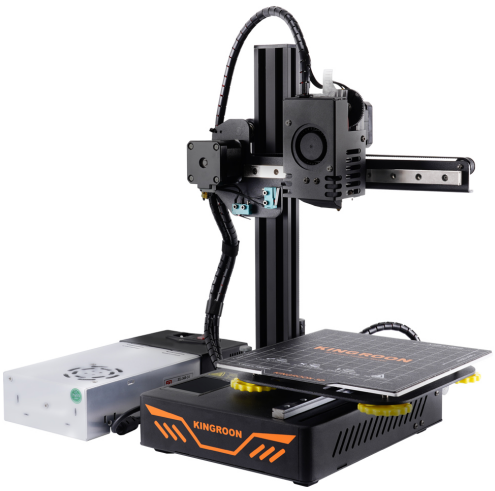
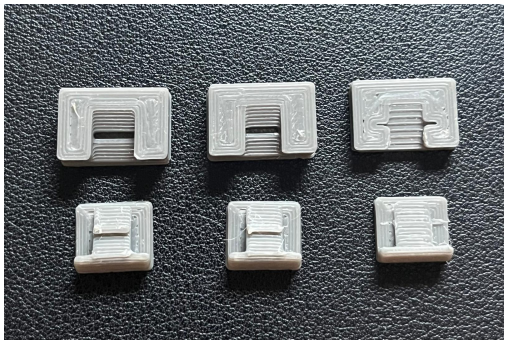
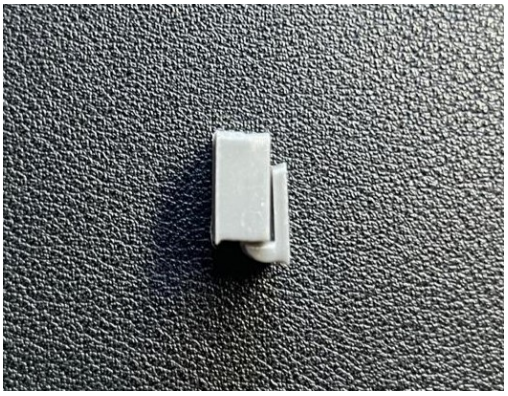


Figura 5.56 – *Impresora propia Kingroon KP3S.*



(a) *Distintas impresiones para buscar el mejor anclaje entre la parte superior e inferior.*



(b) *Anclaje conjunto*

(a) *Tornillo de cabeza plana M3 ISO 7046.*

(b) *Roscado RS PRO 278-534.*

Figura 5.57 – *Tipos de anclajes impresos.*

(a) *Hendiduras del perfil izquierdo para anclaje en la carcasa superior.*

(b) *Detalles del perfil izquierdo para anclaje en la carcasa inferior.*

Figura 5.58 – *Anclajes finales de TIK Modelo 2.*

Figura 5.59 – *Vista frontal del ensamblaje final de TIK Modelo 2.*

Figura 5.60 – *Vista superior del ensamblaje final de TIK Modelo 2.*

Figura 5.61 – *Vista trasera del ensamblaje final de TIK Modelo 2.*

Figura 5.62 – *Vista lateral del ensamblaje final de TIK Modelo 2.*

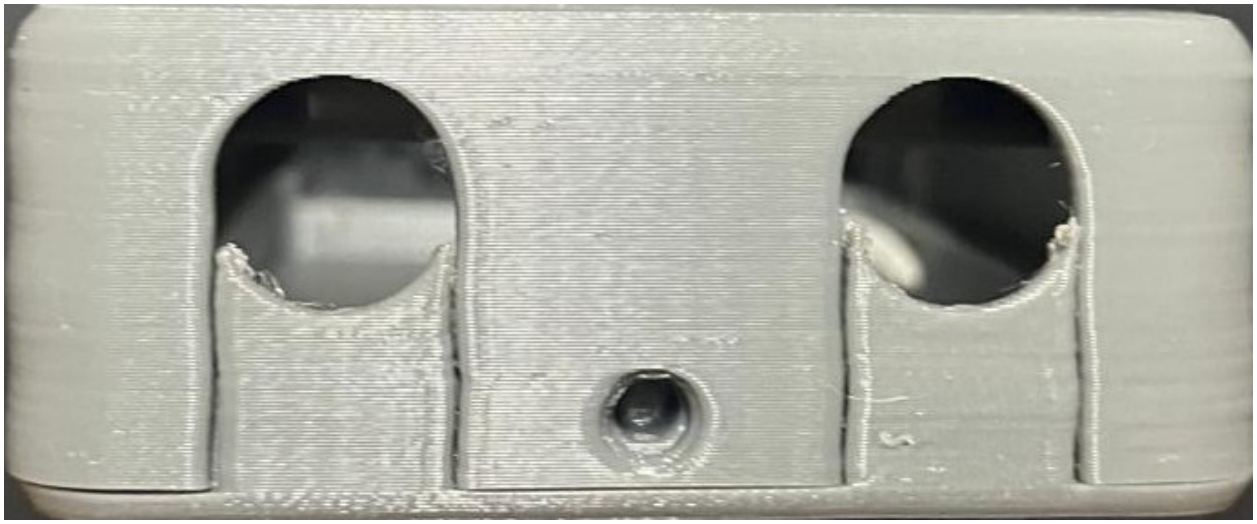


Figura 5.63 – *Vista frontal detallada del ensamblaje final de TIK Modelo 2.*

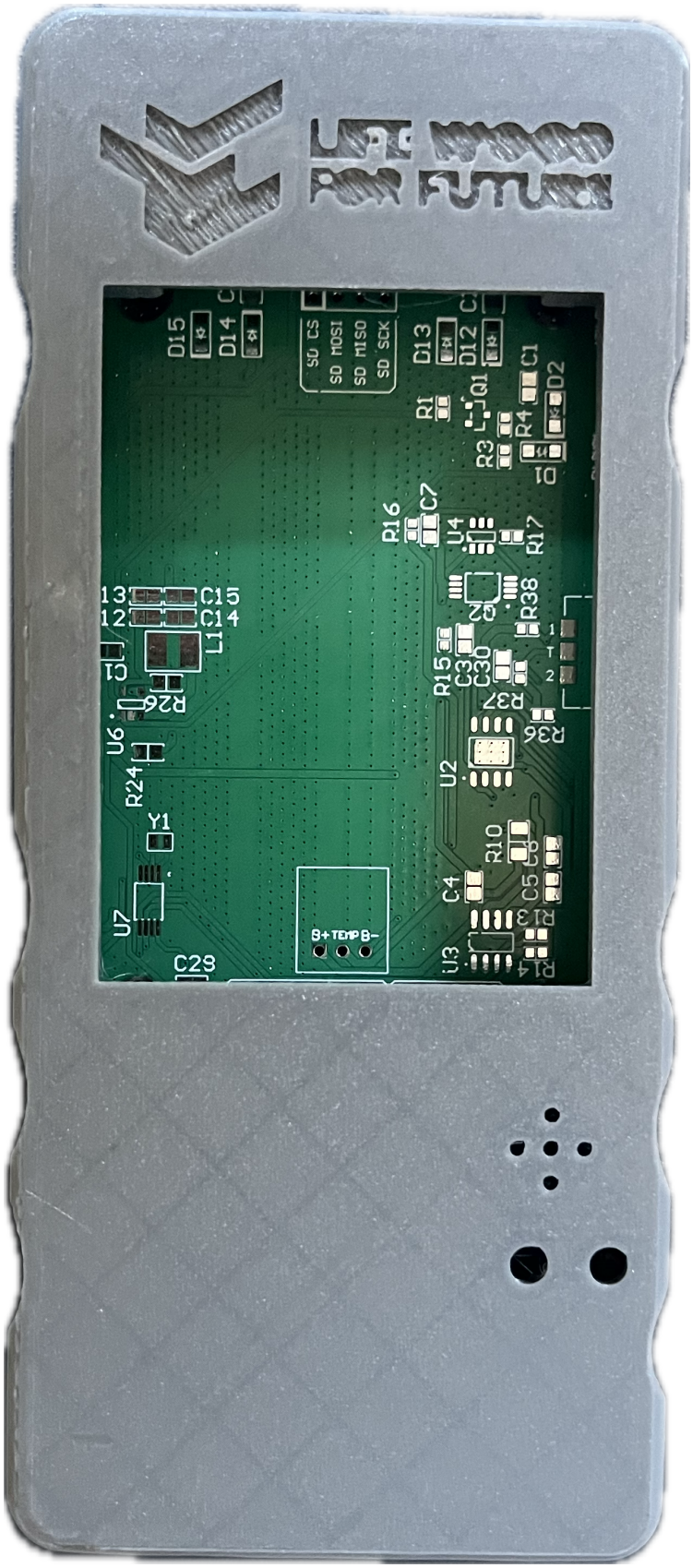


Figura 5.64 – *Vista del ensamblaje real de TIK Modelo 2.*

Capítulo 6

Test y validaciones

- Electrónica: El diseño electrónico del TIK Modelo 2 no ha sido sometido a pruebas exhaustivas, lo cual se debió principalmente a restricciones de tiempo durante la fase final de desarrollo del proyecto. Esta tarea de verificación queda como una labor pendiente y de vital importancia para el próximo alumno que retome el trabajo. La validación de los circuitos, la correcta integración de los componentes y la optimización del consumo energético son algunos de los aspectos que deberán ser abordados en futuras etapas para garantizar el correcto funcionamiento del dispositivo.
- Software: En cuanto al software, a pesar de que la plataforma no ha sido testeada en su totalidad en el nuevo hardware, el código ha sido desarrollado de manera modular y portable. Esto asegura que podrá ser implementado sin mayores complicaciones en el TIK Modelo 2 una vez que la parte electrónica haya sido verificada. La estructura flexible del software permite su adaptación y optimización para cualquier actualización futura del hardware.
- Mecánica: Por otro lado, el ensamblaje mecánico del TIK Modelo 2 ha sido cuidadosamente verificado y validado. Todas las piezas se han ajustado y ensamblado correctamente, cumpliendo con los requerimientos de diseño original. Esto garantiza que la estructura física del dispositivo es robusta y que los componentes se mantendrán en su lugar durante su funcionamiento. La correcta verificación del ensamblaje permite asegurar que el dispositivo estará listo para ser utilizado una vez que las pruebas electrónicas y de software se completen.

Además, en el capítulo 7, se detallan futuras mejoras y datos a tener en cuenta en la fase de test y validaciones.

Capítulo 7

Conclusiones y futuras mejoras

En este capítulo se va ha detallar los objetivos alcanzados y futuras mejoras que se pueden realizar en las siguientes versiones realizadas por estudiantes de los grados de electrónica, telecomunicaciones e informática. También en un proyecto es importante hacer autocrítica de errores y justificar el camino que debe tomar el proyecto ambicioso que haye entre manos por parte de la UGR.

7.1. Hitos alcanzados

Aquí hago un breve repaso sobre los logros conseguidos:

- Tomar un dispositivo desconocido por conmpleto y ser capaz por parte del estudiante de identificar errores superándolos para diseñar una siguiente versión mejorada.
- Se ha conseguido manejar y estudiar como funcionan las señales digitales y analógicas en un sistema embebido como el presente. Además de como un producto debe ser allimentado y manejar eficientemente la energía disponible haciendo hincapié en un dispositivo portátil.
- Manejar y desarrollar un sistema operativo a tiempo real como es RTOS con su estudio pertinente del manejo de memoria y de los tipos de memoria. Además, de comprender como las tareas puedden ejecutarse eficientemente cuando tenemos un microcontrolador de varios núcleos.
- Tomadas las bases del proyecto de la anterior versión, contar ocn la capacidad de evaluar que es necesario y que no, así como desechar y modificar partes del diseño que ayudan a las siguientes versiones.
- Manejo de los programas de diseño asistido por ordenador CAD y EDA (SolidWorks y Altium Designer) para actualizar los desarrollos en los entornos de diseño que manejan las empresas tecnológicas actuálmente.
- Documentar todos los diseños de forma profesional. Tanto los diseños mecánicos como los diseños electrónicos.
- Comprensión y manejo de instrumentos electrónicos del laboratorio de GranaSAT. Además, del control remoto por comandos haciendo que el estudiante se inicie en la instrumentación electrónica de equipos de laboratorio.
- Colaboración entre estudiantes en un entorno de compañerismo en los laboratorios de GranaSAT mejorando las dotes comunicativas entre un equipo formado por 3 integrantes que trabajan conjuntamente en el proyecto LIFE Wood For Future de la UGR.
- Inciación en la documentación de código con Doxygen manejando así el control de cabecera de las funciones, clases y definiones en un código embebido.

7.2. Evaluación personal

EL proyecto realizado ha necesitado de 10 meses de duro trabajo y de compaginación con el último año académico. Sin embargo, lo aprendido durante este año a nivel educativo y profesional tiene un valor incalculable. Estoy muy satisfecho por el trabajo que he logrado hacer contando con todo el conocimiento que he necesitado adquirir para desarrollar el proyecto.Involucrarme en un equipo de trabajo con un proyecto real como es LIFE Wood For Future me ha hecho darme cuenta de lo que soy capaz con trabajo constante.

En el ámbito profesional he sentido lo que es trabajar en un equipo de trabajo como puede ser el de una empresa tecnológica. Donde he podido percibir lo importante que osn las fechas de finalización de una etapa y documentarla. Es esencial acabr una tarea y no arrastrarla, ya que la eficiencia en el manejo de estas es esencial. No acabr una tarea e ir arrastrándola mientras te involucras o inicias en otras hace que personalmente sientas que tu rendimieto no es el adecuado. Por ello, he procurado en la medida de lo posible estar acabando una tarea o tenerla prácticamente acabada una vez que inicaba otra. La multitarea es un aspecto también novedoso para mí en el ámbito profesional. Manejar dos o tres tareas a la vez y distribuir la jornada entre ellas ayuda a ser más eficiente y es un error obcecarse con una de ellas que no sale. Para ello, lo que viene mejor es cambiar de tarea a una menos exigente y distribuir el día en función de la complejidad de las mismas.

Siento que he crecido enormemente como profesional por mi capacidad de manejar programas profesionales para el diseño como lo son Altium designer para la electrónica y SolidWorks para el diseño mecánico y electrónico. Esto me hace destacar somo multidisciplinar en el campo de la tecnología. Hace prácticamente un año no tenía manejo sobre ninguno de ellos y tras un año de esfuerzo, siento que mi capacidad de diseño de PCBs tiene ahora una base sólida, al igual que con los ensamblajes mecánicos. En el ámbito de la documentación, sumergirme en LATEX de una forma organizada me ha hecho capaz de detallar en profundización el trabajo que realizo. En el campo de desarrollo de Firmware he adquerido un salto enorme y más adentrándome en un mundo totalmente desconocido para mí como es la programación en tiempo real y de forma parecida el lenguaje de programación usado C++. Antes del proyecto, desconocía por completo la organización de un proyecto en ficheros del lenguaje mencionado, que significada hacer portable un código, ni sabía que eran herramientas de programación tan importantes como lo son las clases y los punteros. Y ahora puedo decir que conozco todo ello y además soy capaz de desarrollarlo como en el presente proyecto.

Sinceramente deseo todo lo mejor en el proyecto LIFE Wood For Future y que encuentren alumnos de la talla del anterior alumno que inicó toda la base del proyecto TIK para terminar con el desarrollo del proyecto y alcanzar la noble cause que persigue.

7.3. Errores personales

Hacer autocrítica es esencial para entender el por qué de algunos pasos seguidos que no han germinado finalmente en un mejor resultado.

Para empezar, mis nociones de programación son muy básicas. En el grado hay un par de asignaturas que incian en el lenguaje de C++ pero no son suficientes para el presente proyecto. Un alumno de ingeniería electrónica que quiera hacer frente a un proyecto de tal envergadura necesitará tiempo de estudio para comprender como se organiza un proyecto de programación. Siento que aún con el trabajo volcado, el desarrollo del Firmware podría haber sido mejor. Deseo que el siguiente alumno que continúe el trabajo sea consciente de ello.

Por otro lado, la organización de tareas ha sido un punto esencial y que me ha costado hacerme a ella no siendo hasta el final del proyecto cuando he entendido realmente lo importante del manejo de tareas. Habitualmente, me obsesionaba con algo que no conseguía realizar haciendo que me frutrara y que mi trabajo no fuese el más eficaz durante algunas mañanas de trabajo. Finalmente, comprendí que normalmente hacer algo nuevo implica que no salga a la primera pero que el estudio sosegado del problema hace que todo salga adelante.

7.4. Propuesta de futuras mejoras

Realmente, el proyecto ya se encuentra acabado y el hardware necesario para ello como versión final considero que es la presente. Sin embargo, hay ciertas mejoras que por la complejidad de las mismas dan para una versión más de trabajo de fin de grado. A continuación voy a detallar mejoras que he intentado aplicar o he pensado pero por cuestiones de tiempo o dificultad para mí han sido imposibles de implementar.

7.4.1. Mejoras en el hardware

La mejora fundamental que no ha sido posible realizar por cuestiones de tiempo, es integrar el trabajo realizado por Antonio en cuanto a la adquisición de señales en mi código de RTOS. Esto es fundamentar para unificar todo el trabajo realizado este año y poder tener el equipo prácticamente en plena funcionalidad.

Si se desease hacer una segunda versión por nuevas ideas o cambio de rumbo del proyecto en aspectos puntuales, mis propuestas de mejora son:

- Reducción del tamaño de la PCB: El tamaño del producto se ha hecho mayor que la anterior versión en un par de centímetros. Ello se debe los condensadores electrolíticos añadidos para asegurar una tensión estable de polarización de los BNCs. Sin embargo, además del esfuerzo realizado para poner más resistencias y capacitores de 0603 en vez de 0805 se podrían usar integrados de menr tamaño pero ahí dependía directamente del stock en el laboratorio de GranaSAT. También, usar un módulo de pantalla que esté directamente soldado en la PCB nos ayudaría reducir las dimensiones.
- Mejorar la integridad de la PCB con 4 capas: Añadir capas para la potencia de alimentación, señales digitales y/o analógicas ayuda a la integridad e las señales dentro de nuestro equipi haciendo más precisas todo tipo de señales además de eliminar ruido en la alimentación que puede afectar a los integrados.
- Mejorar el control y monitorización de la batería: Integrar una medición más precisa apoyada de la memoria no volatil del microcontrolador nos ayudaría a mostrar en pantalla el estado de la batería de forma precisa, y no depender de la curva de batería obtenidas ya que estas varían con el paso del tiempo.
- Mejorar el convertidor USB-UART: Esto se comentó en la anterior versión. El convertidor CH340C es algo antiguo y consume demasiado para un producto como el presente donde deseamos que la gestión de la batería es esencial. Los módulos de ESP32-DevKitC utilizan el puente CP2102N, lo cual es una excelente opción para una versión futura.
- Usar módulo de pantalla independiente del lector SD: El módulo de pantalla ILI9341 es excelente ya que integra mediante SPI sin embargo, no obliga a tener una loxalización de la SD además, de necesitar un adaptados para micro SDs. Por ello hacer independiente el lector de SD mejoraría y actulizaría nuestro equipo. Además esto ayudaría a integrar protección ESD a la línea de alta velocidad SPI.
- Reductor DC/DC de arranque suave para la pantalla: Como se ha detallado en el capítulo sección 3.4 optar por un regulador DC/DC como el de Texas Instruments TLV62568 que suministra la corriente de forma gradual evitando los parpadeos del encendido de pantalla.

7.4.2. Mejoras en el firmware

- Añadir distintos lenguajes: En un principio se planteó la idea de añadir más lenguajes a parte del inglés que es el presente pero finalmente no se materializó. Hacer esto posible daría un gran saltoa a la actualidad a nuestro equipo.
- Mejorar la interfaz gráfica con LGVL: LGVL (light and Versatile Embedded Graphics Library) es una librería open source que incluye multitud de opción de interfaz y actualiza enormemente la apariencia de nuestro equipo. Además, se encarga de optimizar el núcleo donde se está ejeccutando la tarea haciendo que sea muy fluida su visualización.

7.4.3. Mejoras en el diseño mecánico

- Resistencia al polvo y a la humedad: Al incicio del diseño mecánico se planteó esta opción que es necesaria, ya que cuando el equipo se encuentre a la intemperie este recibirá sin duda polvo y humedad dependiendo en que zona de la vega se encuentre, por ello rediseñar la carcasa y añadir una junta tórico en cada zona abierta es una buena idea.

Apéndice A

Esquemáticos

BACHELOR'S DEGREE IN INDUSTRIAL ELECTRONICS ENGINEERING

Bachelor's Thesis

ACADEMIC COURSE 2023/2024

Tree Inspection Kit

handheld device model 2

AUTHOR:

Juan Manuel Martín Lucena

SUPERVISED BY:

Sr. Andrés Roldán Aranda

DEPARTMENT:

Electronics and Computer Technology

UNIVERSIDAD DE GRANADA

GRANASAT

UIMA

LIFE WOOD FOR FUTURE

Project title: TIK_HandheldDevice_Model2.PrjPcb		
Date: 07/06/2024	Revision: V1.5	Sheet 1 of 12

Index.

LEDS legend:

- Digital signals
- Power bus
- Battery charge completed
- Battery charging in process

GND TEST POINTS

There are 4 GND test points to facilite the testing stage

Notes legend:

- Descriptive/informative
- Changes from previous to current version
- Validation note of correct operation
- Failure/Caution validation note

Changelog

V1.0 01/06/2024

Same schematics of previous version V0.7 except:

- Added connection current consumption.
- Revision ESP32 conecctions (view ESP32 schematic).
- Changed Voltage supply of INA219.
- BAT_CHRG and BAT_STBY pins to TP4056 charger.
- NiMH/NiCd battery charger IC removed and external PNP BJT current driver.
- Type usb changed, from mini B to C.
- Control of brightness added to LCD TFT touchscreen.
- Added real time clock and buzzer schematics.

V1.1 07/06/2024

- Added 3V3 to 1V8 voltage divider in case therefore we wouldn`t need the LDO.
- Revision of notes.
- Doubt whether these schematics will be intended for a joint pcb with Irene or a single device.
- Power width constrains added.
- Added footprint dimensions.

V1.2 24/06/2024

- Added test points.
- New divisor resistor to BNC connector, therefore we can use two differents vbias voltage for both connectors.

V1.3 29/06/2024

- Modified test points.
- corrected some typing errors.
- Added in page 12 more information about how works the circuit.
- Erased the LDO with Vout 1V8 by two voltage for bias the BNC connnector and with the two voltarge dividers is sufficient.
- Added right button button to change the power up button.
- Feedback circuit in OpAmp with estimated resistance and capacitor values on page 12.

V1.4 21/07/2024

- Finished routing.
- Added the mass plane for the top and bottom layer.
- Polygon manager used to set the differents polygons at good layer

V1.5 31/07/2024

- Increased track widths and track spacing, especially on the SPI bus.
- Generation of the gerber files to send at the manufacturer JCLPCB.

Index and changelog

Detailed index and changelog with number pages.
Revise the changelog for see the real model and footprint.

GND1 GND2 GND3 GND4
GND GND GND GND

Designer's signature

Supervisor's signature

Sheet title: Index and changelog.

Project title: TIK_HandheldDevice_Model2.PrjPcb

Designer: Juan Manuel Martín Lucena

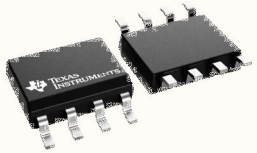
Date: 07/06/2024 | Revision: V1.5 | Sheet 2 of 12

Supervisor:
Sr. Andrés Roldán Aranda
Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain

Connection Mapping with Typical Current

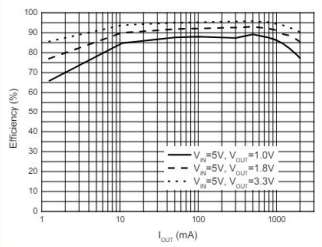
More detailed in consumption Appendix

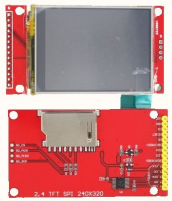
3.7 V 1800mAh 6.66 Wh



INA219
Rshunt = 0.2 Ohms
Ploss = 0.5 W

Buck AP3429KTTR-G1
Vout = 3.3 V 95% Efficiency at 1A





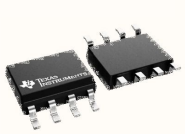
ILI9341 TFT LCD
3V3 variable consumption, depending os the brightness

Buzzer 3 V less 25 mA

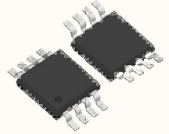
Thumb button 3V3 330 uA

ESP 32 WROOM
32D 3V3 typ 80 mA

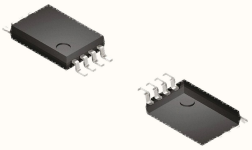
CH340 3V3 typ 12 mA



INA219 Current sense
3V3 40 mA



LMV358IDGKR
Dual OpAmp 615 uA



PFC85363ATT
Reat Time Clock
0.28 uA

Two voltage dividers that polarize the bnc connectors to a voltage of our choice

BNC female connector polarized at VBNC_EMITTER or VBNC_RECEIVER.

Connection mapping with typical current

Brief summary of voltages used and consumptions
more details in the Appendix.

Designer's signature

Supervisor's signature

Sheet title: Connection mapping and typycal current.

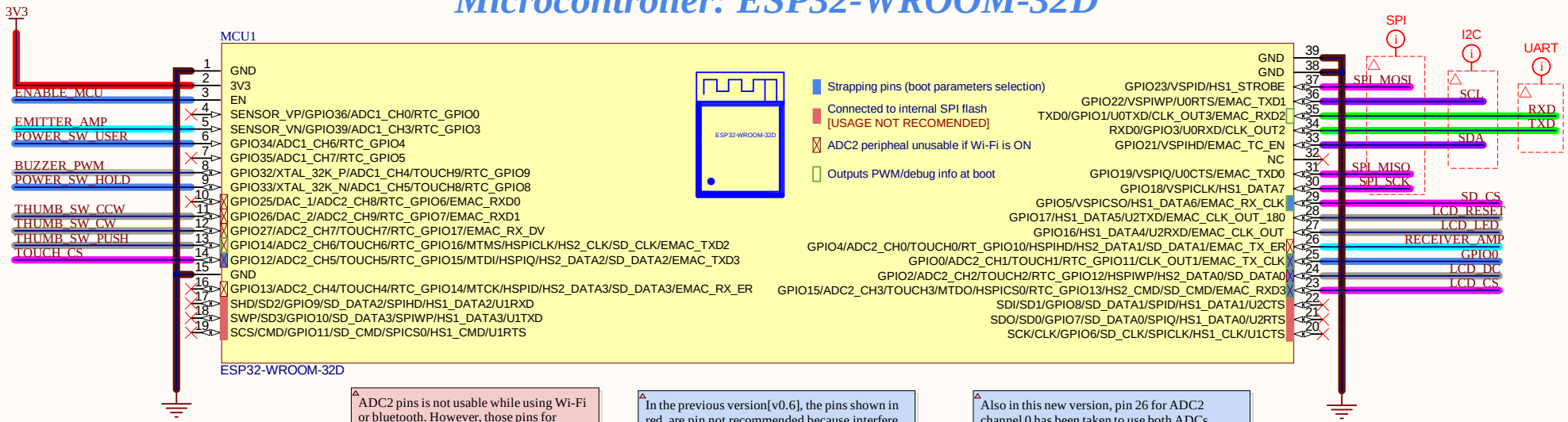
Project title: TIK_HandheldDevice_Model2.PrjPcb

Designer: Juan Manuel Martín Lucena

Date: 07/06/2024 | Revision: 1.5 | Sheet 3 of 12

Supervisor:
Sr. Andrés Roldán Aranda
Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain

Microcontroller: ESP32-WROOM-32D



ADC2 pins is not usable while using Wi-Fi or bluetooth. However, those pins for digital I/O functions work correctly.

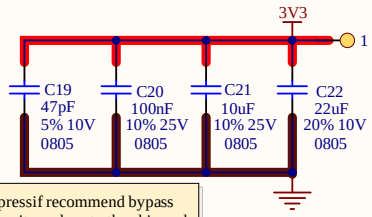
GPIO 34, 35, 36, 39 works only how inputs pins.

In the previous version[v0.6], the pins shown in red, are pin not recommended because interfere with SPI flash during progamming resulting in a programming error. And in this new version model 2, must be remain floating and not used.

Also in this new version, pin 26 for ADC2 channel 0 has been taken to use both ADCs.

And pin 5 (ADC1 channel 3) has been changed tc pin 4 (ADC1 channel 0) because the ADC output data packets contain more bits number of bits being the two most significant bits for the I2S configuration.

The strapping pins show how are by default these pins. Some in pull Up and other in pull down. The connection in the pins must have take care with this [Table 4 ESP32 WROOM 32 D short version].



Espressif recommend bypass capacitors close to the chip and short return paths for ensurance stability (page 21 ESP32-WROOM-32D Datasheet)

Two ceramic capacitors 0.1 uF and 10 uF for low ESR. Added one extra 22 uF electrolytic cap to filter current spikes during ESP32 RF usage and 47 pF capacitor to be more effective filtering high frequencies.

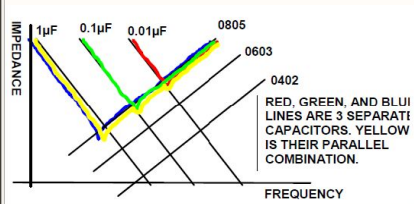
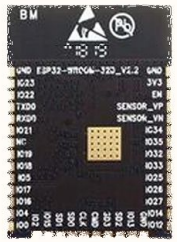
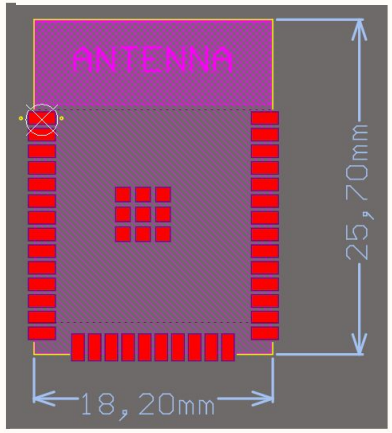


Table 4: Strapping Pins

Voltage of Internal LDO (VDD_SDIO)			
Pin	Default	3.3 V	1.8 V
MTDI	Pull-down	0	1
Booting Mode			
Pin	Default	SPI Boot	Download Boot
GPIO0	Pull-up	1	0
GPIO2	Pull-down	Don't-care	0
Enabling/Disabling Debugging Log Print over U0TXD During Booting			
Pin	Default	U0TXD Active	U0TXD Silent
MTDO	Pull-up	1	0
Timing of SDIO Slave			
Pin	Default	FE Sampling FE Output	FE Sampling RE Output	RE Sampling FE Output	RE Sampling RE Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1





ESP32-WROOM-32D MCU, Wi-Fi + Bluetooth module

This module integrates an ESP32 chip at 240 MHz dual core processor with Wi-Fi and bluetooth capabilities.
In this sheet describes the hardware configurations and Inputs/Outputs pins.

Designer's signature

Supervisor's signature

Sheet title: ESP32-WROOM-32D MCU

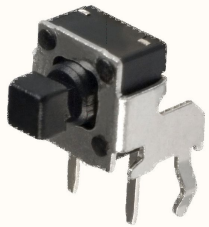
Project title: TIK_HandheldDevice_Model2.PrjPcb

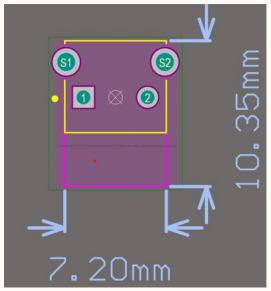
Designer: Juan Manuel Martín Lucena

Date: 07/06/2024 | Revision: V1.5 | Sheet 4 of 12

Supervisor:
Sr. Andrés Roldán Aranda
Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain

Power up button



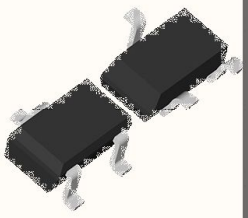


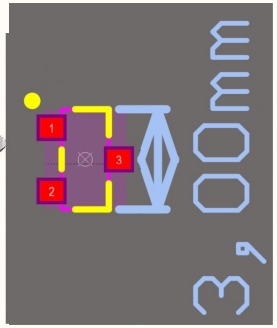
In the previous version (v0.6) this circuit did not work due to bad soldering and improper voltages on the Mosfet gate which caused the voltage to be fixed.

This was fixed in the previous version of PCB pending to check that it works in the next version 1.0.

In addition, the pin POWER_SW_USER was changed from GPIO26 to GPIO33.

High resistors for minimum power dissipation (consult Appendix if it value not is high).





This enable signal goes to the DC/DC buck converter, which starts working and powers up the system with 3,3 bus.

PW_USER detect groung voltage as falling edge therefore our ESP32 know that the curstomer has pressed the button and wants turn on or turn off the device.

This enable signal goes to the DC/DC buck converter, which starts working and powers up the system with 3.3 V.

Power up button

User activity control circuit with logic control monitoring via ESP32 of button activity preventing accidental shutdowns. Show appendix to see the simulations and power up time.

Designer's signature

Supervisor's signature

Sheet title: Power Up Button

Project title: TIK_HandheldDevice_Model2.PrjPcb

Designer: Juan Manuel Martín Lucena

Date: 07/06/2024 | Revision: V1.5 | Sheet 5 of 12

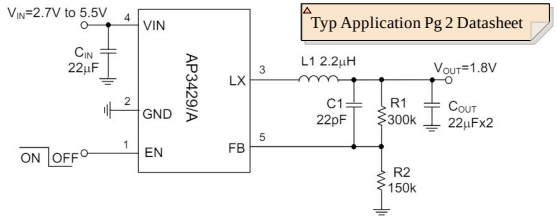
Supervisor:
Sr. Andrés Roldán Aranda
Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain

Main system power 3V3 and Vbias for signal conditioning circuit

Capacitors must be placed close to the chip and traced whith short loops.

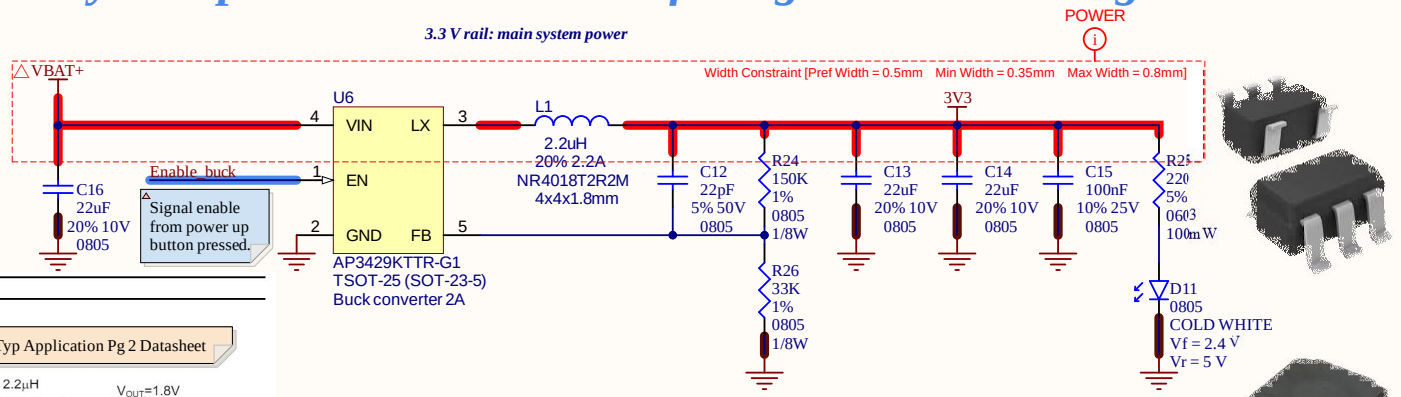
The FB voltage determinate the Vout. We need 0,6V in this pint, and for what the resistors value are chosen with a simply voltage divider at the output: V_FB = Vout*R5/(R4+R5) Finally with Vout = 3V3 and V_FB = 0.6V R5 = 0.22*R4

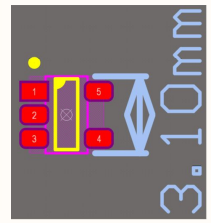
Typical Applications Circuit

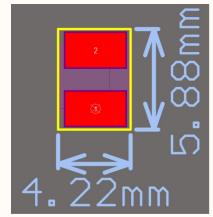


Typ Application Pg 2 Datasheet

3.3 V rail: main system power







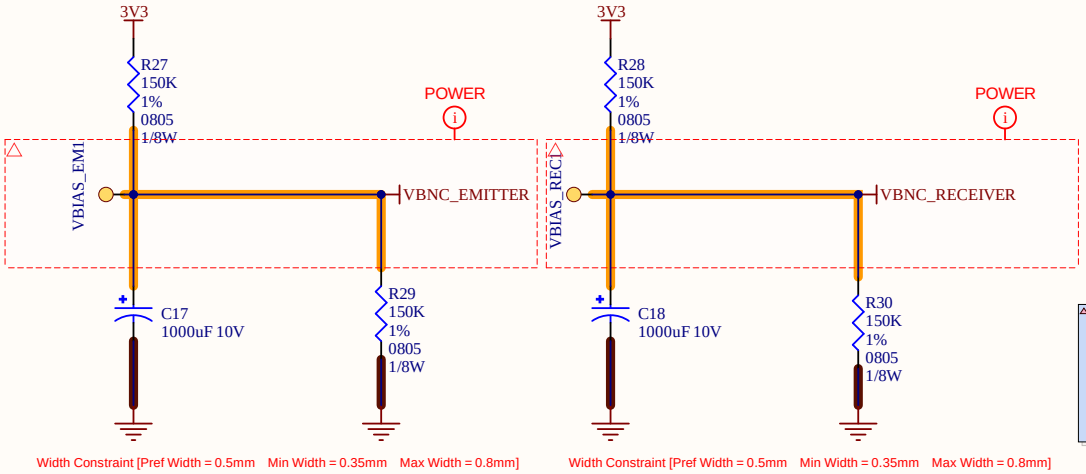
Resisitors value must be for 1.8 V in the output:

Vout = Vin*R29/(R29+R27)

Finally:
R29 = 1.5/1.8 * R27 = 0.84*R27

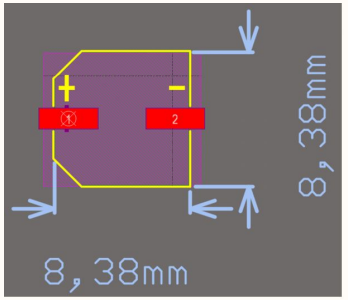
We can see the main problem of this method: commercially it is difficult to obtain resistors with the values given by the upper expression.

1V8 with voltage divider



Take care and not jump both power buses or the votage bias will be 3V3 and this produce errors in the conditioning circuit because the opAmp is powered with 3V3. .

Electrolityc capacitor to fix voltage.



Has been removed the LDO. We need two differents voltage at the BNC connector to bias them, the refore the LDO is not necessary and with these two voltage divider we make differents stable voltages.

Main system power 3V3 and Vbias for signal conditioning circuit

Battery DC voltage converts the voltage with DC/DC Bbuck converter from VBAT to 3.3V stable voltage and for other hand, we have two voltage divisor to bias both BNC connectors with different voltages.

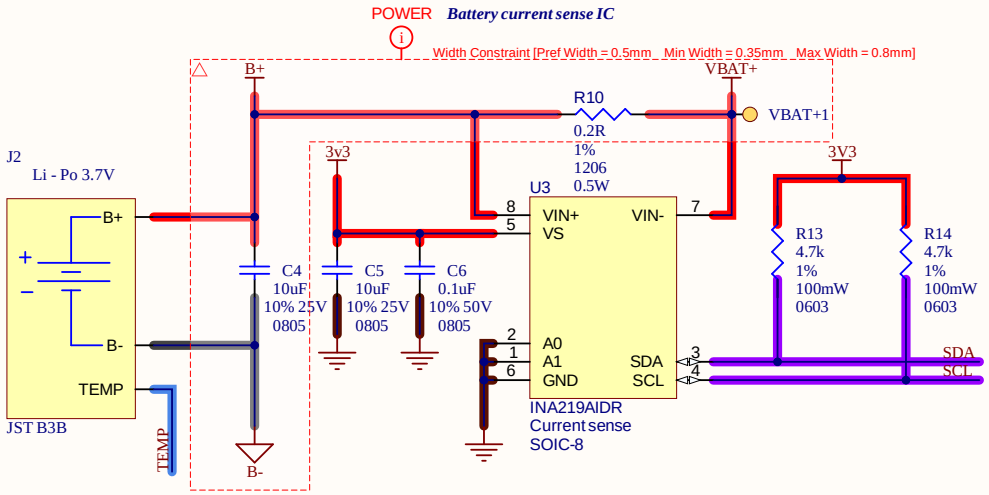
Designer's signature

Supervisor's signature

Sheet title: Main system power			
Project title: TIK_HandheldDevice_Model2.PrjPcb			
Designer: Juan Manuel Martín Lucena			
Date: 07/06/2024	Revision: V1.5	Sheet 6 of 12	

Supervisor:
Sr. Andrés Roldán Aranda
Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain

Lithium polymer battery: Current sense, charger and protection.



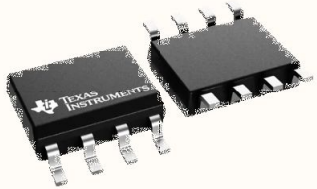
Temperature is a connection from the thermistor that is allocated inside the battery. The battery has three wires and the third is our temperature monitory connection to TP4056.

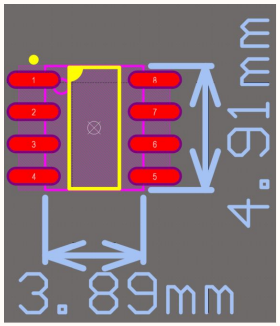
In this new version 1.0, the power supply of the INA219 IC has been modified. In the previous version it depended on the battery value and this can drop below the minimum 3.3V power supply of the INA and therefore, it is no longer possible to monitor the battery and the consumption of the TIK device.

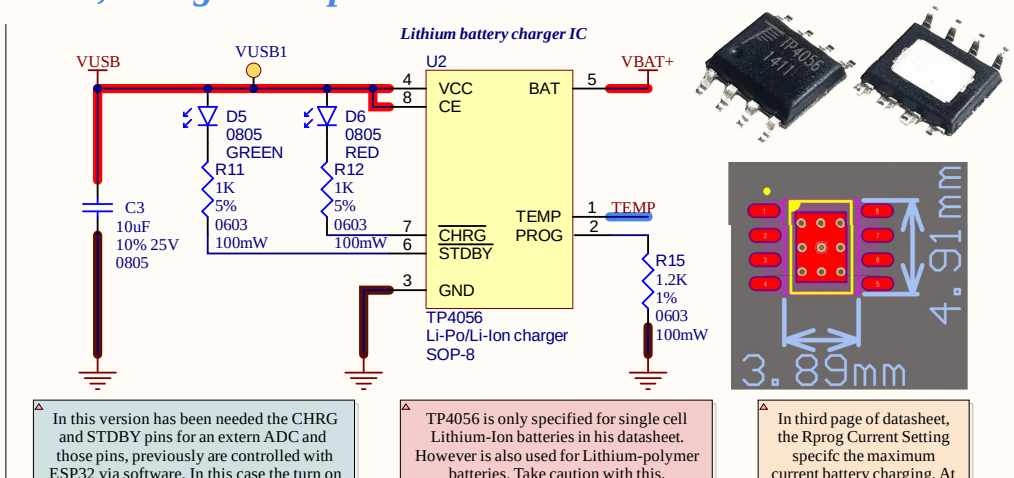
All the current that feeds the device circulates through the Rshunt, so with the INA we can know the power consumed during the time the equipment has been switched on and thus make a diagnosis of how much battery power is left. When the current through the resistor is positive, the battery is discharging, and when it is negative, the equipment is being charged.

This way we can monitor via software the battery level and works like a BMS (Battery Management System).

Placed power-supply bypass capacitors as close as possible to the device to ensure stability.







In this version has been needed the CHRG and STDBY pins for an extern ADC and those pins, previously are controlled with ESP32 via software. In this case the turn on and off is controlled by the TP4056.

TP4056 is only specified for single cell Lithium-Ion batteries in his datasheet. However is also used for Lithium-polymer batteries. Take caution with this.

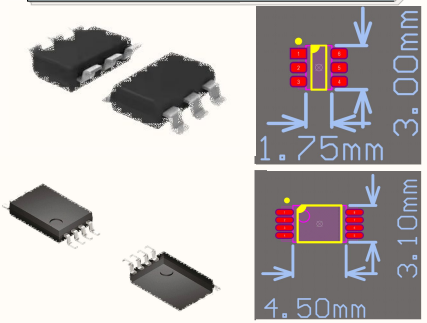
In third page of datasheet, the Rprog Current Setting specifc the maximum current battery charging. At 1.2 kOhm, the current is 1A

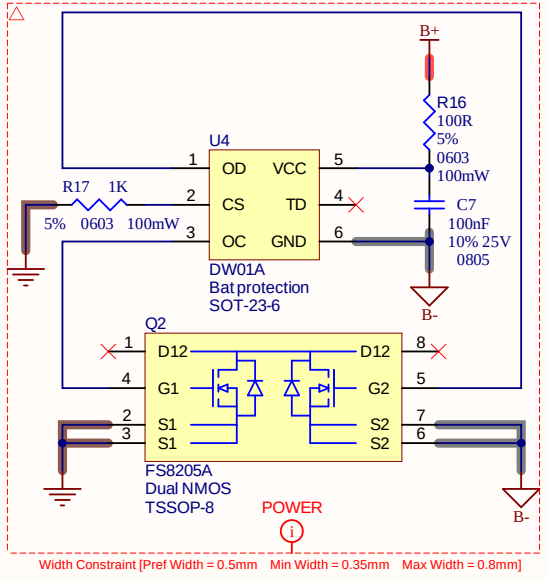
Lithium battery protection

This protection IC is designed for one-cell lithium-ion and lithium-polymer battery pack and has precision overcharge protection voltage (4.3V ± 50 mV) and overdischarge detection voltage (2.4 ± 100mV).

The protection method consists in isolate the battery pin BAT- and be disconnected from GND, like a switch pin.

In this version has been modified the VCC pin and connected at B+ of the battery rather Vmeas.





Lithium polymer battery: Current sense, charger and protection.

Battery monitoring and protection to overchalge and overdischarge and monitoring the lithium battery with a inside thermister in the battery through the TP4056 charger. Implemented battery and management system.

Designer's signature

Supervisor's signature

Sheet title: Lithium polymer battery

Project title: TIK_HandheldDevice_Model2.PrjPcb

Designer: Juan Manuel Martín Lucena

Date: 07/06/2024 | Revision: V1.5 | Sheet 7 of 12

Supervisor:
Sr. Andrés Roldán Aranda
Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain

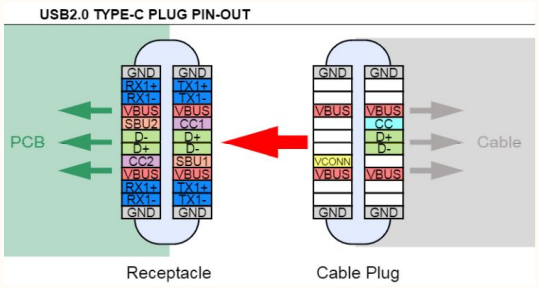
Type C USB connector and ESD protectioin

Sideband Use pins (SBU 1 and 2) are used to carry alternate modes that allow the Type-C connector to transport other high speed protocols for us this pins are not used.

CC pins detect the cable orientation and the resistors are pull-down ressitor to GND.

Pay attention in CC Resistors because in the lab 5.1k are not available and the datasheet recommend this value. Forums say that a similar resistor does the same job, take caution with this.

Ferrite Bead as EMI supressor. Works similar to an inductor: high impedance for high frequencies and very low for DC.



Pg 7/20 Microchip 2015 Datasheet

The USB 2.0 connector is the CGT type-C 4085 with 16 pins because less pins options dont have the data transfer and only can be used for power port (8 pins). More options pins like 24, is unnecessary because we dont use high speed data transfer.

In this new version has been modified the USB connector. In previous version the connector was an USB Mini-B for robutness and easy implementation.

The change is necessary to modernise the device.

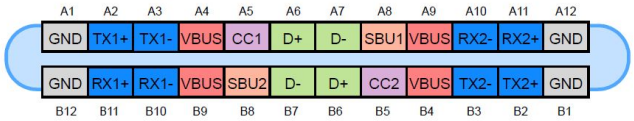
ESD protection IC should be as close as possible to the USB connector to avoid damage in other circuits.

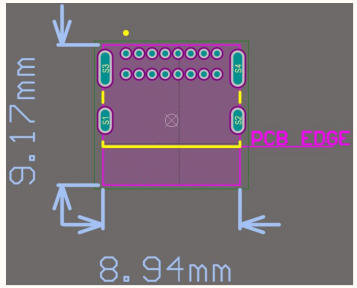
0R ressitors to testing process without ESD protection

Pg 3/20 Microchip 2015 Datasheet

In the component order, 5.1k resistors have been included, use these instead of 4.7k to ensure that there is no failure.

FIGURE 2: USB TYPE-C RECEPTACLE





USB connector and ESD protection circuit

USB-C will be mandatory in Europe by the end of 2024 and we will comply this law and the device is protected with a model of ESD protection.

Designer's signature

Supervisor's signature

Sheet title: Type C connector and ESD protection.

Project title: TIK_HandheldDevice_Model2.PrjPcb

Designer: Juan Manuel Martín Lucena

Date: 07/06/2024 | Revision: V1.5 | Sheet 8 of 12

Supervisor:
Sr. Andrés Roldán Aranda
Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain

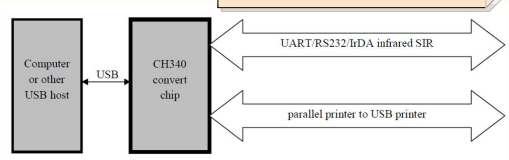
USB to UART, MCU and button programming

USB conver chip changing to serial interface. RS232 protocol for example can be used but in this case is not necessary.

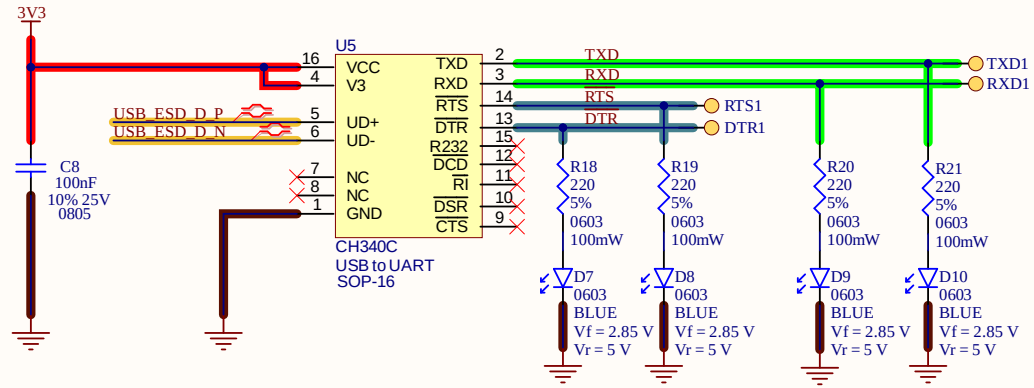
In this version doesn't needed an external oscillator.

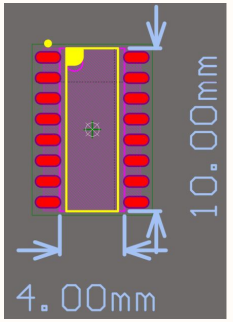
Pins RTS (Ready To send) and CTS (Clear To Send) works to controls the flow of data and therefore the timing.

CH340 feature 1/6 datasheet version 1D

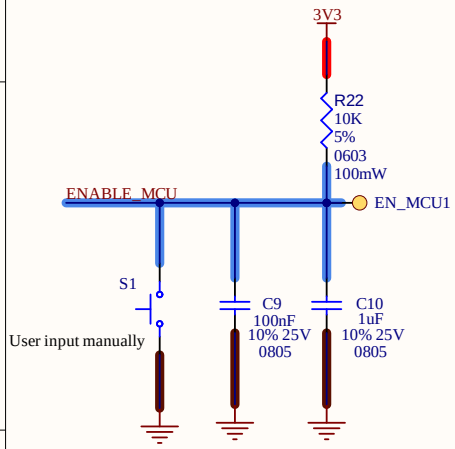


USB to UART conversion



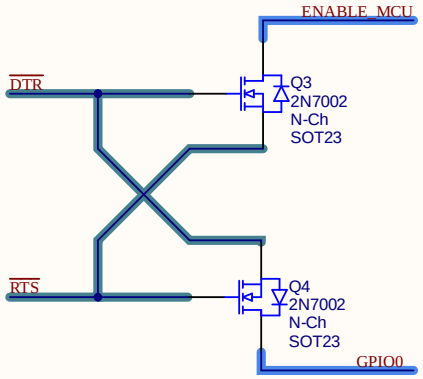


Reset button



If the ENABLE_MCU pin briefly goes from HIGH to LOW, the esp32 is reset.

Auto programming circuit



Truth table

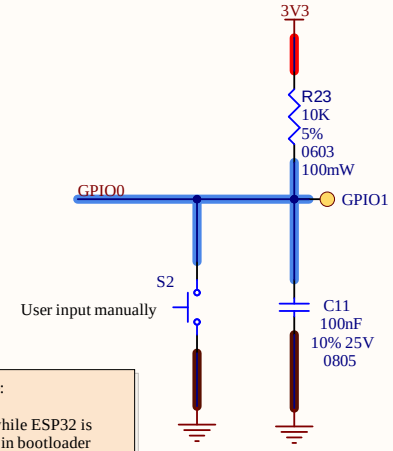
DTR	RTS	ENABLE_MCU	GPIO0
0	0	1	1
0	1	1	0
1	0	0	1
1	1	1	1

GPIO2 is a strapping pin, if GPIO0 is pulled up reset by default. Enable_mCU is pulled up by an external pullup resistor.

When GPIO is HIGH, it boots from internal SPI memory, but when is LOW changes to DOWNLOAD and upload a program.

In summary, the behaviour of the USB-Serial chip to put the ESP32 into bootloader mode to load a new software over USB.

Boot mode selection (debug)



Flashing mode:

- LOW level while ESP32 is powered entry in bootloader mode to upload a new firmware.

- HIGH level while ESP32 is powered execute the firmwared uploaded in the flash memory.

USB to UART and MCU programming

This circuit allows a computer to the ESP32 via USB so it can be reprogrammed. This is possible by sending RTS. connector, it needs to have a protection circuit against electro-static discharge (ESD) and noise.

Designer's signature

Supervisor's signature

Sheet title: USB to UART and programming MCU

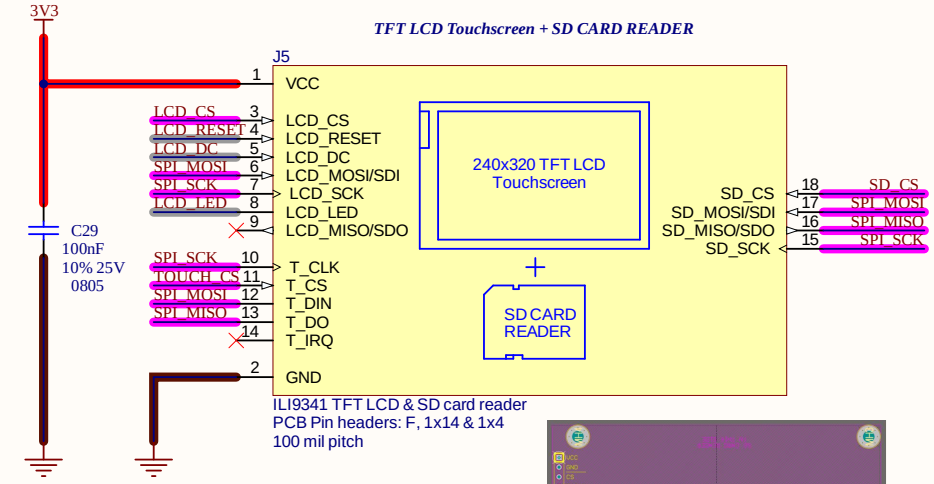
Project title: TIK_HandheldDevice_Model2.PrjPcb

Designer: Juan Manuel Martín Lucena

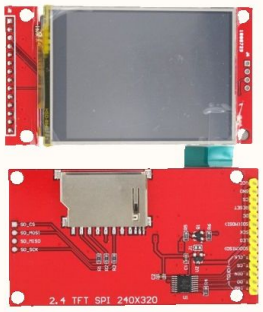
Date: 07/06/2024 | Revision: V1.5 | Sheet 9 of 12

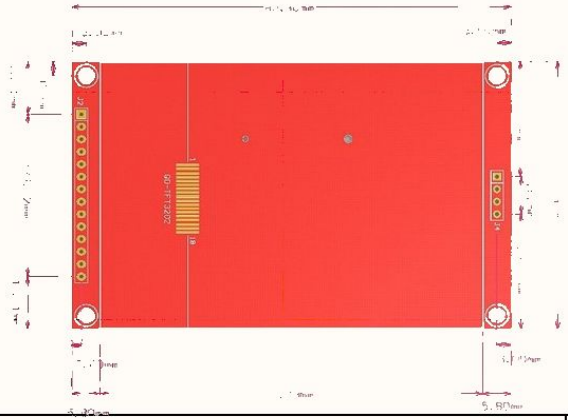
Supervisor:
Sr. Andrés Roldán Aranda
Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain

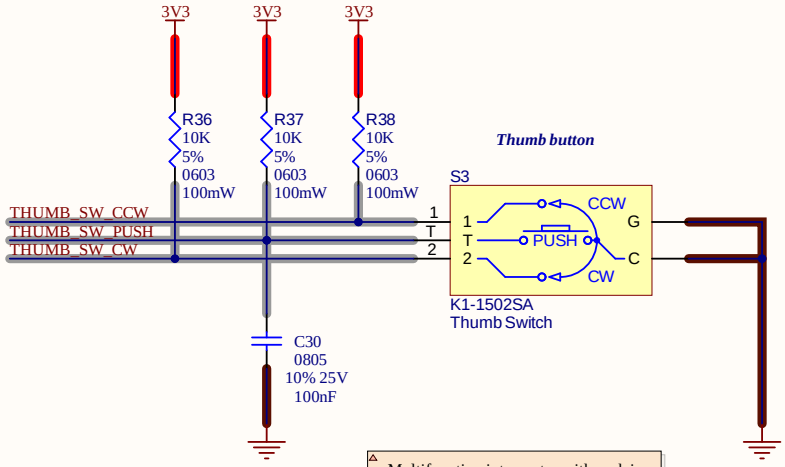
User interface: Screen, SD card reader and thumb button.



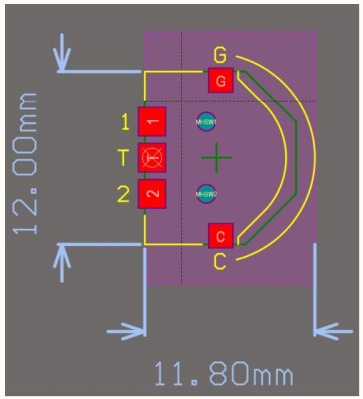
In this new version (model 2), the LCD_LED connection has been changed. Before the pin was connected to VCC and the brightness value could not be changed.







Multifunction interruptor with push in the centre. Thumb button is allocated at the righ side of the device.



Screen, SD card reader and thumb button

This device use a 2.8" TFT LCD display module with IC touch sensor and include a SD card reader to storage the data measures from tree inspection.

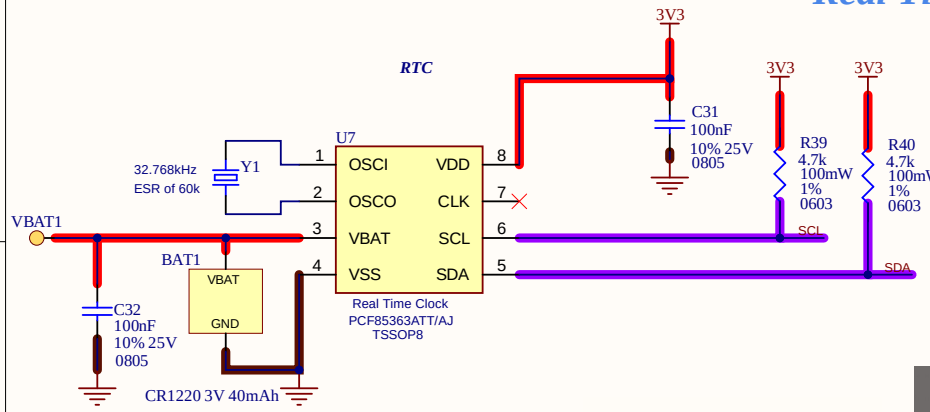
Designer's signature

Supervisor's signature

Sheet title: GUI: User interface components.			
Project title: TIK_HandheldDevice_Model2.PrjPcb			
Designer: Juan Manuel Martín Lucena			
Date: 07/06/2024	Revision: V1.5	Sheet 10 of 12	

Supervisor:
Sr. Andrés Roldán Aranda
Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain

Real Time Clock and Buzzer

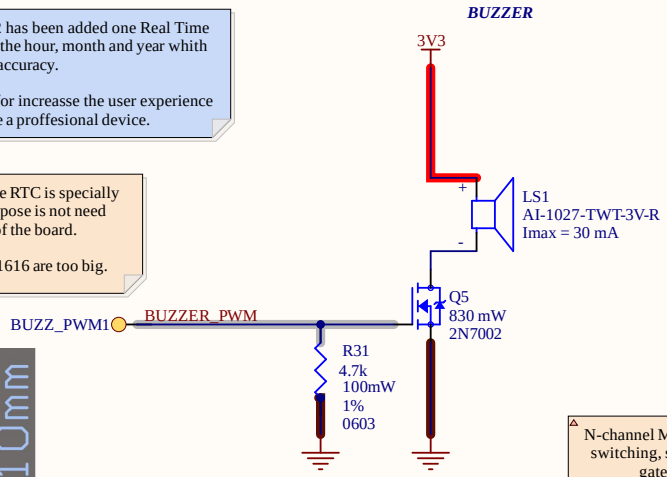


In this new version model 2 has been added one Real Time Clock to know in the device the hour, month and year whith high accuracy.

And the buzzer also added for increasse the user experience with the device like a proffesional device.

The battery chosen for the RTC is specially litttle and the main purpose is not need innecesary space of the board.

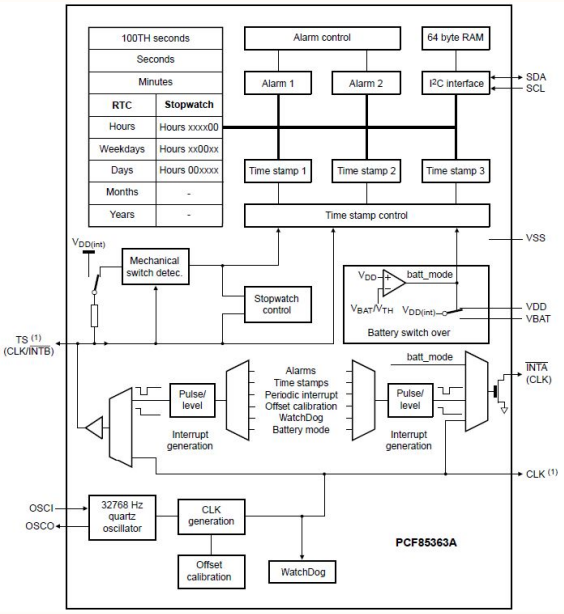
Other options like CR1616 are too big.

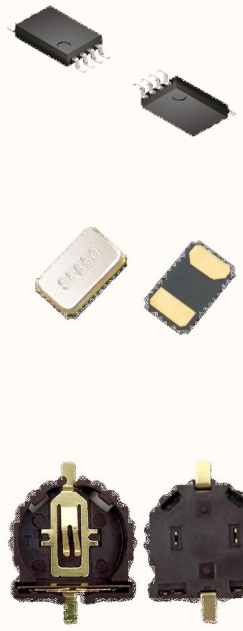


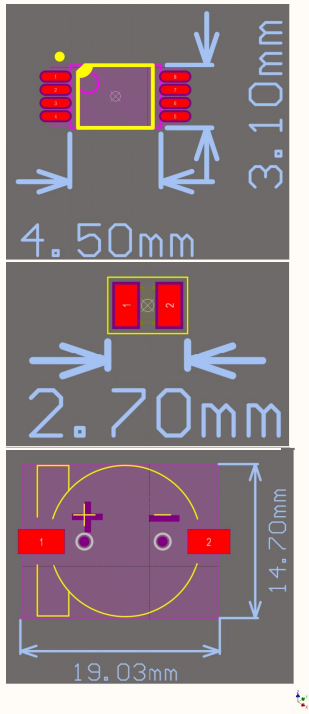
N-channel MOSFET with very fast switching, suitable for logic level gate drive sources.

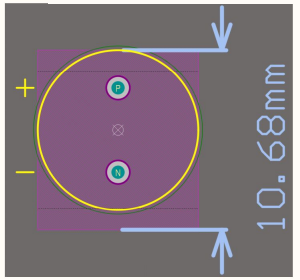
The Square wave signal from the MCU produce in the buzzer sound.

Block Diagram









Real time clock and buzzer

Was added a real time clock IC to show in the LCD scrreen the hours, minutes and seconds and also will be programmed alarms. In addition, a buzzer for sounds like a professional equipment.

Designer's signature

Supervisor's signature

Sheet title: Real time clock and buzzer

Project title: TIK_HandheldDevice_Model2.PrjPcb

Designer: Juan Manuel Martín Lucena

Date: 07/06/2024 | Revision: V1.5 | Sheet 11 of 12

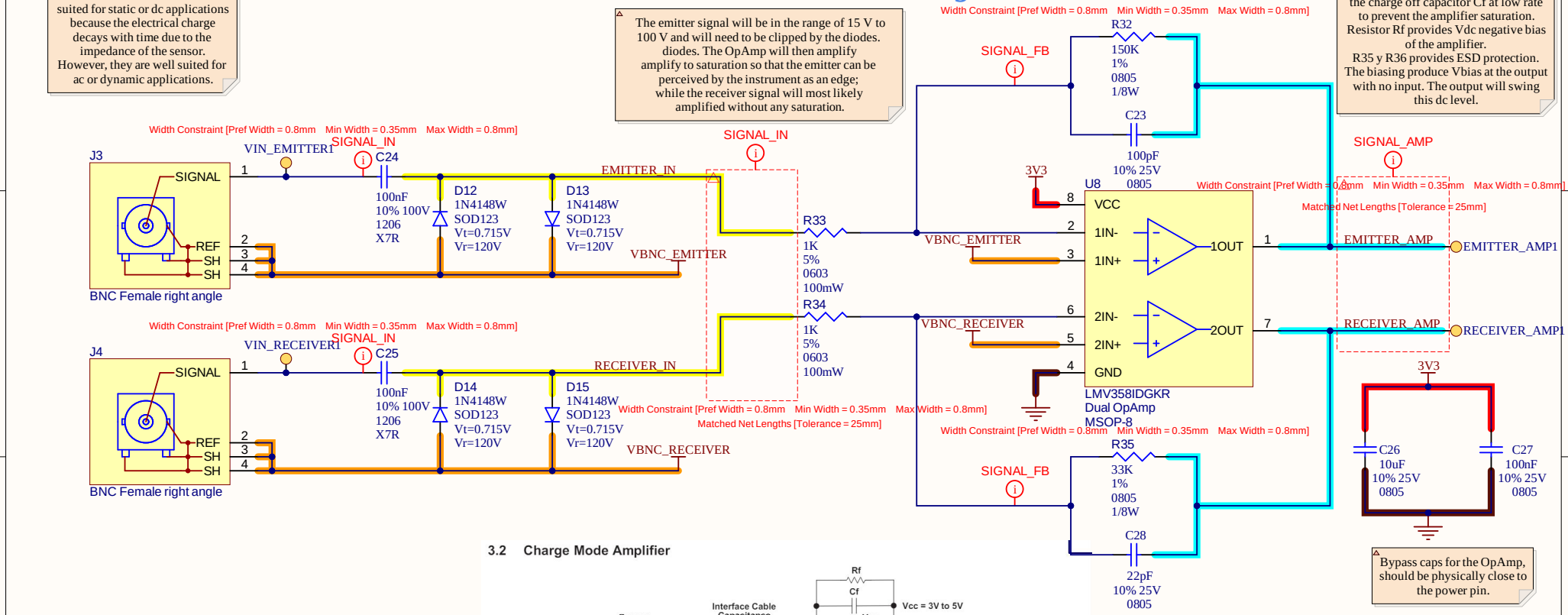
Supervisor:
Sr. Andrés Roldán Aranda
Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain

Piezoelectric sensors and conditioning circuit

Piezoelectric sensors are not suited for static or dc applications because the electrical charge decays with time due to the impedance of the sensor. However, they are well suited for ac or dynamic applications.

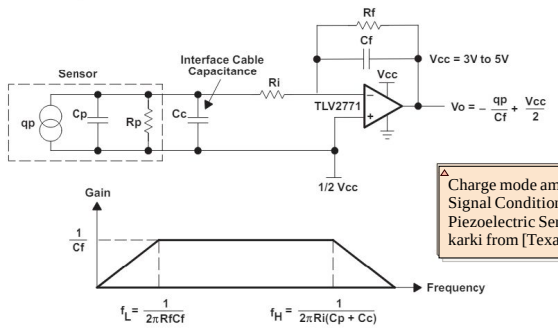
The emitter signal will be in the range of 15 V to 100 V and will need to be clipped by the diodes. diodes. The OpAmp will then amplify amplify to saturation so that the emitter can be perceived by the instrument as an edge; while the receiver signal will most likely amplified without any saturation.

As we can see in the figure 3, the Cf module the gain and ressitors Rf bleeds the charge off capacitor Cf at low rate to prevent the amplifier saturation. Resistor Rf provides Vdc negative bias of the amplifier. R35 y R36 provides ESD protection. The biasing produce Vbias at the output with no input. The output will swing this dc level.

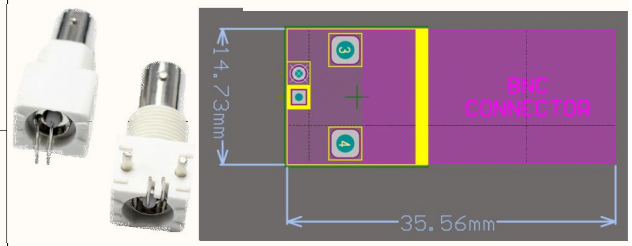


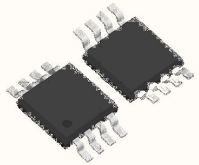
Bypass caps for the OpAmp, should be physically close to the power pin.

3.2 Charge Mode Amplifier



Charge mode amplifier page 4/6 Signal Conditioning Piezoelectric Sensors by James karki from [Texas Instruments]





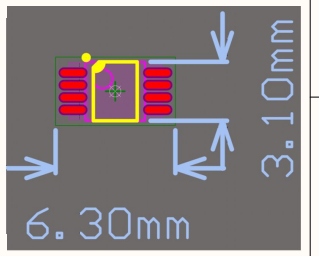


Figure 3. Charge Mode Amplifier Circuit

Piezoelectric sensors and conditioning circuit

There are two analog signals from the tree or trunk. The way piezos work force us to use this circuit to convert current into voltage.

Designer's signature	Sheet title: Piezoelectric sensors contioning circuit		Supervisor: Sr. Andrés Roldán Aranda *Dpto. Electrónica y Tecnología de Computadores* *University of Granada* *C/ Fuente Nueva, s/n, 18001* *Granada, Granada, Spain*
Supervisor's signature	Project title: TIK_HandheldDevice_Model2.PrjPcb		
	Designer: Juan Manuel Martín Lucena		
	Date: 07/06/2024	Revision: V1.5	Sheet 12 of 12

Apéndice B

Costes del proyecto

En este apéndice se detallará todo los gastos generados en la realización del proyecto. Estos incluyen: gasto material, salarios y coste de la energía.

B.1. Costes materiales

B.1.1. Componentes electrónicos

Muchos de los componentes incorporados en el TIK Modelo 2 vienen de la versión anterior y por tanto el coste de estos ya se realizó en su día y viene detallado en el apéndice A de [6]. En esta nueva versión los componentes y su coste vienen detallados en la tabla B.1.1.

Item	Breve Descripción	Cantidad	Coste Total (€)
SMFLP2-3.0	Bombillas tubo de LED 3in LGTH, 2mm core Flexpipe, 5.1mm lens	2	3.26
USB Type C	Conector Original USB4085-GF-A CONN RCPT USB2.0 tipo C 24POS RA	5	13.45
Pila de botón y de litio 3V	Batería de litio de 3V CR1220	10	5.39
Batería de LiPo	Batería recargable LiPo de polímero de litio, 3,7 V, 1800mAh, 803450 JST, 1,25mm, 3 pines	2	12.68
Connector JST B3B	Juego de terminales de paso XH2.54, 230mm/PH2.0, 2,54mm	1	2.94
LEDs SMD 0805 green y 0603 blue	Juego de diodos emisores de luz led, juego de diodos de luz de color 100, 0402, 0603, 0805 smd, 1206 Uds.	2	2.54
JST de ángulo recto	Conector macho JST de ángulo recto, soporte de aguja de pie de 90 grados, color blanco, PH 2,0mm	50	2.94
Right button angle	interruptor de Reinicio automático de 4 pines, ángulo recto con stent	20	1.84
Hembra BNCs	Conector hembra BNC de alta calidad, Conector de mampara con tuerca, montaje PCB, ángulo recto	10	3.99
Resistencia SMD 5.1kΩ	100mW Thick Film Resistors 75V ±100ppm/°C ±1 % 5.1kΩ 0603 Chip Resistor - Surface Mount ROHS	100	2.00
Espressif Systems ESP32-WROOM-32E-N8R2	LCC-38(18x25.5) WiFi Modules ROHS	5	19.65
PCF85363A	Reloj de tiempo real Low Power Real time clocks	20	5.08
AP3429AKTTR-G1	Reguladores de tensión de conmutación 1.0MHz 2A Step Down No Latch Off	10	8.33
RVT1000UF10V34RV0081	1000uF 10V SMD,D8xL10.5mm Aluminum Electrolytic Capacitors - SMD ROHS	100	8.63
		Total	92.62 €

B.1.2. Fabricación de PCB y stencil

Esta PCB es de 2 capas y 2 onzas de peso, con la certificación RoHS ordenada a fabricar a JCLPCB. El fabricante nos impide un mínimo de 5 placas por pedido. El coste total viene detallado en B.1.2. Los costes de envío han sido bajos ya que no ha sido inidicado envío de urgencia.

Item	Breve Descripción	Cantidad	Coste Total (€)
PCB	149.5x61.5mm, 2-layer, FR4, HASL-RoHS	5	7.87
Stencil PCB	Stainless steel, 380x280mm, top solder only	1	7.02
		Total	14.89 €

B.1.3. Impresión de carcasa

Para imprimir la carcasa, por cuestiones de tiempo no ha sido posible el servicio de impresión que tiene la UGR disponible en la facultad de ciencias, por lo que ha sido impresa con la impresora propia del alumno corriendo a su cuenta los gastos de material y electricidad.

Tabla B.1 – *Coste de impresión de la carcasa en 3D*

Item	Breve Descripción	Cantidad (g)	Precio/gramo	Coste Total (€)
Carcasa superior	Usando JAYO Filamento 1.75mm color Antracita	29	0.0139	0.41
Carcasa superior	Usando JAYO Filamento 1.75mm color Antracita	25	0.0139	0.35
Tapa de pila inferior	Usando JAYO Filamento 1.75mm color Antracita	1	0.0139	0.0139
			Total	0.77 €

B.1.4. Costes de Software

Los programas usados par desarrollar el proyecto son en su mayoría open source y/o licencias de estudiante que proporcionan las empresas desarrolladoras, en la tabla B.1.4.

Categoría	Software y versión	Tipo de lincencia	Coste
Sistema operativo	Microsoft Windows 11	Licencia propia	Ninguno
Programación y depuración	Jupyter Lab >3.4	Open source	Ninguno
Programación y depuración	Python >3.9	Open source	Ninguno
Programación y depuración	Visual Studio Code >1.8	Open source	Ninguno
Programación y depuración	Plataform IO >Core 6.1.15	Extensión gratis de Visual Studio Code	Ninguno
Documentación	Latex (pdflatex) v2024 en VS Code	Extensión gratis de Visual Studio Code	Ninguno
EDA, análisis electrónico y simulación	LTSpice v2019	Licencia propia	Ninguno
EDA, análisis electrónico y simulación	Altium Designer 21	Licencia propia	Ninguno
CAD, diseño mecánico	Solidworks 2021	Licencia propia	Ninguno
CAD, impresión 3D	Ultimaker Cura	Open source	Ninguno
Diseño de imágenes	Inkscape	Open source	Ninguno
		Total	Ninguno

B.2. Salarios

La media de salario de un ingeniero electrónico ronda los 31 000 euros [46] con una retención a la seguridd social en el régimen general del 6.35 % de su salario bruto, mientras que al empleador aporta aproximadamente un 30 % del salario bruto [47]. Sin embargo, este salario es el considerado para un ingeniero senior, para un ingeniero junior el rango va desde los 21 000 a los 28 000 mil/año brutos, considerando aproximadamente 25 000 euros [48].

En España, la media de horas trabajadas con una jornada completa son de 1752 horas por año a 17.90 euros/hora y para un ingeniero junior rondan las 1200 horas, que con una los 12.9 euros/hora. Lo cual nos da un total de 15 480 euros de salario bruto para un ingeniero junior.

Las horas gastadas de un ingeniero senior y del junior son distintas ya que el proyecto ha sido realizado por el segundo y supervisado por el primero. El supervisor de esta tesis ha gastado cerca de 50 horas en este proyecto lo que hace un total de 895 euros mientras que las horas del ingeniero junior vienen detalladas en la tabla B.2. Por otro lado tenemos el coste de tener en nómina un trabajador que paga el empleador. Este porcentaje es en base al salario bruto.

Tarea	Número de horas	Coste a 12.90 euros/hora
Documentación y formación	30	387.00
Reuniones	25	323.00
Instrumentacion	22	384.00
Simulaciones	10	290.00
Diseño de PCB	120	1548.00
Desarrollo de Firmware	65	838.00
Desarrollo de Firmware	85	1096.00
Diseño de carcasa	55	709.00
Escritura de la memoria	217	2799.00
	Horas totales	629
	Salario bruto	8374.00
	Salario neto	7842.251

Concepto	Tipo de contribución	Porcentaje sobre el salario
Contingencias comunes	Riesgos comunes por enfermedades y accidentes no laboralores	23.6 %
Desempleo	Para contrato indefinido	5.50 %
Fondo de Garantía Salarial	Cubre indemnizaciones y salarios en caso de insolvencia del empleador	0.20 %
Formación Profesional	Financia programas de formación del trabajador	0.60 %
Contingencias profesionales	Cubre accidentes laborales y enfermedades profesionales	3.2 %
	Total	33.1 %
	Coste	2771.8

B.3. Costes eléctricos

En cuanto al apartado eléctrico también debemos de tener en cuenta los vatios consumidos por todos los equipos eléctricos: Osciloscopios, Ordenador, estación de soldadura y aire acondicionado. En la tabla ?? vemos el total. Para ello además debemos de contar con el precio del kilovatio por hora. Desconozco la tarifa contratada por los laboratorio de GranaSAT, supongo que al final el precio es pagado por la UGR y se puede acoger al modelo proprio, a la tarifa regulada conocida como PVPC 2.0TD o Tarifa 3.0 TD. A la tarifa regulada PVPC 2.0 TD dudo que se acoja la UGR ya que la potencia contratada no puede superar los 10 KW asi que, será la versión 3.0 TD.

Esta modalidad 3.0 TD para potencias contratadas de más de 10 KW discrmina hasta 6 periodos, debe de contar con maxímetro y disponer de contador de energía reactiva para su penalización. La factura eléctrica consta de un término de potencia contratada, energía que finalmente es consumida, impuesto de electricidad e IVA que últimamente

han variado mucho para mitigar los precios y, el alquiler de equipos de medida y control como los contadores (aunque también puede ser comprado). En la figura B.1 podemos ver la evolución del precio y haciendo una media, nos quedamos en los 90 €/MWh. Finalmente podemos ver el coste en B.2.

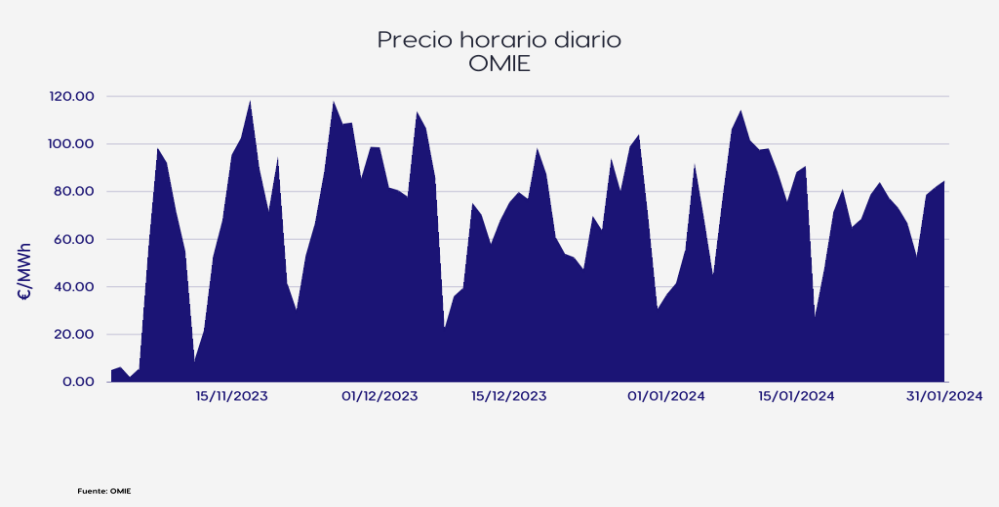


Figura B.1 – *Informe mercado energético.* [13]

Tabla B.2 – *Coste eléctrico.*

Equipo	Horas de uso	Consumo medio de potencia (W)	Energía consumida (MWh)	Coste
Portátil	607	30	0.0182	1.638
Instrumentación electrónica	25	80	0.002	0.18
Estación de soldadura	500	5	0.001	0.09
Aire acondicionado	900	70	0.063	5.67
			Coste total	7.58

Tabla B.3 – *Coste total del proyecto con beneficio industrial.*

Tipo	Concepto	Coste (euros)
Materiales	Componentes electrónicos, PCB y stencil, impresión 3D y envío	162.28
Recursos humanos	Salarios de ingeniero senior del supervisor y del ingeniero	12 288.27
Energía	Coste eléctrico	7.58
	Beneficio industrial (17 %)	2 117.88
	COSTE TOTAL CON BENEFICIO	14 576.01

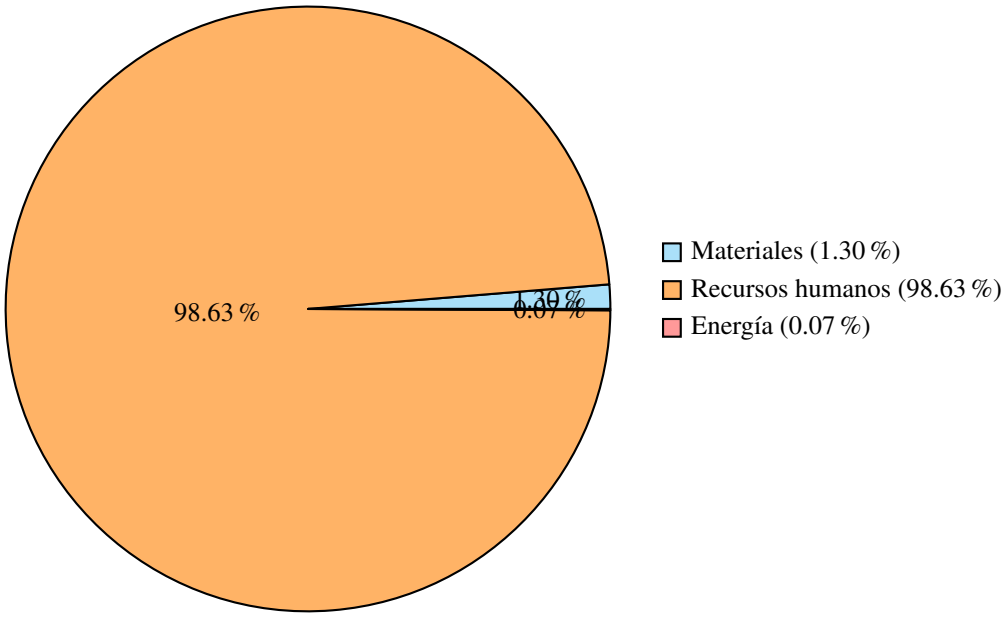


Figura B.2 – *Distribución de los costes del proyecto.*

Apéndice C

Cálculo de la Transformada de Fourier rápida en microcontrolador

C.1. Cálculo de la FFT por el microccontrolador

Para comprobar que el microcontrolador responde adecuadamente ante el procesado de señales, se ha implementado el algoritmo de la transformada rápida de Fourier y así comprobar que la frecuencia de resonancia calculada es correcta comparada con otros resultados y cálculos con el equipo TIK Timber desarrollado por Irene Gil Martín. Para ello, Irene me proporcionó una serie de señales de medidas realizadas con TIK TIMBER equipment a una troza de pino y otra de chopo. He usado el trabajo de Robin Scheimbler [49] para implementar el mencionado algoritmo ajustándolo a nuestras necesidades.

C.1.1. Serier de fourier para señales periódicas

El anólisis de Fourier nos permite analizar una señal aperiódica y obtener su espectro en frecuencia C.1. Si la señal fuese periódica, esta se definiría mediante la serie de fourier donde el espectro de frecuencias contaría con dos frecuencias sucesivas armónicamente relacionadas $\frac{1}{T_p}$ con T_p de periodo fundamental. El rango de frecuencias para señales discrets se limita a (-π,π) o (0, 2π). Estas frecuencias tienen un periodo fundamental N y estás separadas $2\pi/N$ radianes o $f = 1/N$ ciclos. Una señal periódica contendrá como máximo N componentes en frecuencia.

$$f(t) = a_0 + \sum_{n=1}^{\infty} \left[a_n \cos\left(\frac{2\pi nt}{T}\right) + b_n \sin\left(\frac{2\pi nt}{T}\right) \right] \tag{C.1.1}$$

donde los coeficientes a_n y b_n se calculan como:

$$a_0 = \frac{2}{T} \int_0^T f(t)\, dt \tag{C.1.2}$$

$$a_n = \frac{2}{T} \int_0^T f(t) \cos\left(\frac{2\pi nt}{T}\right) dt \tag{C.1.3}$$

$$b_n = \frac{2}{T} \int_0^T f(t) \sin\left(\frac{2\pi nt}{T}\right) dt \tag{C.1.4}$$

- Función periódica:

$$f(t) = a_0 + \sum_{n=1}^{\infty} \left[a_n \cos\left(\frac{2\pi nt}{T}\right) + b_n \sin\left(\frac{2\pi nt}{T}\right) \right]$$

- Término constante: a_0, que representa el promedio de la función $f(t)$ durante un período:

$$a_0 = \frac{2}{T} \int_0^T f(t)\, dt$$

- Coeficiente del coseno: a_n, que determina la amplitud de la componente cosenoidal en la serie:

$$a_n = \frac{2}{T} \int_0^T f(t) \cos\left(\frac{2\pi nt}{T}\right) dt$$

- Coeficiente del seno: b_n, que determina la amplitud de la componente senoidal en la serie:

$$b_n = \frac{2}{T} \int_0^T f(t) \sin\left(\frac{2\pi nt}{T}\right) dt$$

- Número de términos: n, que representa las diferentes frecuencias armónicas presentes en la señal.

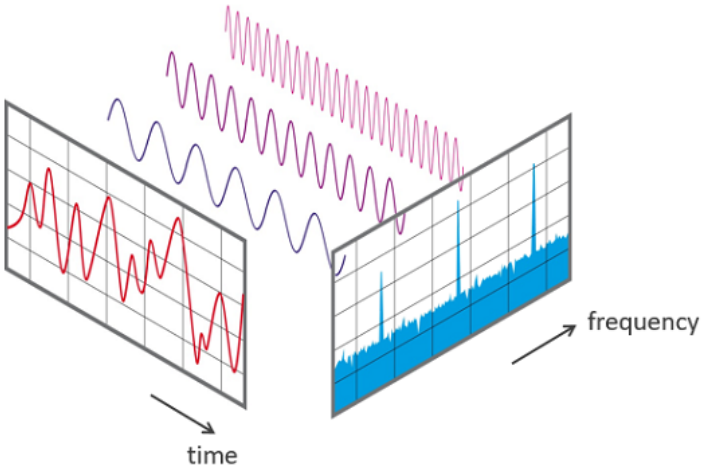


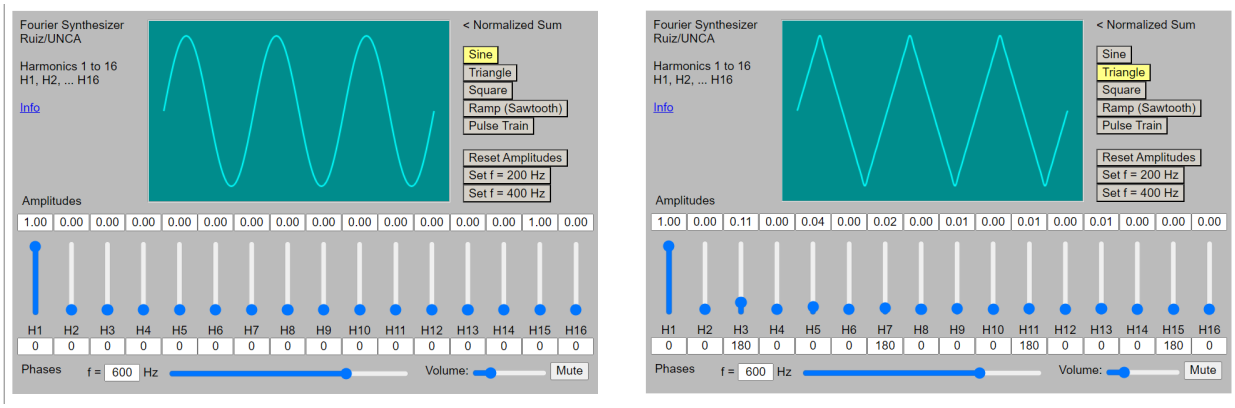
Figura C.1 – *Transformada de Fourier de una señal en sus armónicos. [14]*

C.2. FFT

Sea x_n una señal aperiódica discreta en el tiempo como son los valores de tensión recibidos al hacer el END sobre una troza:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-j\frac{2\pi}{N}kn}, \quad k = 0, \ldots, N-1 \tag{C.2.1}$$

Tenemos como resultado X_k como componenete en frecuencia k. Este número tiene su respectiva magnitud y fase en la frecuencia k.



(a) *Señal sinuidal periódica con solo un armónico.*

(b) *Señal triangular periódica con dos armónicos.*

Figura C.2 – *Sintetizador de fourier. Extraido de [15]*

- X_k: Es el resultado de la transformada rápida y representa la componente de frecuencia k en el dominio de la frecuencia. Este es un número complejo que contiene la magnitud y fase de la frecuencia k.
- $\sum_{n=0}^{N-1}$: Es la sumatoria que va desde $n = 0$ hasta $n = N - 1$, lo que significa que estamos sumando todos los valores de la señal x_n.
- x_n: Es el valor de la señal de entrada en el dominio del tiempo en el instante n.
- $e^{-j\frac{2\pi}{N}kn}$: Es el término exponencial complejo que se utiliza para convertir la señal en el dominio del tiempo al dominio de la frecuencia. Aquí, j es la unidad imaginaria, $\frac{2\pi}{N}$ es el factor de escala de la frecuencia, y k y n son los índices de la frecuencia y el tiempo, respectivamente.
- $k = 0, \ldots, N - 1$: Indica que la transformada se calcula para k desde 0 hasta $N - 1$, es decir, para todas las posibles componentes de frecuencia en la señal.

Conseguimos convertir una secuencia de valores en el dominio del tiempo en una secuencia de valores en el dominio de la frecuencia, ye sto exáctamente es lo que hace el algoritmo. Primero definimos unos parámetros de nuestros datos, como son la frecuencia de sampleo y número de datos. Posteriormente definimos el buffer donde almacenaremos los datos a usar junto a la definición e introducción de los mismos en el buffer mediante un bucle que se repite el número de datos. Después, Ejecutamos el algoritmo con fft_execute(buffer con datos de tensión) que en esencia realiza la sumatoria mostrada en la eccuación C.2.1. Imprimimos el resultado en el formato deseado para representarlo en python.

```
1
2     #include <Arduino.h>
3     #include <FFT.h> // Entre comillas no coge el .h !!
4     #include <FFT_signal.h>
5     #include <SPI.h>
6     #include <FS.h>
7     #include <SPIFFS.h>
8     #include <iostream>
9     #include <cstring>
10
11    char ram_memory_2[50]; // Buffer with Ram memory remin
12    char ram_memory_1[50];
13
14    void setup() {
15      Serial.begin(115200); // use the serial port
16      char print_buf[300]; // Espacio en el buffer a imprimir
```

```
17      fft_config_t *real_fft_plan = fft_init(FFT_N, FFT_REAL, FFT_FORWARD,
    fft_input, fft_output); // Buffer con almacenamiento de los datos
18
19      for (int k = 0 ; k < FFT_N ; k++) // k indica la longitud del archivo,
     en V0 tomamos el de un acelerómetro
20        real_fft_plan->input[k] = (float)fft_signal_1[k]; // Almacenamos los
     datos en el buffer
21
22      long int t1 = micros(); // Inicio de DTFT
23
24      sprintf(ram_memory_1, "Remaining RAM memory before FFT: %llukB\n",
    xPortGetFreeHeapSize() / 1024);
25
26      // Execute transformation
27      fft_execute(real_fft_plan);
28
29      // Print the output
30      for (int k = 1 ; k < real_fft_plan->size / 2 ; k++)
31      {
32        /*The real part of a magnitude at a frequency is followed by the
    corresponding imaginary part in the output*/
33        float mag = sqrt(pow(real_fft_plan->output[2*k],2) + pow(
    real_fft_plan->output[2*k+1],2))/1;
34        float freq = k*1.0/TOTAL_TIME;
35        //sprintf(print_buf,"%f Hz : %f", freq, mag); // From Robin
36        sprintf(print_buf,"[%f , %f],", freq, mag); // Manipulo para
    conseguir los datos como yo quiero
37
38        Serial.println(print_buf);
39
40        if(mag > max_magnitude)
41        {
42            max_magnitude = mag;
43            fundamental_freq = freq;
44        }
45      }
46      long int t2 = micros();
47
48      Serial.print(ram_memory_1);
49
50      sprintf(ram_memory_2, "Remaining RAM memory after FFT: %llukB\n",
    xPortGetFreeHeapSize());
51      Serial.print(ram_memory_2); // Remaining RAM memory after do the FFT
52
53      Serial.println();
54      /*Multiply the magnitude of the DC component with (1/FFT_N) to obtain
    the DC component*/
55      sprintf(print_buf,"DC component : %f g\n", (real_fft_plan->output[0])
    /10000/FFT_N);  // DC is at [0]
56      Serial.println(print_buf);
57
58      /*Multiply the magnitude at all other frequencies with (2/FFT_N) to
    obtain the amplitude at that frequency*/
59      sprintf(print_buf,"Fundamental Freq : %f Hz\t Mag: %f g\n",
    fundamental_freq, (max_magnitude/10000)*2/FFT_N);
60      Serial.println(print_buf);
61
```

```
62        Serial.print("Time taken: ");
63        Serial.print((t2-t1)*1.0/1000);
64        Serial.println(" milliseconds!");
65
66        // Clean up at the end to free the memory allocated
67        fft_destroy(real_fft_plan);
68
69    }
70
71    void loop() {
72
73    }
```

Listado C.1 – *Código en C++*

```
1
2    #
-------------------------------------------------------------------------------
3    #                    CONVERSIÓN DE DATOS
4    #
-------------------------------------------------------------------------------
5    # _____________________________ CHOPO ____________________________________
6
7    chopo_measures = np.array(chopo_measures)
8
9    freq_chopo = chopo_measures[:,0];
10   mag_chopo = chopo_measures[:,1];
11   mag_chopo = mag_chopo/len(mag_chopo) # Dividimos entre la longitud del
vector de magnitud
12
13   fig, ax = plt.subplots(figsize=(30, 6))
14   ax.plot(freq_chopo, mag_chopo, color='blue', linestyle='-', marker='o',
linewidth=1, label='Datos')
15
16   ax.set_xlabel('Frequency (KHz)', fontsize=14)
17   ax.set_ylabel('Magnitud', fontsize=14)
18   ax.set_title('FFT Chopo Results Summary 1', fontsize=16)
19
20   ax.grid()
21   plt.show()
```

Listado C.2 – *Código en Python*

Como podemos ver en la imagen C.4, tenemos la frecuencia fundamental a 2 kHz aproximándonos a los 1.7 kHz calculados con el TIK Timber de la compañera Irene, esto se puede deber a diferencias en el algoritmo usado por ella y el mio.

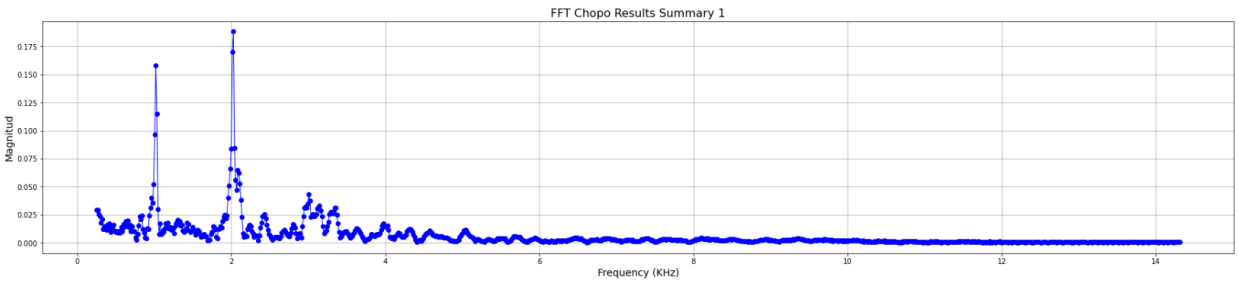


Figura C.3 – *Representación de la transformada discreta de Fourier calculada con el microcontrolador*

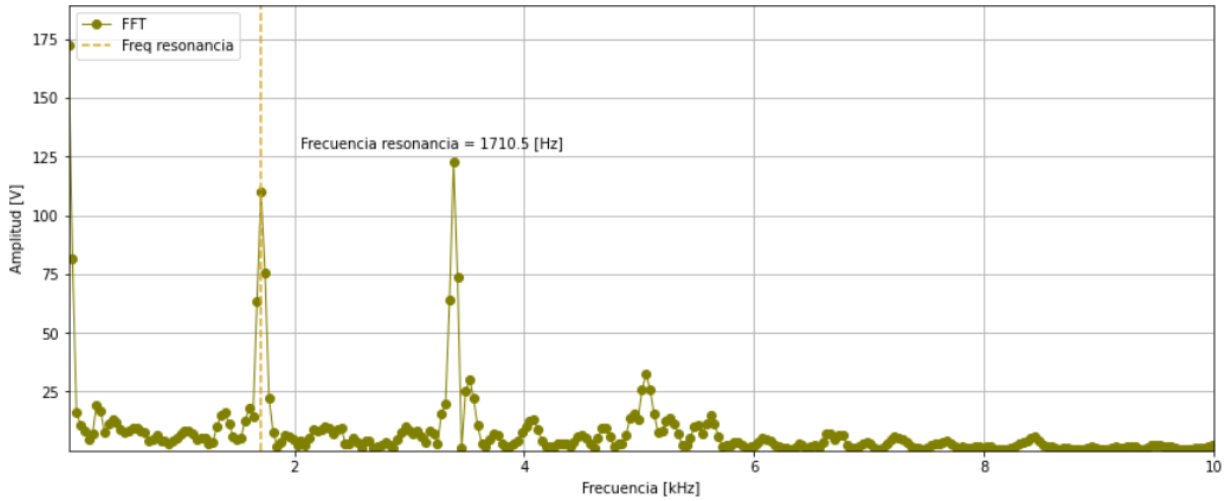


Figura C.4 – *Transformada de Fourier realizada con el equipo TIK Timber de Irene Gil Martín.*

Apéndice D

Documentación de código

D.1. Documentación con Doxygen

La herramienta utilizada para documentar software es Doxygen. Compatible con C/C++ entre otros. La generación es automática ya que se reconocen una serie de comentarios especiales en el código y estos deben de estar ubicados en ellugar correcto. Con ello podemos describir funciones,clases, métodos, y otros elementos del código. Doxygen puede generar la documentación en distintos formatos permitiendo la adaptación en varios formatos como HTML, LaTeX, RTF, PDF, Man Pages o XML. Además con la versión de doxygen 1.11.0 podemos generar diagramas de dependencia y jerarquias de clase con Graphviz.

En la página web de [50] tenemos todo lo necesario para realizar la instalación y como realizar la dcumentación del código que es lo que nos interesa.

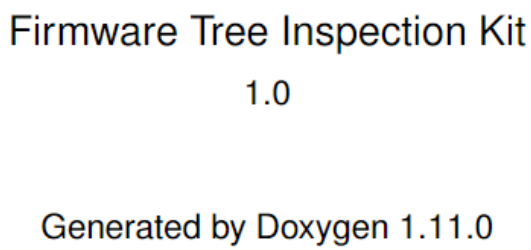
Figura D.1 – *Logo de Doxygen.*

La configuración la realizamos con un archivo que incluimos en el fichero del proyecto llamado Doxyfile donde especificamos las opciones de como documentar el código, que archivos documentar...etc. Una vez que hemos añadido las cabeceras de carácteres especiales en el código fuente, lanzando el archivo doxyfile desde la terminal logramos el archivo documentado en el formato de salida escogido (ver figura D.2a)

Para el texto que aparece en PDF debemos de elegir una u otra cabecera en el código. Al principio del fichero .h o .cpp hacemos una descripción de lo que se va a documentar como en mi caso:

- @mainpage: Lo ubicamos en el main.cpp y ponemos título al documento a generar.
- @file: Especificamos que archivo es el presente. Por ejemplo, @file gui.cpp.
- @section: Especificamos una sección del documento, como por ejemplo una introduccion donde damos una breve descripción del proyecto o una sección de registro donde indiquemos los avances y fechas de los mismos.

(a) *Ejecutación de la documentación mediante el archivo doxyfile*



(b) *Portada simple que genera doxygen 1.11.0*

- @include: Donde añadimos que librerías han sido usadas e incluidas en el presente fichero.
- @verbatim: Con cada @b incluimos las funcines declaradas en el void setup del fichero main.cpp

Para las cabeceras de cada función, clase o define he usado lo siguiente:

- @brief: Una breve descripción sobre lo que se va a documentar
- @note: Anotaciones concretas de la función, clase o define.

D.1.1. Generación de gráficos con Graphviz

Graphviz es una herramienta que integra DOxygen 1.11.0 que puede generar diagramas y gráficos avanzados. Este software de visualización es de código abiero y permite representar de forma estructural con diagramas el código programado. Para la utilización de esta herramienta necesitamos modificar del archivo doxyfile integrado en nuestro proyecto algunos parámetros:

- HAVE_DOT = YES
  - Activa el uso de Graphviz.
- DOT_PATH = /ruta/a/graphviz
  - Especifica la ruta donde se encuentra la herramienta ‘dot‘. En mi caso DOT_PATH = "$C:/ProgramFiles/Graphviz/bin$".
- CALL_GRAPH = YES
  - Genera gráficos de llamadas para cada función.
- CALLER_GRAPH = YES
  - Genera gráficos de los llamadores para cada función.

- CLASS_GRAPH = YES
  - Genera gráficos de herencia para las clases.
- INCLUDE_GRAPH = YES
  - Genera gráficos de dependencias de inclusión para cada archivo.
- DIRECTORY_GRAPH = YES
  - Genera gráficos de dependencias de directorios.

Para la correcta visualización de los diagramas necesitamos ajustar la profundidad de las imagenes que además es directamente proporcional con la complejidad de los diagramas generados. Este valor va de 0 a 1024, un valor intermedio puede ser una buena opción. Por defecto este valor MAX_DOT_GRAPH_DEPTH estará a 0 y debe de ser modificado.

Firmware Tree Inspection Kit

An advanced tool for monitoring and inspecting tree health

Version 1.3

Developed by Juan Manuel Martín Lucena

Contact: juanmanuelmartinlucena@gmail.com

September 4, 2024

Generated by Doxygen 1.11.0

Chapter 1

Program to calibrate the INA219 according to the Rshunt used and programmed as a BMS (Battery Management System) to monitor the battery of the TIK equipment.

Libraries.

Libraries

1.1 Dependencies

Arduino.h> // Library for managing Arduino projects

1.2 License

BSD license; all text here must be included in any redistribution.

1.3 Implementation Details

Read from INA219 values of current consumption and voltage bus and this miltiply value and compare with the max value power capacity battery in Wh

/**

Includes the necessary libraries for the program to work.

Chapter 2

Topic Index

2.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

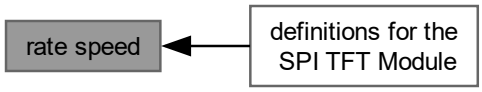
Chapter 6

Topic Documentation

6.1 rate speed

Definition of baud rate speed

Collaboration diagram for rate speed:



Topics

- **definitions for the SPI TFT Module**
  Pins connected to the device for use the SPI protocol. Is needed for use the screen, touch chip and SD capabilities.

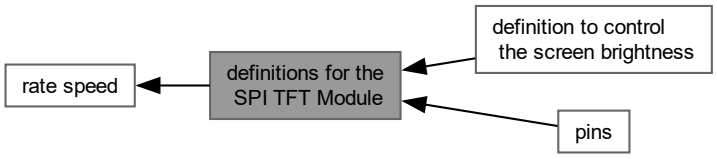
6.1.1 Detailed Description

Definition of baud rate speed

6.1.2 definitions for the SPI TFT Module

Pins connected to the device for use the SPI protocol. Is needed for use the screen, touch chip and SD capabilities.

Collaboration diagram for definitions for the SPI TFT Module:



Topics

- **definition to control the screen brightness**

  Pin to adjust the brightness of the display thanks to the mosfet n that is integrated in the ili9341 module.
- **pins**

  These pins manage the on/off of the device, reading the user activity and hold the enable buck with the 3.3V to supply the system.

Macros

- #define **TFT_MISO** 1

  Pin for the Master Input Slave Output.
- #define **TFT_MOSI** 2

  Pin fot the Master Output Slave Input.
- #define **TFT_SCLK** 1

  Pin for clock SPI.
- #define **TFT_CS** 1

  Pin for the Chip Select screen.
- #define **TFT_DC**

  Pin for data command.
- #define **TFT_RST**

  Pin for reset screen.
- #define **TOUCH_CS** 1

  Pin for chip select from the touch IC.
- #define **PIN_SD_CS**

  Pin for chip select from the SD.
- #define **SPI_FREQUENCY** 80000000

  SPI frequency.
- #define **SPI_READ_FREQUENCY** 20000000

  SPI read frequency.
- #define **SPI_TOUCH_FREQUENCY** 2500000

  SPI touch frequency.
- #define **PIN_EMITTER** 3

Emitter analog signal from ESP32 through BNC connector.

- #define **PIN_RECEIVER** 3

  Receiver analog signal from ESP32 through BNC connector.

- #define **PIN_THUMB_SW_CCW** 2

  Pin from the thumb button counter clockwise.

- #define **PIN_THUMB_SW_PUSH** 1

  Pin from the thumb button to push the center like a taditional button.

- #define **PIN_THUMB_SW_CW** 1

  Pin from the thumb button clockwise.

- #define **PIN_GPIO0**

  General purpouse input output 0 to flashing system.

- #define **PIN_I2C_SCL** 2

  I2C pin with clock signal.

- #define **PIN_I2C_SDA** 2

  I2C pin with data transfer.

6.1.2.1 Detailed Description

Pins connected to the device for use the SPI protocol. Is needed for use the screen, touch chip and SD capabilities.

Pin to define the I2C connections at the ESP32. Therefore, SDA Pin it's a bidirectional data transfer and SCL have the clock signal to coordinate the information between the multiples periferics like a RTC, thumb button ...

For definitions the system pulled up this pin. Must be LOW to enter flashing mode.

Pins connected to the thumb button to use the device without touching the screen.

Pins connected to the device to read the signals from the tree through at the BNC connectors.

6.1.2.2 Macro Definition Documentation

6.1.2.2.1 PIN_EMITTER

```
#define PIN_EMITTER 3
```

Emitter analog signal from ESP32 through BNC connector.

6.1.2.2.2 PIN_GPIO0

```
#define PIN_GPIO0
```

General purpouse input output 0 to flashing system.

6.1.2.2.3 PIN_I2C_SCL

```
#define PIN_I2C_SCL 2
```

I2C pin with clock signal.

6.1.2.2.4 PIN_I2C_SDA

```
#define PIN_I2C_SDA 2
```

I2C pin with data transfer.

6.1.2.2.5 PIN_RECEIVER

```
#define PIN_RECEIVER 3
```

Receiver analog signal from ESP32 through BNC connector.

6.1.2.2.6 PIN_SD_CS

```
#define PIN_SD_CS
```

Pin for chip select from the SD.

6.1.2.2.7 PIN_THUMB_SW_CCW

```
#define PIN_THUMB_SW_CCW 2
```

Pin from the thumb button counter clockwise.

6.1.2.2.8 PIN_THUMB_SW_CW

```
#define PIN_THUMB_SW_CW 1
```

Pin from the thumb button clockwise.

6.1.2.2.9 PIN_THUMB_SW_PUSH

```
#define PIN_THUMB_SW_PUSH 1
```

Pin from the thumb button to push the center like a taditional button.

6.1.2.2.10 SPI_FREQUENCY

```
#define SPI_FREQUENCY 80000000
```

SPI frequency.

6.1.2.2.11 SPI_READ_FREQUENCY

```
#define SPI_READ_FREQUENCY 20000000
```

SPI read frequency.

6.1.2.2.12 SPI_TOUCH_FREQUENCY

```
#define SPI_TOUCH_FREQUENCY 2500000
```

SPI touch frequency.

6.1.2.2.13 TFT_CS

```
#define TFT_CS 1
```

Pin for the Chip Select screen.

6.1.2.2.14 TFT_DC

```
#define TFT_DC
```

Pin for data command.

6.1.2.2.15 TFT_MISO

```
#define TFT_MISO 1
```

Pin for the Master Input Slave Output.

6.1.2.2.16 TFT_MOSI

```
#define TFT_MOSI 2
```

Pin fot the Master Output Slave Input.

6.1.2.2.17 TFT_RST

```
#define TFT_RST
```

Pin for reset screen.

6.1.2.2.18 TFT_SCLK

```
#define TFT_SCLK 1
```

Pin for clock SPI.

6.1.2.2.19 TOUCH_CS

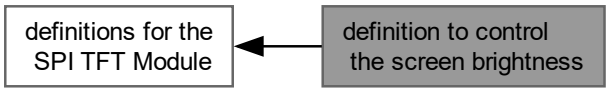
```
#define TOUCH_CS 1
```

Pin for chip select from the touch IC.

6.1.2.3 definition to control the screen brightness

Pin to adjust the brightness of the display thanks to the mosfet n that is integrated in the ili9341 module.

Collaboration diagram for definition to control the screen brightness:



Macros

- #define **PIN_LED** 1

  Brightness control through the N-Mosfet integrated on the module ILI9341.
- #define **PIN_TEST** 17

6.1.2.3.1 Detailed Description

Pin to adjust the brightness of the display thanks to the mosfet n that is integrated in the ili9341 module.

6.1.2.3.2 Macro Definition Documentation

6.1.2.3.2.1 PIN_LED

```
#define PIN_LED 1
```

Brightness control through the N-Mosfet integrated on the module ILI9341.

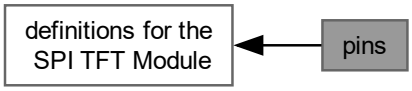
6.1.2.3.2.2 PIN_TEST

```
#define PIN_TEST 17
```

6.1.2.4 pins

These pins manage the on/off of the device, reading the user activity and hold the enable buck with the 3.3V to supply the system.

Collaboration diagram for pins:



Macros

- #define **PIN_POWER_SW_USER** 3

  Defined like a input pin to read the activity user.
- #define **PIN_POWER_SW_HOLD** 2

  Defined like a output pin to hold a high logic value and power on the enable buck.

6.1.2.4.1 Detailed Description

These pins manage the on/off of the device, reading the user activity and hold the enable buck with the 3.3V to supply the system.

6.1.2.4.2 Macro Definition Documentation

6.1.2.4.2.1 PIN_POWER_SW_HOLD

```
#define PIN_POWER_SW_HOLD 2
```

Defined like a output pin to hold a high logic value and power on the enable buck.

6.1.2.4.2.2 PIN_POWER_SW_USER

```
#define PIN_POWER_SW_USER 3
```

Defined like a input pin to read the activity user.

6.2 Proyects_pins

Chapter 7

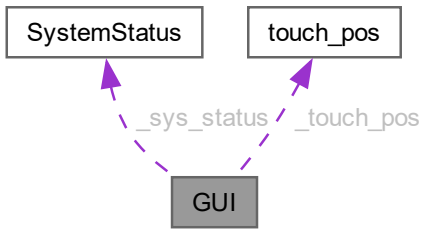
Class Documentation

7.1 GUI Class Reference

GUI (p. 19) class manages the graphical user interface for a TFT display.

`#include <GUI.h>`

Collaboration diagram for GUI:



Public Member Functions

- **GUI** (**SystemStatus** *sys_status, SemaphoreHandle_t *mutex_sys_status, **touch_pos** * **touch_pos**, SemaphoreHandle_t *mutex_touch_pos, **screens** *screen_id, SemaphoreHandle_t *mutex_spi)

  *Constructs a new **GUI** (p. 19) object.*
- void **drawStatusBar** ()

  Draws the status bar at the top of the screen.
- void **drawMainScreen** ()

  Draws the main screen content.
- void **drawConnectivityScreen** ()
- void **drawMeasureScreen** (TaskHandle_t *sampler_taskHandler)

  Draws the measurement screen content.
- void **drawSettingsScreen** ()

Draws the settings screen content.

- void **drawHelpScreen** ()

  Draws the help screen content.
- void **drawShutdownConfirmScreen** ()

  Draws the shutdown confirmation screen content.
- void **drawCurveOnGraph1** (uint16_t *vector, uint16_t util, uint16_t start, uint16_t end)

  Draws a curve on the first graph.
- void **drawCurveOnGraph2** (uint16_t *vector, uint16_t util, uint16_t start, uint16_t end)

  Draws a curve on the first graph.
- void **drawDisplayScreen** ()

  Draws the brightness adjustment screen content.
- void **drawBatteryScreen** ()

  Draws the battery status screen content.
- void **drawMemoryScreen** ()

  Draws the memory usage screen content.
- void **thumb_button** ()

  Placeholder function for handling thumb button events.
- void **_infoMemory** (fs::FS &fs, const char *path, const char *message)

  Displays information about memory usage.
- void **_appendMemory** (fs::FS &fs, const char *path, const char *message)

  Appends memory usage information.
- void **drawEngineerMode** ()

  Appends memory usage information.
- void **init** ()

  *Initializes the **GUI** (p. 19) and TFT display.*
- void **calibrate** ()

  Performs screen calibration if necessary.

Public Attributes

- TFT_eSPI **_tft** = TFT_eSPI()

  TFT_eSPI library instance.
- TFT_eSprite **_spr_bt** = TFT_eSprite(& **_tft**)

  TFT_eSprite instance for buttons.
- bool **bluetooth_on_off** = false
- bool **wifi_on_off** = false

Private Member Functions

- bool **_buttonMonitor** (**TFT_eSPI_Button_Mod** *button)

  Monitors the state of a button.
- void **_drawTitle** ()

  Draws the title bar at the top of the screen.
- void **_clearScreen** ()

  Clears the entire screen.
- void **_clearScreenEngineerMode** ()
- void **_drawGraphGrid** (uint16_t y_pos, float x_lims[2], float y1_lims[2], float y2_lims[2], char *title, char *x_title, char *y1_title, char *y2_title)

  Draws a grid for a graph on the screen.
- void **_drawGraphs** ()

  Draws various graphs on the screen.
- void **_drawCurve** (uint16_t *vector, uint16_t util, uint16_t start_index, uint16_t end_index, uint16_t x_↩ graph_start, uint16_t x_graph_end, uint16_t y_graph_start, uint16_t y_graph_end)

  Draws a curve on a graph.

Private Attributes

- **SystemStatus** * **_sys_status**
  Pointer to system status object.
- SemaphoreHandle_t * **_mutex_sys_status**
  Mutex for system status access.
- **touch_pos** * **_touch_pos**
  Pointer to touch position structure.
- SemaphoreHandle_t * **_mutex_touch_pos**
  Mutex for touch position access.
- **screens** * **_screen_id**
  Pointer to current screen ID.
- SemaphoreHandle_t * **_mutex_spi**
  Mutex for SPI communication.

7.1.1 Detailed Description

GUI (p. 19) class manages the graphical user interface for a TFT display.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 GUI()

```
GUI::GUI (
              SystemStatus * sys_status,
             SemaphoreHandle_t * mutex_sys_status,
              touch_pos * touch_pos,
             SemaphoreHandle_t * mutex_touch_pos,
              screens * screen_id,
             SemaphoreHandle_t * mutex_spi)
```

Constructs a new **GUI** (p. 19) object.

Parameters

sys_status	Pointer to system status object.
mutex_sys_status	Mutex for system status access.
touch_pos *(p. 65)*	Pointer to touch position structure.
mutex_touch_pos	Mutex for touch position access.
screen_id	Pointer to current screen ID.
mutex_spi	Mutex for SPI communication.

7.1.3 Member Function Documentation

7.1.3.1 _appendMemory()

```
void GUI::_appendMemory (
             fs::FS & fs,
             const char * path,
             const char * message)
```

Appends memory usage information.

Parameters

fs	File system object.
path	Path to the file.
message	Message to append.

Here is the caller graph for this function:

7.1.3.2 _buttonMonitor()

```
bool GUI::_buttonMonitor (
            TFT_eSPI_Button_Mod * button)  [private]
```

Monitors the state of a button.

Parameters

button	Pointer to the button object to monitor.

Returns

True if the button state changed, false otherwise.

Monitors the state of a button.

Parameters

button	Pointer to the button object to monitor.

Returns

True if the button state changed, false otherwise.

Parameters

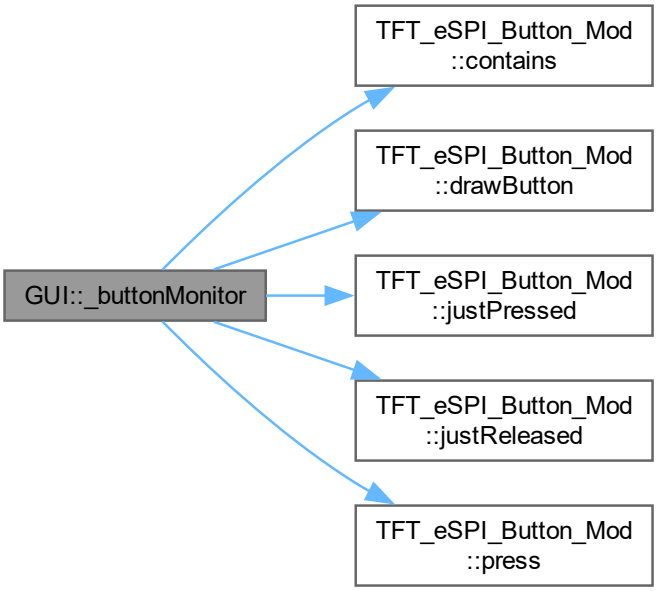
button	

Returns

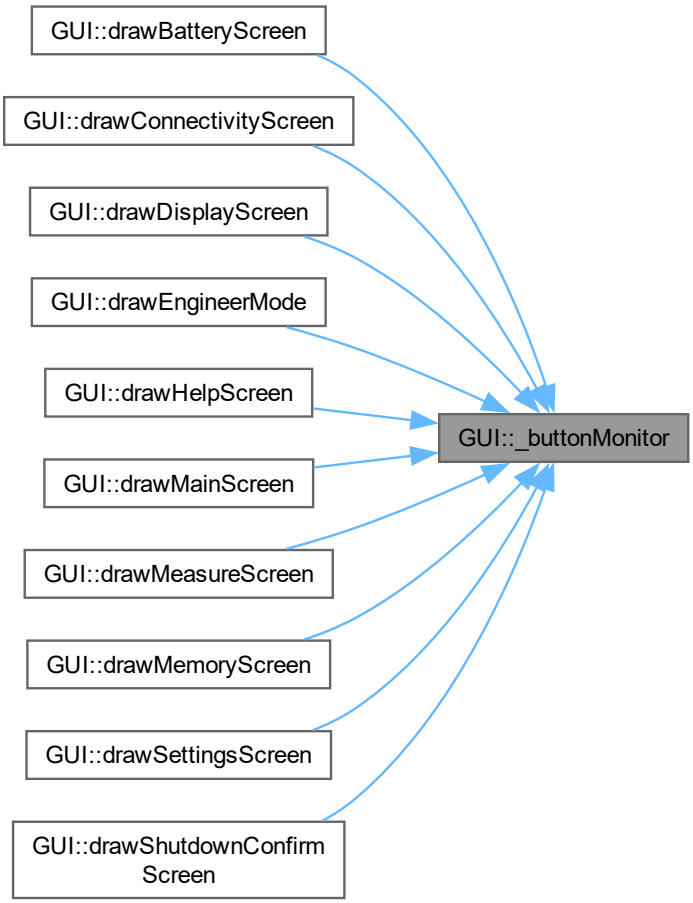
true

false

Here is the call graph for this function:



Here is the caller graph for this function:

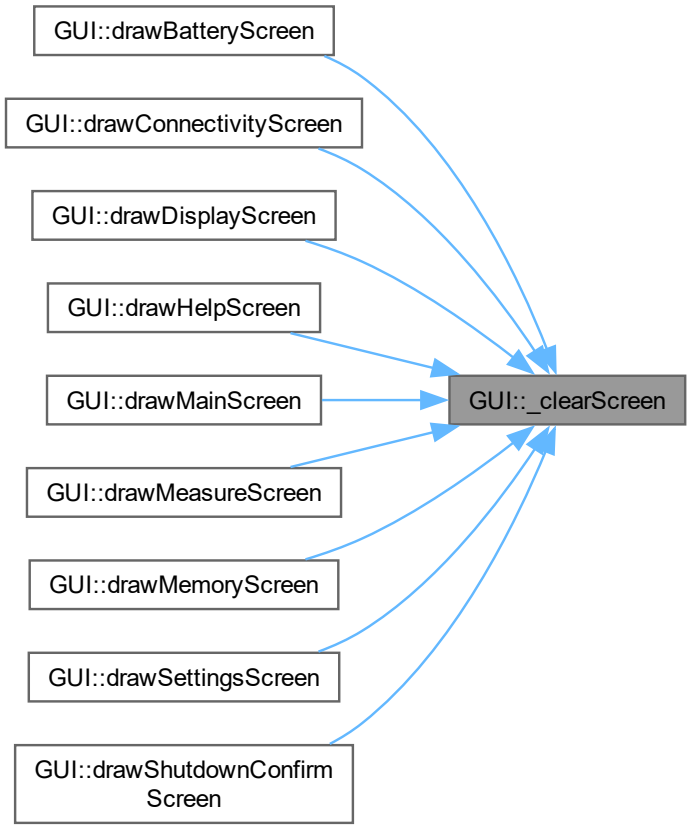


7.1.3.3 _clearScreen()

```
void GUI::_clearScreen ()  [private]
```

Clears the entire screen.

Here is the caller graph for this function:



7.1.3.4 _clearScreenEngineerMode()

```
void GUI::_clearScreenEngineerMode ()  [private]
```

Here is the caller graph for this function:

7.1.3.5 _drawCurve()

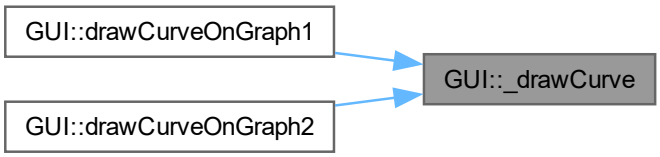
```
void GUI::_drawCurve (
            uint16_t * vector,
            uint16_t util,
            uint16_t start_index,
            uint16_t end_index,
            uint16_t x_graph_start,
            uint16_t x_graph_end,
            uint16_t y_graph_start,
            uint16_t y_graph_end)  [private]
```

Draws a curve on a graph.

Parameters

vector	Pointer to the data vector.
util	Number of elements used in the vector.
start_index	Start index of the curve data.
end_index	End index of the curve data.
x_graph_start	Starting x-coordinate of the graph area.
x_graph_end	Ending x-coordinate of the graph area.
y_graph_start	Starting y-coordinate of the graph area.
y_graph_end	Ending y-coordinate of the graph area.
vector	
util	
start_index	
end_index	
x_graph_start	
x_graph_end	
y_graph_start	
y_graph_end	

Here is the caller graph for this function:



7.1.3.6 _drawGraphGrid()

```
void GUI::_drawGraphGrid (
            uint16_t y_pos,
```

```
float x_lims[2],
float y1_lims[2],
float y2_lims[2],
char * title,
char * x_title,
char * y1_title,
char * y2_title)  [private]
```

Draws a grid for a graph on the screen.

Parameters

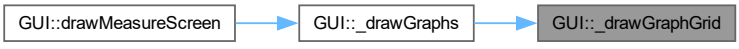
y_pos	Vertical position of the graph on the screen.
x_lims	Limits of the x-axis [start, end].
y1_lims	Limits of the primary y-axis [start, end].
y2_lims	Limits of the secondary y-axis [start, end].
title	Title of the graph.
x_title	Title of the x-axis.
y1_title	Title of the primary y-axis.
y2_title	Title of the secondary y-axis.
button	

Returns

true

false

Here is the caller graph for this function:

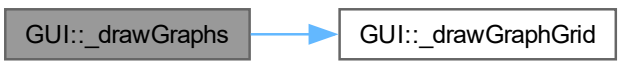


7.1.3.7 _drawGraphs()

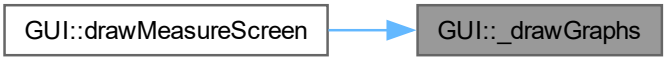
```
void GUI::_drawGraphs ()  [private]
```

Draws various graphs on the screen.

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.3.8 _drawTitle()

```
void GUI::_drawTitle ()  [private]
```

Draws the title bar at the top of the screen.

7.1.3.9 _infoMemory()

```
void GUI::_infoMemory (
            fs::FS & fs,
            const char * path,
            const char * message)
```

Displays information about memory usage.

Parameters

fs	File system object.
path	Path to the file.
message	Message to display.

7.1.3.10 calibrate()

```
void GUI::calibrate ()
```

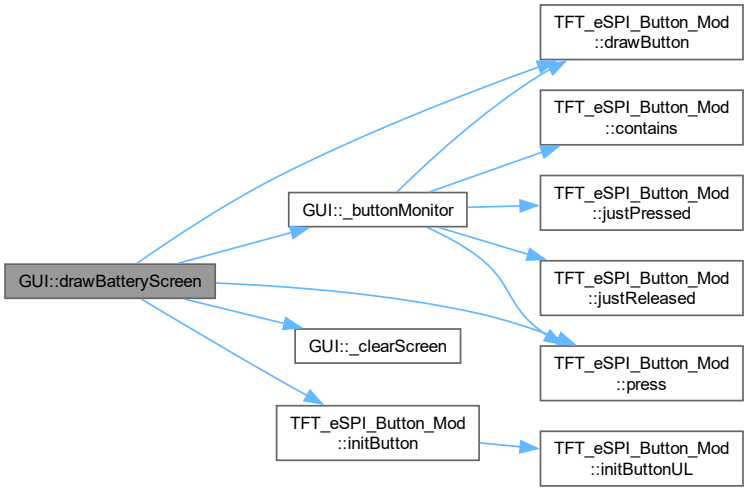
Performs screen calibration if necessary.

7.1.3.11 drawBatteryScreen()

```
void GUI::drawBatteryScreen ()
```

Draws the battery status screen content.

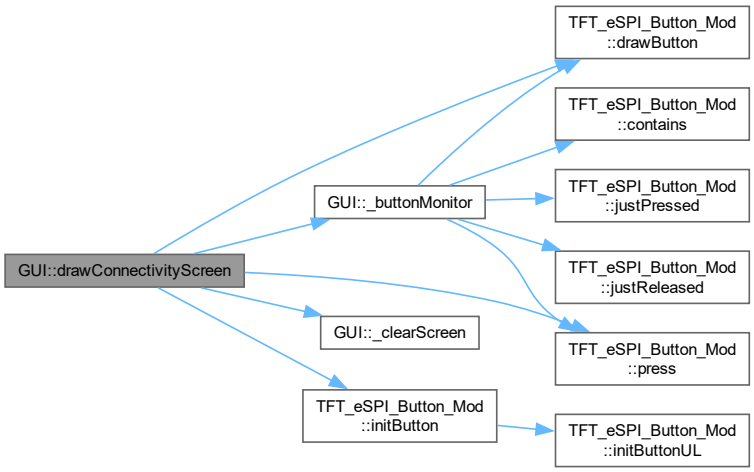
Here is the call graph for this function:



7.1.3.12 drawConnectivityScreen()

```
void GUI::drawConnectivityScreen ()
```

Here is the call graph for this function:



7.1.3.13 drawCurveOnGraph1()

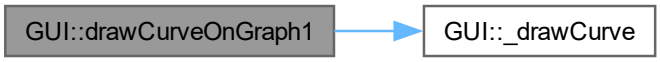
```
void GUI::drawCurveOnGraph1 (
            uint16_t * vector,
            uint16_t util,
            uint16_t start_index,
            uint16_t end_index)
```

Draws a curve on the first graph.

Parameters

vector	Pointer to the data vector.
util	Number of elements used in the vector.
start	Start index of the curve data.
end	End index of the curve data.
vector	
util	
start_index	
end_index	

Here is the call graph for this function:



7.1.3.14 drawCurveOnGraph2()

```
void GUI::drawCurveOnGraph2 (
            uint16_t * vector,
            uint16_t util,
            uint16_t start_index,
            uint16_t end_index)
```

Draws a curve on the first graph.

Parameters

vector	Pointer to the data vector.
util	Number of elements used in the vector.
start	Start index of the curve data.
end	End index of the curve data.
vector	
util	
start_index	

end_index	

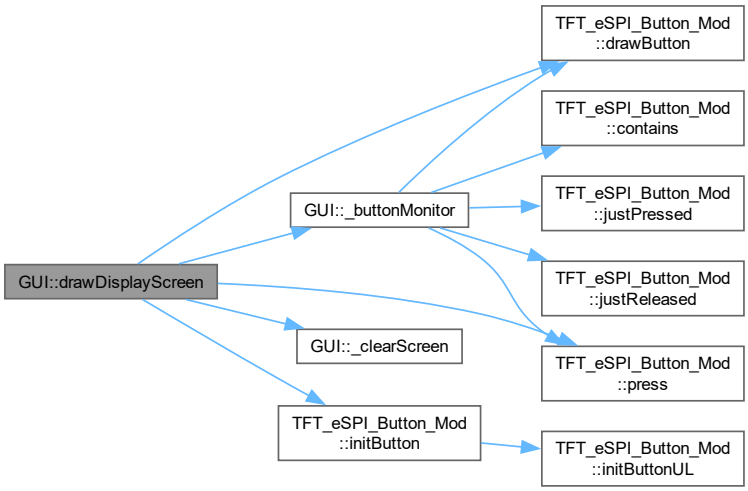
Here is the call graph for this function:

7.1.3.15 drawDisplayScreen()

```
void GUI::drawDisplayScreen ()
```

Draws the brightness adjustment screen content.

Here is the call graph for this function:



7.1.3.16 drawEngineerMode()

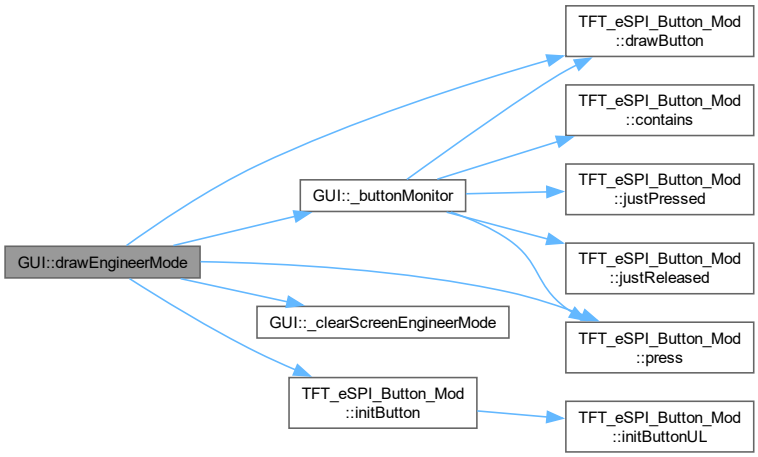
```
void GUI::drawEngineerMode ()
```

Appends memory usage information.

Parameters

fs	File system object.
path	Path to the file.
message	Message to append.

Here is the call graph for this function:

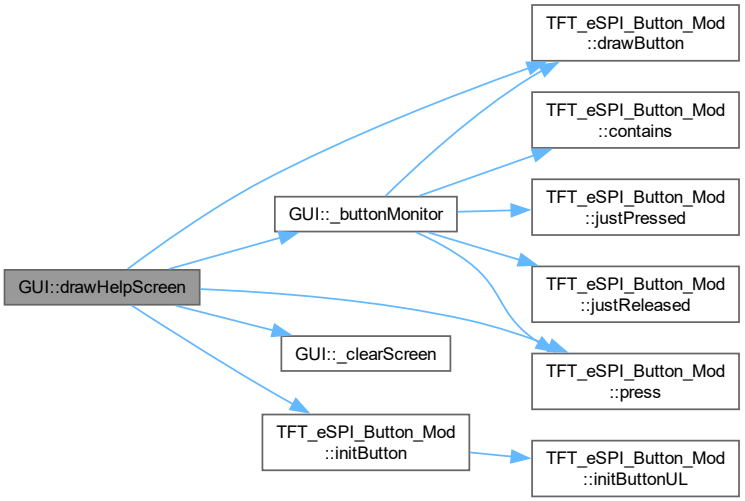


7.1.3.17 drawHelpScreen()

```
void GUI::drawHelpScreen ()
```

Draws the help screen content.

Here is the call graph for this function:

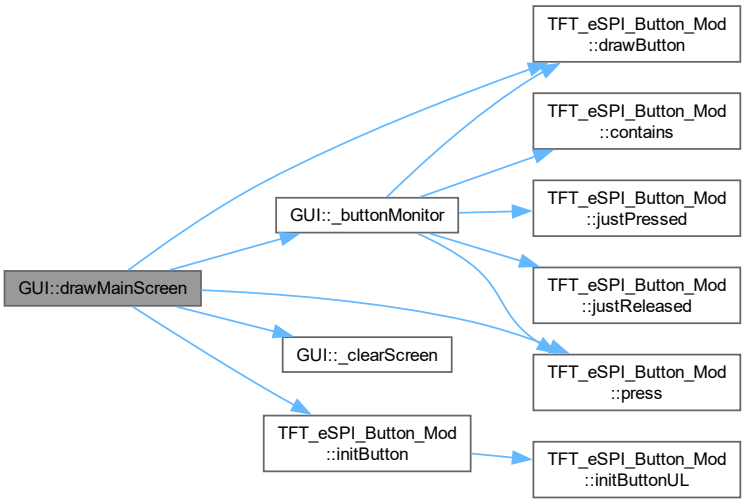


7.1.3.18 drawMainScreen()

```
void GUI::drawMainScreen ()
```

Draws the main screen content.

Here is the call graph for this function:



7.1.3.19 drawMeasureScreen()

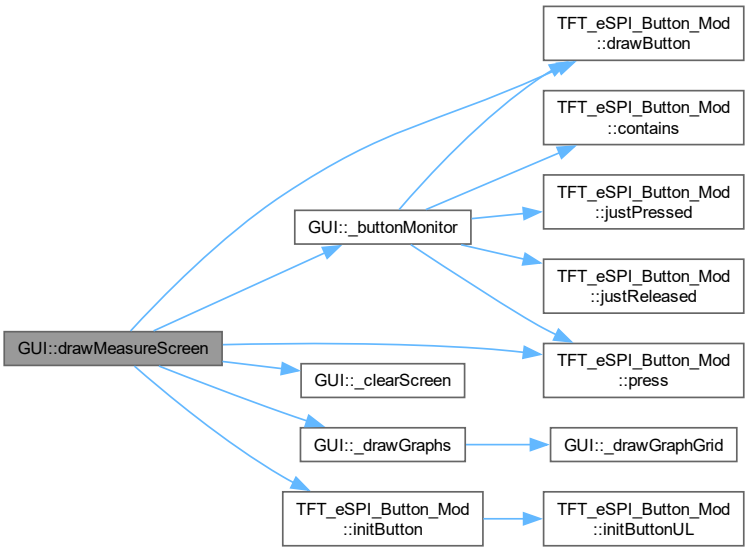
```
void GUI::drawMeasureScreen (
            TaskHandle_t * sampler_taskHandler)
```

Draws the measurement screen content.

Parameters

sampler_taskHandler	Handle to the sampling task.

Here is the call graph for this function:

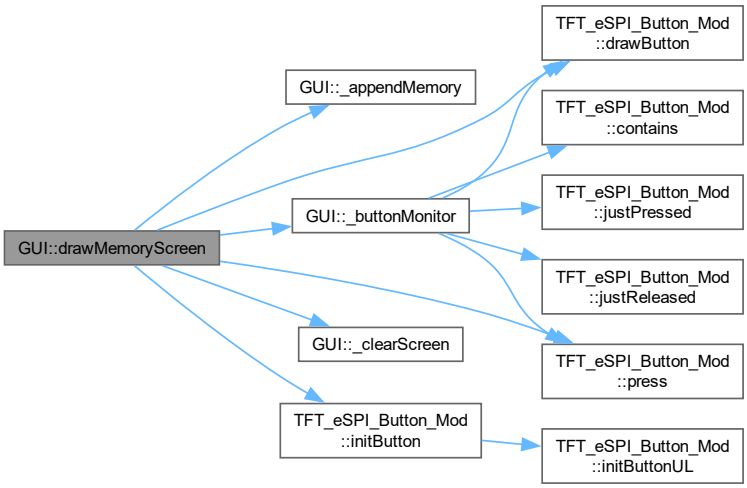


7.1.3.20 drawMemoryScreen()

```
void GUI::drawMemoryScreen ()
```

Draws the memory usage screen content.

Here is the call graph for this function:

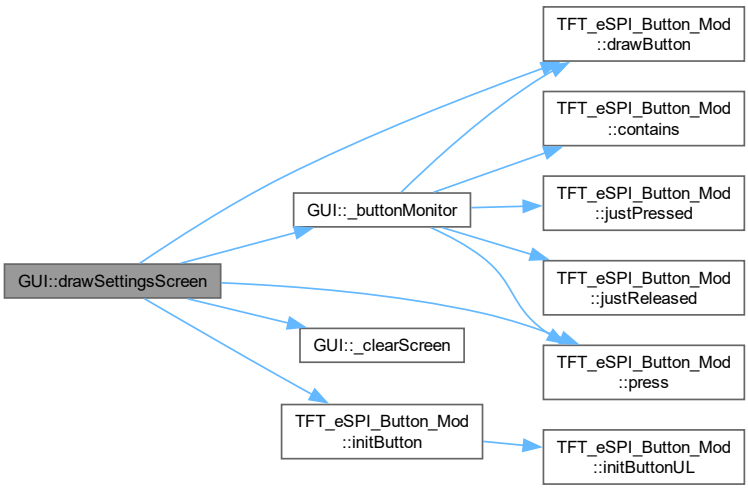


7.1.3.21 drawSettingsScreen()

```
void GUI::drawSettingsScreen ()
```

Draws the settings screen content.

Here is the call graph for this function:

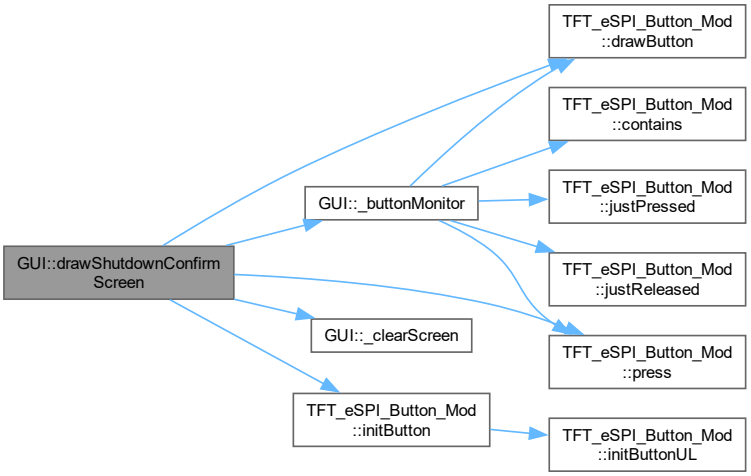


7.1.3.22 drawShutdownConfirmScreen()

```
void GUI::drawShutdownConfirmScreen ()
```

Draws the shutdown confirmation screen content.

Here is the call graph for this function:



7.1.3.23 drawStatusBar()

```
void GUI::drawStatusBar ()
```

Draws the status bar at the top of the screen.

7.1.3.24 init()

```
void GUI::init ()
```

Initializes the **GUI** (p. 19) and TFT display.

7.1.3.25 thumb_button()

```
void GUI::thumb_button ()
```

Placeholder function for handling thumb button events.

7.1.4 Member Data Documentation

7.1.4.1 _mutex_spi

```
SemaphoreHandle_t* GUI::_mutex_spi  [private]
```

Mutex for SPI communication.

7.1.4.2 _mutex_sys_status

```
SemaphoreHandle_t* GUI::_mutex_sys_status  [private]
```

Mutex for system status access.

7.1.4.3 _mutex_touch_pos

```
SemaphoreHandle_t* GUI::_mutex_touch_pos  [private]
```

Mutex for touch position access.

7.1.4.4 _screen_id

```
screens* GUI::_screen_id  [private]
```

Pointer to current screen ID.

7.1.4.5 _spr_bt

```
TFT_eSprite GUI::_spr_bt = TFT_eSprite(& _tft)
```

TFT_eSprite instance for buttons.

7.1.4.6 _sys_status

```
SystemStatus* GUI::_sys_status  [private]
```

Pointer to system status object.

7.1.4.7 _tft

```
TFT_eSPI GUI::_tft = TFT_eSPI()
```

TFT_eSPI library instance.

7.1.4.8 _touch_pos

```
touch_pos* GUI::_touch_pos  [private]
```

Pointer to touch position structure.

7.1.4.9 bluetooth_on_off

```
bool GUI::bluetooth_on_off = false
```

7.1.4.10 wifi_on_off

```
bool GUI::wifi_on_off = false
```

The documentation for this class was generated from the following files:

- src/GUI/ **GUI.h**
- src/GUI/ **GUI.cpp**

7.2 Public Class Reference

Class to handle the touch UI interface of TFT_eSPI.

7.2.1 Detailed Description

Class to handle the touch UI interface of TFT_eSPI.

The documentation for this class was generated from the following file:

- src/GUI/button/ **ButtonMod.h**

7.3 SystemStatus Struct Reference

Structure to hold various system status information.

```
#include <SystemStatus.h>
```

Public Attributes

- uint8_t **battery_level_percentage**
- uint16_t **battery_voltage_V**
- uint16_t **battery_current_mA**
- uint16_t **battery_power_mW**
- uint16_t **battery_total_capacity_mWh**
- uint16_t **battery_used_capacity_mWh**
- bool **battery_charging**
- bool **battery_fully_charged**
- bool **battery_100**
- bool **battery_75**
- bool **battery_50**
- bool **battery_25**
- bool **wifi_connected**
- int8_t **wifi_rssi_dBm**
- bool **bt_connected**
- int8_t **bt_rssi_dBm**
- bool **triggered**
- bool **ready**

7.3.1 Detailed Description

Structure to hold various system status information.

7.3.2 Member Data Documentation

7.3.2.1 battery_100

```
bool SystemStatus::battery_100
```

7.3.2.2 battery_25

```
bool SystemStatus::battery_25
```

7.3.2.3 battery_50

```
bool SystemStatus::battery_50
```

7.3.2.4 battery_75

```
bool SystemStatus::battery_75
```

7.3.2.5 battery_charging

```
bool SystemStatus::battery_charging
```

7.3.2.6 battery_current_mA

```
uint16_t SystemStatus::battery_current_mA
```

7.3.2.7 battery_fully_charged

```
bool SystemStatus::battery_fully_charged
```

7.3.2.8 battery_level_percentage

```
uint8_t SystemStatus::battery_level_percentage
```

7.3.2.9 battery_power_mW

```
uint16_t SystemStatus::battery_power_mW
```

7.3.2.10 battery_total_capacity_mWh

```
uint16_t SystemStatus::battery_total_capacity_mWh
```

7.3.2.11 battery_used_capacity_mWh

```
uint16_t SystemStatus::battery_used_capacity_mWh
```

7.3.2.12 battery_voltage_V

```
uint16_t SystemStatus::battery_voltage_V
```

7.3.2.13 bt_connected

```
bool SystemStatus::bt_connected
```

7.3.2.14 bt_rssi_dBm

```
int8_t SystemStatus::bt_rssi_dBm
```

7.3.2.15 ready

```
bool SystemStatus::ready
```

7.3.2.16 triggered

```
bool SystemStatus::triggered
```

7.3.2.17 wifi_connected

```
bool SystemStatus::wifi_connected
```

7.3.2.18 wifi_rssi_dBm

```
int8_t SystemStatus::wifi_rssi_dBm
```

The documentation for this struct was generated from the following file:

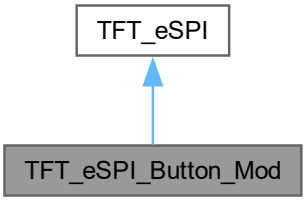
- src/SYSTEM/ **SystemStatus.h**

7.4 TFT_eSPI_Button_Mod Class Reference

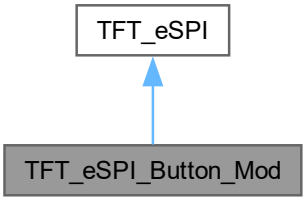
Class to handle buttons modified from the TFT_eSPI library.

```
#include <ButtonMod.h>
```

Inheritance diagram for TFT_eSPI_Button_Mod:



Collaboration diagram for TFT_eSPI_Button_Mod:



Public Member Functions

- **TFT_eSPI_Button_Mod** (void)

  *Constructor for the **TFT_eSPI_Button_Mod*** (p. *41*) *class.*
- void **initButton** (TFT_eSPI ∗gfx, int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t outline, uint16_t fill, uint16_t textcolor, char ∗label, uint8_t textfont)

  Initialize the button using the center and size.
- void **initButtonUL** (TFT_eSPI ∗gfx, int16_t x1, int16_t y1, uint16_t w, uint16_t h, uint16_t outline, uint16_t fill, uint16_t textcolor, char ∗label, uint8_t textfont)

  Initialize the button using the top-left corner and size.
- void **setLabelDatum** (int16_t x_delta, int16_t y_delta, uint8_t datum=MC_DATUM)

  Set the text datum and x, y deltas.
- void **drawButton** (bool inverted=false, String long_name="")

  Draw the button on the screen.
- bool **contains** (int16_t x, int16_t y)

  Check if the coordinates (x, y) are within the button.
- void **press** (bool p)

  Set the button's pressed state.
- bool **isPressed** ()

  Check if the button is pressed.
- bool **justPressed** ()

  Check if the button was just pressed.
- bool **justReleased** ()

  Check if the button was just released.

Private Attributes

- TFT_eSPI ∗ **_gfx**
- int16_t **_x1** = 0
- int16_t **_y1** = 0
- int16_t **_xd** = 0
- int16_t **_yd** = 0
- uint16_t **_w** = 0
- uint16_t **_h** = 0
- uint8_t **_textfont** = 0
- uint8_t **_textdatum**
- uint16_t **_outlinecolor** = 0
- uint16_t **_fillcolor** = 0
- uint16_t **_textcolor** = 0
- char **_label** [10]
- bool **currstate** = false
- bool **laststate** = false

7.4.1 Detailed Description

Class to handle buttons modified from the TFT_eSPI library.

Private class to handle buttons modified from the TFT_eSPI library.

Private part of the class to handle buttons modified from the TFT_eSPI library.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 TFT_eSPI_Button_Mod()

```
TFT_eSPI_Button_Mod::TFT_eSPI_Button_Mod (
            void )
```

Constructor for the **TFT_eSPI_Button_Mod** (p. 41) class.

7.4.3 Member Function Documentation

7.4.3.1 contains()

```
bool TFT_eSPI_Button_Mod::contains (
            int16_t x,
            int16_t y)
```

Check if the coordinates (x, y) are within the button.

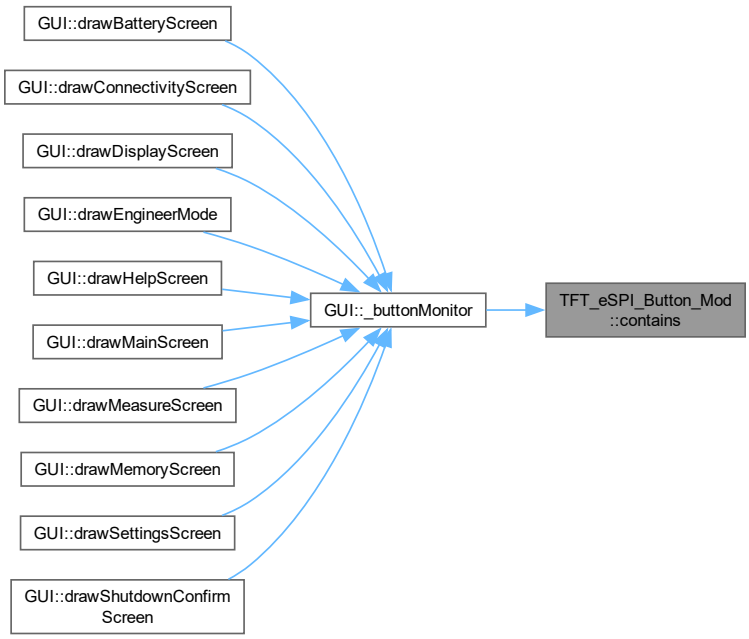
Parameters

x	X-coordinate.
y	Y-coordinate.

Returns

true if the coordinates are within the button, false otherwise.

Here is the caller graph for this function:



7.4.3.2 drawButton()

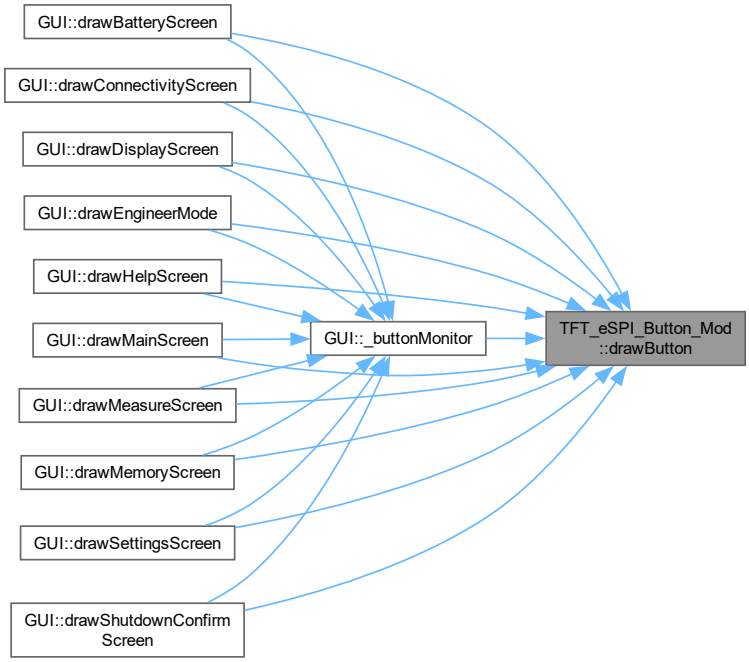
```
void TFT_eSPI_Button_Mod::drawButton (
            bool inverted = false,
            String long_name = "")
```

Draw the button on the screen.

Parameters

inverted	Indicates if the button should be drawn inverted.
long_name	Long name for the button label.

Here is the caller graph for this function:



7.4.3.3 initButton()

```
void TFT_eSPI_Button_Mod::initButton (
            TFT_eSPI * gfx,
            int16_t x,
            int16_t y,
            uint16_t w,
            uint16_t h,
            uint16_t outline,
            uint16_t fill,
            uint16_t textcolor,
```

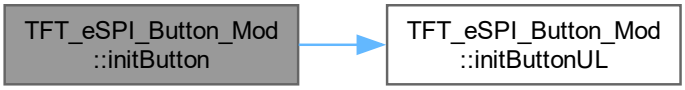
```
            char * label,
            uint8_t textfont)
```

Initialize the button using the center and size.

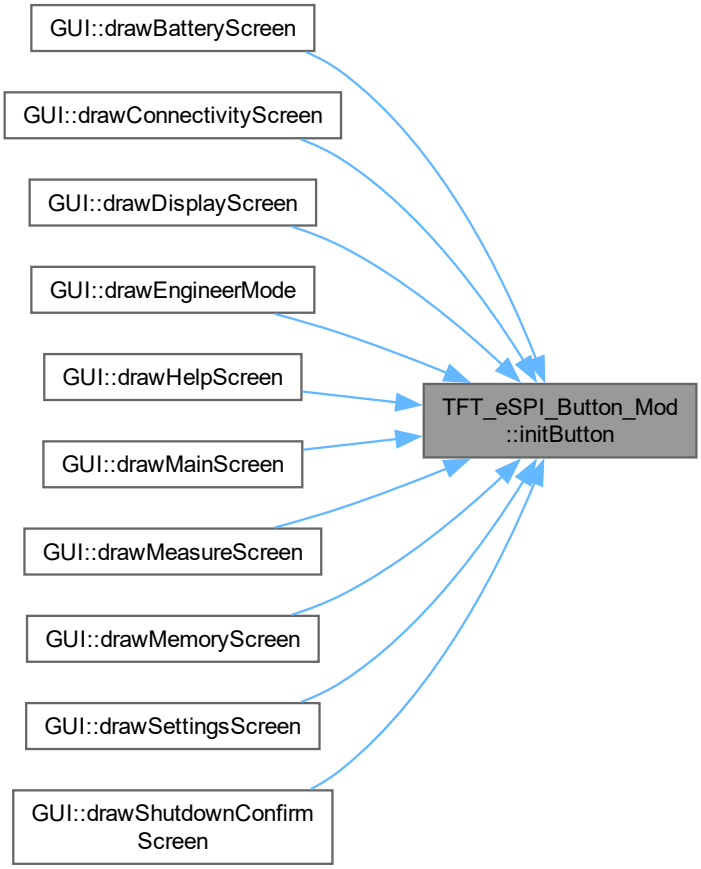
Parameters

gfx	Pointer to the TFT_eSPI object.
x	X-coordinate of the button center.
y	Y-coordinate of the button center.
w	Width of the button.
h	Height of the button.
outline	Outline color of the button.
fill	Fill color of the button.
textcolor	Text color of the button.
label	Button label.
textfont	Text font of the button.

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3.4 initButtonUL()

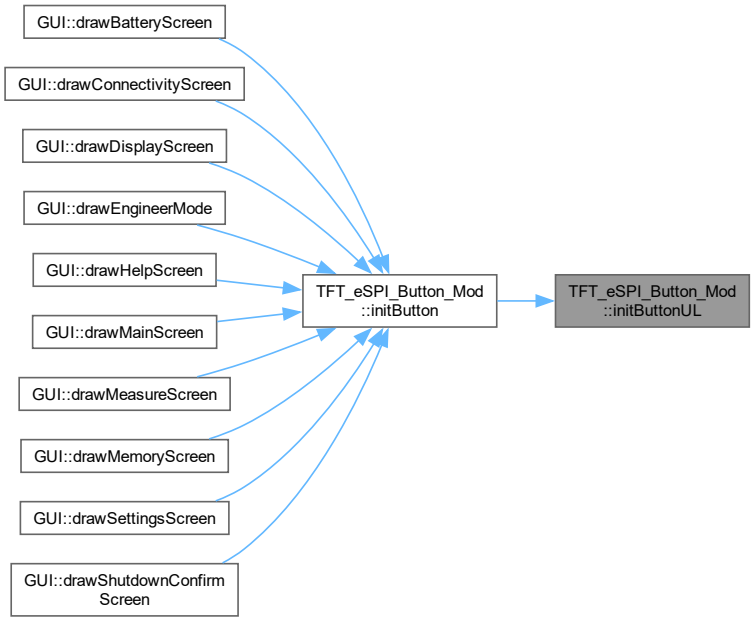
```
void TFT_eSPI_Button_Mod::initButtonUL (
             TFT_eSPI * gfx,
             int16_t x1,
             int16_t y1,
             uint16_t w,
             uint16_t h,
             uint16_t outline,
             uint16_t fill,
             uint16_t textcolor,
             char * label,
             uint8_t textfont)
```

Initialize the button using the top-left corner and size.

Parameters

gfx	Pointer to the TFT_eSPI object.
x1	X-coordinate of the top-left corner of the button.
y1	Y-coordinate of the top-left corner of the button.
w	Width of the button.
h	Height of the button.
outline	Outline color of the button.
fill	Fill color of the button.
textcolor	Text color of the button.
label	Button label.
textfont	Text font of the button.

Here is the caller graph for this function:



7.4.3.5 isPressed()

```
bool TFT_eSPI_Button_Mod::isPressed ()
```

Check if the button is pressed.

Returns

true if the button is pressed, false otherwise.

7.4.3.6 justPressed()

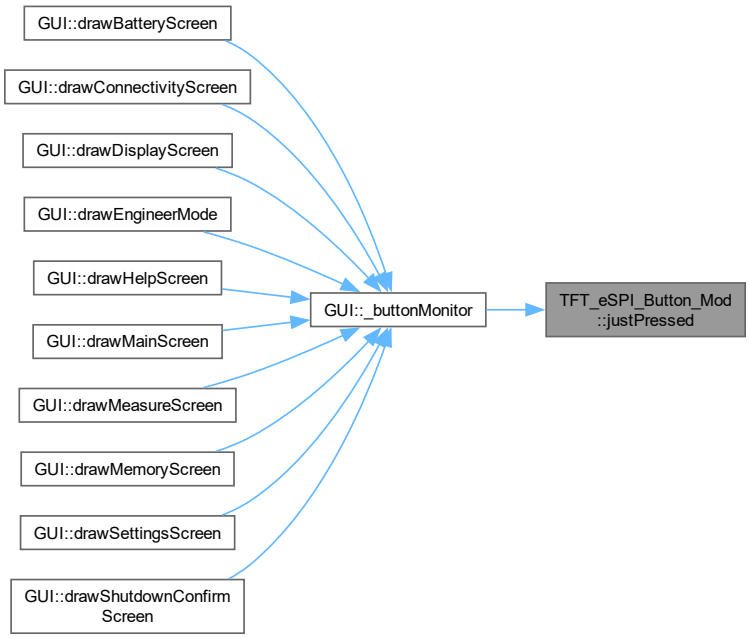
```
bool TFT_eSPI_Button_Mod::justPressed ()
```

Check if the button was just pressed.

Returns

true if the button was just pressed, false otherwise.

Here is the caller graph for this function:



7.4.3.7 justReleased()

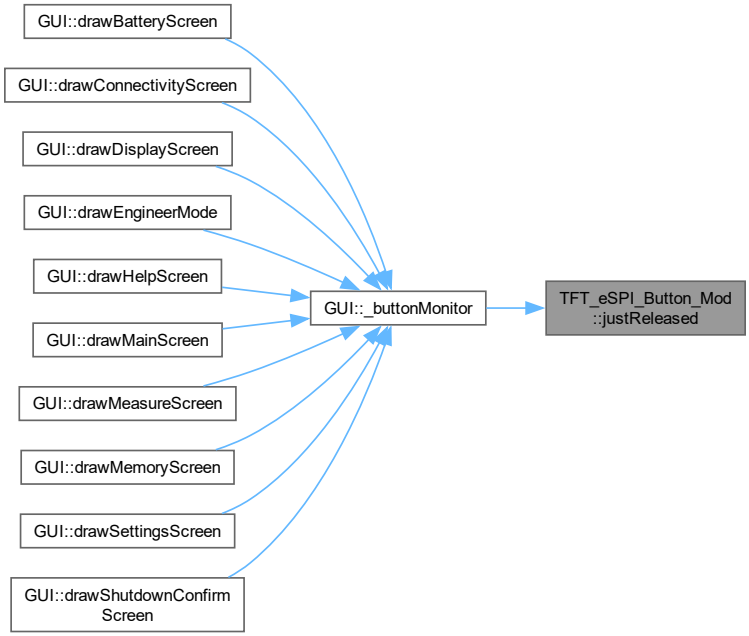
```
bool TFT_eSPI_Button_Mod::justReleased ()
```

Check if the button was just released.

Returns

true if the button was just released, false otherwise.

Here is the caller graph for this function:



7.4.3.8 press()

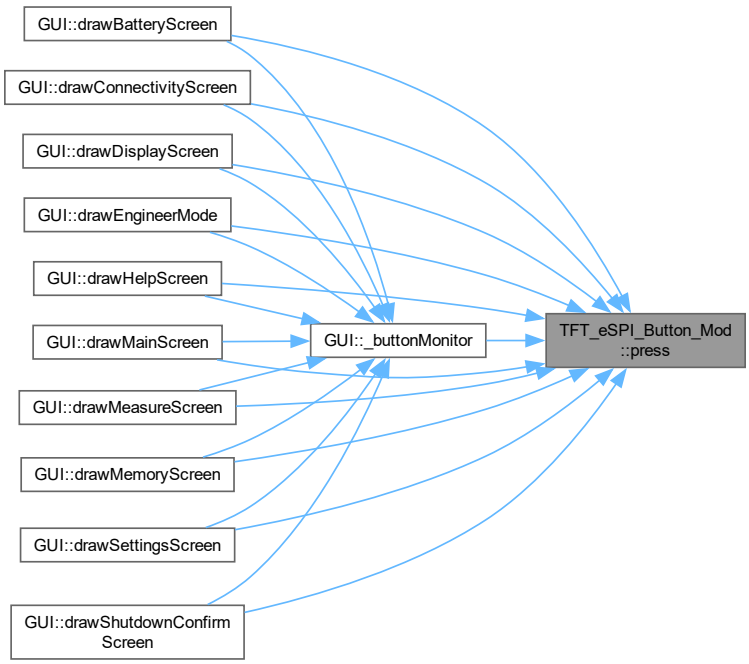
```
void TFT_eSPI_Button_Mod::press (
            bool p)
```

Set the button's pressed state.

Parameters

p	Pressed state of the button.

Here is the caller graph for this function:



7.4.3.9 setLabelDatum()

```
void TFT_eSPI_Button_Mod::setLabelDatum (
             int16_t x_delta,
             int16_t y_delta,
             uint8_t datum = MC_DATUM)
```

Set the text datum and x, y deltas.

Parameters

x_delta	X delta for the text position.
y_delta	Y delta for the text position.
datum	Text datum.

7.4.4 Member Data Documentation

7.4.4.1 _fillcolor

```
uint16_t TFT_eSPI_Button_Mod::_fillcolor = 0  [private]
```

7.4.4.2 _gfx

```
TFT_eSPI* TFT_eSPI_Button_Mod::_gfx  [private]
```

7.4.4.3 _h

```
uint16_t TFT_eSPI_Button_Mod::_h = 0  [private]
```

7.4.4.4 _label

```
char TFT_eSPI_Button_Mod::_label[10]  [private]
```

7.4.4.5 _outlinecolor

```
uint16_t TFT_eSPI_Button_Mod::_outlinecolor = 0  [private]
```

7.4.4.6 _textcolor

```
uint16_t TFT_eSPI_Button_Mod::_textcolor = 0  [private]
```

7.4.4.7 _textdatum

```
uint8_t TFT_eSPI_Button_Mod::_textdatum  [private]
```

7.4.4.8 _textfont

```
uint8_t TFT_eSPI_Button_Mod::_textfont = 0  [private]
```

7.4.4.9 _w

```
uint16_t TFT_eSPI_Button_Mod::_w = 0  [private]
```

7.4.4.10 _x1

```
int16_t TFT_eSPI_Button_Mod::_x1 = 0  [private]
```

7.4.4.11 _xd

```
int16_t TFT_eSPI_Button_Mod::_xd = 0  [private]
```

7.4.4.12 _y1

```
int16_t TFT_eSPI_Button_Mod::_y1 = 0  [private]
```

7.4.4.13 _yd

```
int16_t TFT_eSPI_Button_Mod::_yd = 0  [private]
```

7.4.4.14 currstate

```
bool TFT_eSPI_Button_Mod::currstate = false  [private]
```

7.4.4.15 laststate

```
bool TFT_eSPI_Button_Mod::laststate = false  [private]
```

The documentation for this class was generated from the following files:

- src/GUI/button/ **ButtonMod.h**
- src/GUI/button/ **ButtonMod.cpp**

7.5 TFT_eSPI_TouchUI Class Reference

```
#include <ButtonMod.h>
```

Public Member Functions

- **TFT_eSPI_TouchUI** (void)

  *Constructor for the **TFT_eSPI_TouchUI*** (p. *53) class.*
- void **initButton** (TFT_eSPI ∗gfx, int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t outline, uint16_t oncolor, uint16_t offcolor, char ∗label, uint8_t knob)

  Initialize the button using the center and size.
- void **initButtonUL** (TFT_eSPI ∗gfx, int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t outline, uint16_t oncolor, uint16_t offcolor, char ∗label, uint8_t knob)

  Initialize the button using the top-left corner and size.
- void **drawButton** (boolean inverted=false)

  Draw the button on the screen.
- void **initButtonG** (TFT_eSPI ∗gfx, int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t oncolor, uint16_t offcolor, const unsigned char ∗image, uint8_t knob)

  Initialize the graphic button using the center and size.
- void **initButtonGUL** (TFT_eSPI ∗gfx, int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t oncolor, uint16_t offcolor, const unsigned char ∗image, uint8_t knob)

  Initialize the graphic button using the top-left corner and size.
- void **drawButtonG** (boolean inverted=false)

  Draw the graphic button on the screen.
- boolean **contains** (int16_t x, int16_t y)

Check if the coordinates (x, y) are within the button.

- void **press** (boolean p)

  Set the button's pressed state.
- boolean **isPressed** ()

  Check if the button is pressed.
- boolean **justPressed** ()

  Check if the button was just pressed.
- boolean **justReleased** ()

  Check if the button was just released.
- void **initButtonS** (TFT_eSPI ∗gfx, int16_t x, int16_t y, uint16_t w, uint16_t l, uint16_t outline, uint16_t oncolor, uint16_t offcolor, uint8_t knob, boolean lb)

  Initialize the slider button.
- void **drawButtonS** (boolean inverted=false)

  Draw the slider button on the screen.
- boolean **justPressedOn** (int16_t x)

  Check if the slider button was just pressed at the x position.
- boolean **justPressedOff** (int16_t x)

  Check if the slider button was just released at the x position.
- void **initSliderH** (TFT_eSPI ∗gfx, int16_t mim1, int16_t max1, int16_t x, int16_t y, uint16_t w, uint16_t l, uint16_t outline, uint16_t fill, uint16_t nofill, uint8_t knob, int8_t dv)

  Initialize the horizontal slider.
- void **drawSliderH** (int16_t x)

  Draw the horizontal slider on the screen.
- boolean **containsH** (int16_t x, int16_t y)

  Check if the coordinates (x, y) are within the horizontal slider.
- int16_t **getValueH** (int16_t v)

  Get the value of the horizontal slider.
- void **initSliderV** (TFT_eSPI ∗gfx, int16_t mim1, int16_t max1, int16_t x, int16_t y, uint16_t w, uint16_t l, uint16_t outline, uint16_t fill, uint16_t nofill, uint8_t knob, int8_t dv)

  Initialize the vertical slider.
- void **drawSliderV** (int16_t x)

  Draw the vertical slider on the screen.
- boolean **containsV** (int16_t x, int16_t y)

  Check if the coordinates (x, y) are within the vertical slider.
- int16_t **getValueV** (int16_t v)

  Get the value of the vertical slider.

Private Attributes

- TFT_eSPI ∗ **_gfx**
- int16_t **_x**
- int16_t **_y**

7.5.1 Constructor & Destructor Documentation

7.5.1.1 TFT_eSPI_TouchUI()

```
TFT_eSPI_TouchUI::TFT_eSPI_TouchUI (
            void )
```

Constructor for the **TFT_eSPI_TouchUI** (p. 53) class.

7.5.2 Member Function Documentation

7.5.2.1 contains()

```
boolean TFT_eSPI_TouchUI::contains (
            int16_t x,
            int16_t y)
```

Check if the coordinates (x, y) are within the button.

Parameters

x	X-coordinate.
y	Y-coordinate.

Returns

true if the coordinates are within the button, false otherwise.

7.5.2.2 containsH()

```
boolean TFT_eSPI_TouchUI::containsH (
            int16_t x,
            int16_t y)
```

Check if the coordinates (x, y) are within the horizontal slider.

Parameters

x	X-coordinate.
y	Y-coordinate.

Returns

true if the coordinates are within the horizontal slider, false otherwise.

7.5.2.3 containsV()

```
boolean TFT_eSPI_TouchUI::containsV (
            int16_t x,
            int16_t y)
```

Check if the coordinates (x, y) are within the vertical slider.

Parameters

x	X-coordinate.
y	Y-coordinate.

Returns

true if the coordinates are within the vertical slider, false otherwise.

7.5.2.4 drawButton()

```
void TFT_eSPI_TouchUI::drawButton (
            boolean inverted = false)
```

Draw the button on the screen.

Parameters

inverted	Indicates if the button should be drawn inverted.

7.5.2.5 drawButtonG()

```
void TFT_eSPI_TouchUI::drawButtonG (
            boolean inverted = false)
```

Draw the graphic button on the screen.

Parameters

inverted	Indicates if the button should be drawn inverted.

7.5.2.6 drawButtonS()

```
void TFT_eSPI_TouchUI::drawButtonS (
            boolean inverted = false)
```

Draw the slider button on the screen.

Parameters

inverted	Indicates if the button should be drawn inverted.

7.5.2.7 drawSliderH()

```
void TFT_eSPI_TouchUI::drawSliderH (
            int16_t x)
```

Draw the horizontal slider on the screen.

Parameters

x	X-coordinate.

7.5.2.8 drawSliderV()

```
void TFT_eSPI_TouchUI::drawSliderV (
            int16_t x)
```

Draw the vertical slider on the screen.

Parameters

x	X-coordinate.

7.5.2.9 getValueH()

```
int16_t TFT_eSPI_TouchUI::getValueH (
            int16_t v)
```

Get the value of the horizontal slider.

Parameters

v	Slider value.

Returns

Value of the horizontal slider.

7.5.2.10 getValueV()

```
int16_t TFT_eSPI_TouchUI::getValueV (
            int16_t v)
```

Get the value of the vertical slider.

Parameters

v	Slider value.

Returns

Value of the vertical slider.

7.5.2.11 initButton()

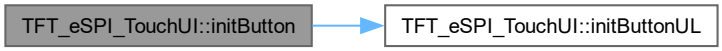
```
void TFT_eSPI_TouchUI::initButton (
            TFT_eSPI * gfx,
            int16_t x,
            int16_t y,
            uint16_t w,
            uint16_t h,
            uint16_t outline,
            uint16_t oncolor,
            uint16_t offcolor,
            char * label,
            uint8_t knob)
```

Initialize the button using the center and size.

Parameters

gfx	Pointer to the TFT_eSPI object.
x	X-coordinate of the button center.
y	Y-coordinate of the button center.
w	Width of the button.
h	Height of the button.
outline	Outline color of the button.
oncolor	Color of the button when on.
offcolor	Color of the button when off.
label	Button label.
knob	Button style.

Here is the call graph for this function:



7.5.2.12 initButtonG()

```
void TFT_eSPI_TouchUI::initButtonG (
             TFT_eSPI * gfx,
             int16_t x,
             int16_t y,
             uint16_t w,
             uint16_t h,
             uint16_t oncolor,
             uint16_t offcolor,
             const unsigned char * image,
             uint8_t knob)
```

Initialize the graphic button using the center and size.

Parameters

gfx	Pointer to the TFT_eSPI object.
x	X-coordinate of the button center.
y	Y-coordinate of the button center.
w	Width of the button.
h	Height of the button.
oncolor	Color of the button when on.
offcolor	Color of the button when off.
image	Image of the graphic button.
knob	Button style.

Here is the call graph for this function:

7.5.2.13 initButtonGUL()

```
void TFT_eSPI_TouchUI::initButtonGUL (
            TFT_eSPI * gfx,
            int16_t x,
            int16_t y,
            uint16_t w,
            uint16_t h,
            uint16_t oncolor,
            uint16_t offcolor,
            const unsigned char * image,
            uint8_t knob)
```

Initialize the graphic button using the top-left corner and size.

Parameters

gfx	Pointer to the TFT_eSPI object.
x	X-coordinate of the top-left corner of the button.
y	Y-coordinate of the top-left corner of the button.
w	Width of the button.
h	Height of the button.
oncolor	Color of the button when on.
offcolor	Color of the button when off.
image	Image of the graphic button.
knob	Button style.

Here is the caller graph for this function:

7.5.2.14 initButtonS()

```
void TFT_eSPI_TouchUI::initButtonS (
            TFT_eSPI * gfx,
            int16_t x,
            int16_t y,
            uint16_t w,
            uint16_t l,
            uint16_t outline,
            uint16_t oncolor,
            uint16_t offcolor,
            uint8_t knob,
            boolean lb)
```

Initialize the slider button.

Parameters

gfx	Pointer to the TFT_eSPI object.
x	X-coordinate of the center of the button.
y	Y-coordinate of the center of the button.
w	Width of the button.
l	Length of the button.
outline	Outline color of the button.
oncolor	Color of the button when on.
offcolor	Color of the button when off.
knob	Button style.
lb	Indicates if the button has a label.

7.5.2.15 initButtonUL()

```
void TFT_eSPI_TouchUI::initButtonUL (
            TFT_eSPI * gfx,
            int16_t x,
            int16_t y,
            uint16_t w,
            uint16_t h,
            uint16_t outline,
            uint16_t oncolor,
            uint16_t offcolor,
            char * label,
            uint8_t knob)
```

Initialize the button using the top-left corner and size.

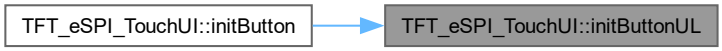
Parameters

gfx	Pointer to the TFT_eSPI object.
x	X-coordinate of the top-left corner of the button.
y	Y-coordinate of the top-left corner of the button.
w	Width of the button.
h	Height of the button.
outline	Outline color of the button.

Parameters

oncolor	Color of the button when on.
offcolor	Color of the button when off.
label	Button label.
knob	Button style.

Here is the caller graph for this function:



7.5.2.16 initSliderH()

```
void TFT_eSPI_TouchUI::initSliderH (
            TFT_eSPI * gfx,
            int16_t mim1,
            int16_t max1,
            int16_t x,
            int16_t y,
            uint16_t w,
            uint16_t l,
            uint16_t outline,
            uint16_t fill,
            uint16_t nofill,
            uint8_t knob,
            int8_t dv)
```

Initialize the horizontal slider.

Parameters

gfx	Pointer to the TFT_eSPI object.
mim1	Minimum value of the slider.
max1	Maximum value of the slider.
x	X-coordinate of the center of the slider.
y	Y-coordinate of the center of the slider.
w	Width of the slider.
l	Length of the slider.
outline	Outline color of the slider.
fill	Fill color of the slider.
nofill	Background color of the slider.
knob	Slider style.
dv	Slider value.

7.5.2.17 initSliderV()

```
void TFT_eSPI_TouchUI::initSliderV (
            TFT_eSPI * gfx,
            int16_t mim1,
            int16_t max1,
            int16_t x,
            int16_t y,
            uint16_t w,
            uint16_t l,
            uint16_t outline,
            uint16_t fill,
            uint16_t nofill,
            uint8_t knob,
            int8_t dv)
```

Initialize the vertical slider.

Parameters

gfx	Pointer to the TFT_eSPI object.
mim1	Minimum value of the slider.
max1	Maximum value of the slider.
x	X-coordinate of the center of the slider.
y	Y-coordinate of the center of the slider.
w	Width of the slider.
l	Length of the slider.
outline	Outline color of the slider.
fill	Fill color of the slider.
nofill	Background color of the slider.
knob	Slider style.
dv	Slider value.

7.5.2.18 isPressed()

```
boolean TFT_eSPI_TouchUI::isPressed ()
```

Check if the button is pressed.

Returns

true if the button is pressed, false otherwise.

7.5.2.19 justPressed()

```
boolean TFT_eSPI_TouchUI::justPressed ()
```

Check if the button was just pressed.

Returns

true if the button was just pressed, false otherwise.

7.5.2.20 justPressedOff()

```
boolean TFT_eSPI_TouchUI::justPressedOff (
            int16_t x)
```

Check if the slider button was just released at the x position.

Parameters

x	X-coordinate.

Returns

true if the slider button was just released at the x position, false otherwise.

7.5.2.21 justPressedOn()

```
boolean TFT_eSPI_TouchUI::justPressedOn (
            int16_t x)
```

Check if the slider button was just pressed at the x position.

Parameters

x	X-coordinate.

Returns

true if the slider button was just pressed at the x position, false otherwise.

7.5.2.22 justReleased()

```
boolean TFT_eSPI_TouchUI::justReleased ()
```

Check if the button was just released.

Returns

true if the button was just released, false otherwise.

7.5.2.23 press()

```
void TFT_eSPI_TouchUI::press (
            boolean p)
```

Set the button's pressed state.

Parameters

p	Pressed state of the button.

7.5.3 Member Data Documentation

7.5.3.1 _gfx

```
TFT_eSPI* TFT_eSPI_TouchUI::_gfx  [private]
```

7.5.3.2 _x

```
int16_t TFT_eSPI_TouchUI::_x  [private]
```

7.5.3.3 _y

```
int16_t TFT_eSPI_TouchUI::_y  [private]
```

The documentation for this class was generated from the following files:

- src/GUI/button/ **ButtonMod.h**
- src/GUI/button/ **ButtonMod.cpp**

7.6 touch_pos Struct Reference

```
#include <GUI.h>
```

Public Attributes

- bool **pressed** = false
- uint16_t **x** = 0
- uint16_t **y** = 0

7.6.1 Member Data Documentation

7.6.1.1 pressed

```
bool touch_pos::pressed = false
```

7.6.1.2 x

```
uint16_t touch_pos::x = 0
```

7.6.1.3 y

```
uint16_t touch_pos::y = 0
```

The documentation for this struct was generated from the following file:

- src/GUI/ **GUI.h**

Chapter 8

File Documentation

8.1 src/BMS/BMS.cpp File Reference

Functions

- uint8_t **monitorBattery** ()

  Calculates the battery percentage based on power consumption and maximum capacity.

Variables

- float **Rshunt_ohms** = 0.2294

  Calibration parameters for Adafruit INA219.
- uint32_t **cal_value** = 1785.527462946817785527462946817
- float **power_capacity_Wh** = 5.55

  Battery capacity in Wh.
- uint8_t **batt_percentage**
- float **max_voltage** = 4.2
- float **okay_voltage** = 3.9
- float **med_voltage** = 3.6
- float **low_voltage** = 3.3

8.1.1 Function Documentation

8.1.1.1 monitorBattery()

```
uint8_t monitorBattery ()
```

Calculates the battery percentage based on power consumption and maximum capacity.

Parameters

power_consumption_Wh	Power consumption in watt-hours.
max_capacity_battery	Maximum capacity of the battery in watt-hours.

Returns

uint8_t Battery percentage (0-100).

8.1.2 Variable Documentation

8.1.2.1 batt_percentage

```
uint8_t batt_percentage
```

8.1.2.2 cal_value

```
uint32_t cal_value = 1785.527462946817785527462946817785
```

8.1.2.3 low_voltage

```
float low_voltage = 3.3
```

8.1.2.4 max_voltage

```
float max_voltage = 4.2
```

8.1.2.5 med_voltage

```
float med_voltage = 3.6
```

8.1.2.6 okay_voltage

```
float okay_voltage = 3.9
```

8.1.2.7 power_capacity_Wh

```
float power_capacity_Wh = 5.55
```

Battery capacity in Wh.

This variable defines the battery capacity in watt-hours.

8.1.2.8 Rshunt_ohms

```
float Rshunt_ohms = 0.2294
```

Calibration parameters for Adafruit INA219.

Calibration values change for each calibration mode and the selected Rshunt. This library has been modified to calculate the calibration value using the following expression:

$cal\_value = \frac{0.04096}{CurrentLSB \times Rshunt}$

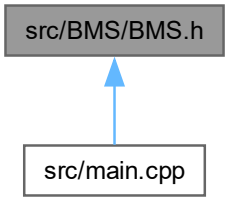
Where CurrentLSB is:

- 100 uA/bit for setCalibration_32V_2A (0.0001)
- 40 uA/bit for setCalibration_32V_1A (0.00004)
- 50 uA/bit for setCalibration_16V_400mA (0.00005)

In this case, setCalibration_32V_2A with an Rshunt of 0.2294 ohms is used: $cal\_value = \frac{0.04096}{0.0001 \times 0.2294} = 1785.527462946817785527462946817785$

8.2 src/BMS/BMS.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define **baud_rate** 115200

  Serial port communication speed.

Variables

- float **Rshunt_ohms**

  Calibration parameters for Adafruit INA219.
- uint32_t **cal_value**
- float **power_capacity_Wh**

  Battery capacity in Wh.
- uint8_t **percentage**

  Last percentage.
- float **max_voltage**
- float **okay_voltage**
- float **med_voltage**
- float **low_voltage**

8.2.1 Macro Definition Documentation

8.2.1.1 baud_rate

```
#define baud_rate 115200
```

Serial port communication speed.

This variable defines the baud rate for serial port communication.

8.2.2 Variable Documentation

8.2.2.1 cal_value

```
uint32_t cal_value  [extern]
```

8.2.2.2 low_voltage

```
float low_voltage  [extern]
```

8.2.2.3 max_voltage

```
float max_voltage  [extern]
```

8.2.2.4 med_voltage

```
float med_voltage  [extern]
```

8.2.2.5 okay_voltage

```
float okay_voltage  [extern]
```

8.2.2.6 percentage

```
uint8_t percentage  [extern]
```

Last percentage.

This variable contain the last percentage measured in case of swith the device

8.2.2.7 power_capacity_Wh

```
float power_capacity_Wh  [extern]
```

Battery capacity in Wh.

This variable defines the battery capacity in watt-hours.

8.2.2.8 Rshunt_ohms

```
float Rshunt_ohms  [extern]
```

Calibration parameters for Adafruit INA219.

Calibration values change for each calibration mode and the selected Rshunt. This library has been modified to calculate the calibration value using the following expression:

$cal\_value = \frac{0.04096}{CurrentLSB \times Rshunt}$

Where CurrentLSB is:

- 100 uA/bit for setCalibration_32V_2A (0.0001)
- 40 uA/bit for setCalibration_32V_1A (0.00004)
- 50 uA/bit for setCalibration_16V_400mA (0.00005)

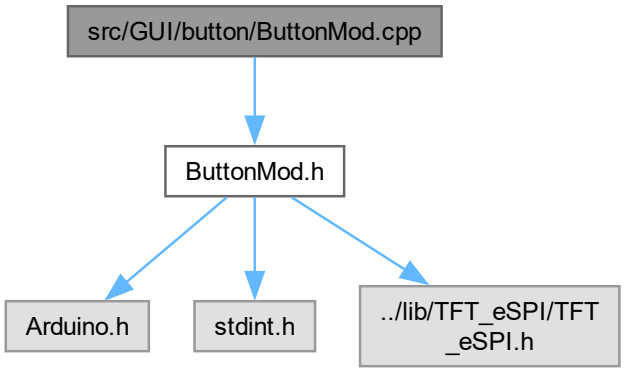
In this case, setCalibration_32V_2A with an Rshunt of 0.2294 ohms is used: $cal\_value = \frac{0.04096}{0.0001 \times 0.2294} = 1785.52746294681778552746294681 78$

8.3 src/GUI/button/ButtonMod.cpp File Reference

Implementation of the modified button functionality.

```
#include "ButtonMod.h"
```

Include dependency graph for ButtonMod.cpp:



8.3.1 Detailed Description

Implementation of the modified button functionality.

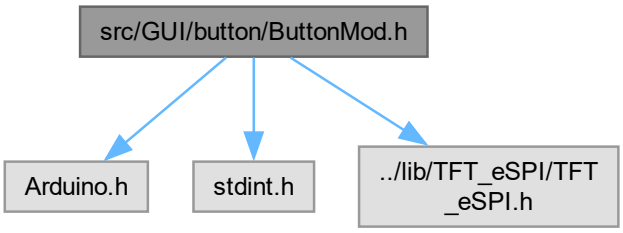
This file implements the functions for the modified button class that extends the TFT_eSPI library to handle square buttons.

8.4 src/GUI/button/ButtonMod.h File Reference

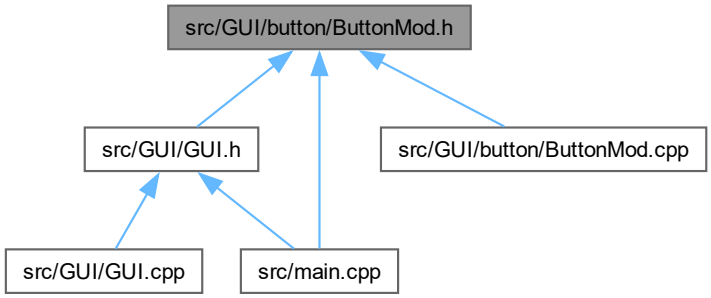
ButtonMod.h (p. 72) modified for this firmware.

```
#include <Arduino.h>
#include <stdint.h>
#include "../lib/TFT_eSPI/TFT_eSPI.h"
```

Include dependency graph for ButtonMod.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **TFT_eSPI_Button_Mod**

  Class to handle buttons modified from the TFT_eSPI library.
- class **TFT_eSPI_TouchUI**

Macros

- #define **BUTTON_MOD_H**

8.4.1 Detailed Description

ButtonMod.h (p. 72) modified for this firmware.

The following button class has been ported over from the Adafruit_GFX library so should be compatible. A slightly different implementation in this TFT_eSPI library allows the button legends to be in any font, allow longer labels and to adjust text positioning within button

8.4.2 Libraries

The necessary libraries for the program to function correctly are included.

- Arduino.h: Library to manage Arduino projects.
- stdint.h: Library that defines integer types, limits of specified width integer types, limits of other integer types and macros for integer constant expressions.
- TFT_eSPI.h: Library hardware specific for our display.

8.4.3 Macro Definition Documentation

8.4.3.1 BUTTON_MOD_H

```
#define BUTTON_MOD_H
```

8.5 src/GUI/GUI.cpp File Reference

```
#include "SYSTEM/pins.h"
#include "GUI.h"
#include <TFT_eSPI.h>
#include <SD.h>
#include <SPI.h>
#include <Arduino.h>
#include <time.h>
#include <iostream>
#include "../src/GUI/sprites/BRIGHTNESS/empty_sun.h"
#include "../src/GUI/sprites/BRIGHTNESS/full_sun.h"
```

Include dependency graph for GUI.cpp:

Macros

- #define **pos_measure** 4
- #define **pos_settings** 3
- #define **pos_help** 2
- #define **pos_shutdown** 1
- #define **max_value_pos** 5
- #define **min_value_pos** 0

Variables

- File **myFile**
- int **current_pos** = 4
- int **actualState**
- int **lastState**
- const uint8_t * **wifi_icon_sprite** = **WIFI_GOOD_ICON_SPRITE**

  Returns whether the display is available or not.
- const uint8_t * **bat_level_icon_sprite** = **BAT_LEVEL_HIGH_ICON_SPRITE**

8.5.1 Detailed Description

Version

0.1

Date

2022-04-15

Copyright

8.5.2 Macro Definition Documentation

8.5.2.1 max_value_pos

```
#define max_value_pos 5
```

8.5.2.2 min_value_pos

```
#define min_value_pos 0
```

8.5.2.3 pos_help

```
#define pos_help 2
```

8.5.2.4 pos_measure

```
#define pos_measure 4
```

8.5.2.5 pos_settings

```
#define pos_settings 3
```

8.5.2.6 pos_shutdown

```
#define pos_shutdown 1
```

8.5.3 Variable Documentation

8.5.3.1 actualState

```
int actualState
```

8.5.3.2 bat_level_icon_sprite

```
const uint8_t* bat_level_icon_sprite =  BAT_LEVEL_HIGH_ICON_SPRITE
```

8.5.3.3 current_pos

```
int current_pos = 4
```

8.5.3.4 lastState

```
int lastState
```

8.5.3.5 myFile

```
File myFile
```

8.5.3.6 wifi_icon_sprite

```
const uint8_t* wifi_icon_sprite =  WIFI_GOOD_ICON_SPRITE
```

Returns whether the display is available or not.

Returns

true cannot use the screen, being used by an internal function

false can use the screen without problems

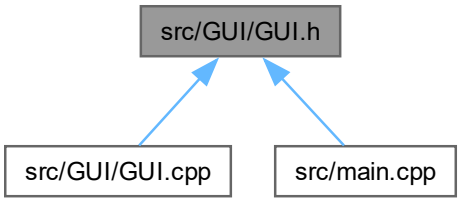
8.6 src/GUI/GUI.h File Reference

Header file for the **GUI** (p. 19) class.

```
#include <Arduino.h>
#include <stdint.h>
#include <freertos/FreeRTOS.h>
#include "FS.h"
#include <SPI.h>
#include "../lib/TFT_eSPI/TFT_eSPI.h"
#include "SYSTEM/SystemStatus.h"
#include "SYSTEM/version.h"
#include "button/ButtonMod.h"
#include "SYSTEM/pins.h"
#include "../src/BMS/BMS.h"
#include "sprites/WIFI/WIFI_SPRITES.c"
#include "sprites/BT/BT_ICON_SPRITES.c"
#include "sprites/BAT/BAT_CH_ICON_SPRITE.c"
#include "sprites/BAT/BAT_LEVELS_SPRITES.c"
#include "sprites/HOME/HOME.c"
```

Include dependency graph for GUI.h:

This graph shows which files directly or indirectly include this file:



Classes

- struct **touch_pos**
- class **GUI**

  GUI *(p. 19) class manages the graphical user interface for a TFT display.*

Macros

- #define **SCREEN_X_PIXELS** 240
- #define **SCREEN_Y_PIXELS** 320
- #define **SCREEN_ROTATION** 0
- #define **CALIBRATION_FILE** "/TouchCalData3"
- #define **REPEAT_CAL** false
- #define **WIFI_ICON_X_POS** 120
- #define **WIFI_ICON_Y_POS** 2
- #define **BT_ICON_X_POS** 142
- #define **BT_ICON_Y_POS** 3
- #define **BAT_CH_ICON_X_POS** 157
- #define **BAT_CH_ICON_Y_POS** 2
- #define **BAT_LEVEL_ICON_X_POS** 205
- #define **BAT_LEVEL_ICON_Y_POS** 4
- #define **HOME_GRANASAT_X_POS** **SCREEN_X_PIXELS**/2
- #define **HOME_GRANASAT_Y_POS** **SCREEN_Y_PIXELS**/2
- #define **STATUSBAR_HEIGHT** 20
- #define **TITLE_RIGHT_MARGIN** 20
- #define **TITLE_RIGHT_X_POS** **SCREEN_X_PIXELS** - **TITLE_RIGHT_MARGIN**
- #define **TITLE_CENTER_X_POS** **SCREEN_X_PIXELS**/2
- #define **TITLE_CENTER_Y_POS** 30
- #define **BUTTON_CENTER_WIDTH** 220
- #define **BUTTON_CENTER_HEIGHT** 40
- #define **BUTTON_CENTER_MARGIN** 10
- #define **BUTTON_CENTER_X_POS** **SCREEN_X_PIXELS**/2
- #define **BUTTON_CENTER_Y_POS_0** 120 - **BUTTON_CENTER_HEIGHT**/2 - **BUTTON_CENTER_↩ MARGIN**
- #define **BUTTON_CENTER_Y_POS_1** 120 + **BUTTON_CENTER_HEIGHT**/2
- #define **BUTTON_CENTER_Y_POS_2** **BUTTON_CENTER_Y_POS_1** + **BUTTON_CENTER_HEIGHT** + **BUTTON_CENTER_MARGIN**
- #define **BUTTON_CENTER_Y_POS_3** **BUTTON_CENTER_Y_POS_2** + **BUTTON_CENTER_HEIGHT** + **BUTTON_CENTER_MARGIN**
- #define **BUTTON_CENTER_Y_POS_4** **BUTTON_CENTER_Y_POS_3** + **BUTTON_CENTER_HEIGHT** + **BUTTON_CENTER_MARGIN**
- #define **BUTTON_BACK_WIDTH** 80
- #define **BUTTON_BACK_HEIGHT** 30
- #define **BUTTON_BACK_X_POS** **BUTTON_BACK_WIDTH**/2
- #define **BUTTON_BACK_Y_POS** **SCREEN_Y_PIXELS** - **BUTTON_BACK_HEIGHT**/2
- #define **BUTTON_CONNECTIVITY_WIDTH** 100
- #define **BUTTON_CONNECTIVITY_HEIGHT** 40
- #define **BUTTON_CONNECTIVITY_X_POS1** **SCREEN_X_PIXELS**/4
- #define **BUTTON_CONNECTIVITY_X_POS2** 3∗ **SCREEN_X_PIXELS**/4
- #define **BUTTON_START_WIDTH** 80
- #define **BUTTON_START_HEIGHT** 30
- #define **BUTTON_START_X_POS** **BUTTON_START_WIDTH**∗2 + **BUTTON_START_WIDTH**/2
- #define **BUTTON_START_Y_POS** **SCREEN_Y_PIXELS** - **BUTTON_START_HEIGHT**/2
- #define **BUTTON_NUMBER_WIDTH** 24
- #define **BUTTON_NUMBER_HEIGHT** 30
- #define **BUTTON_ZERO_X_POS** **BUTTON_NUMBER_WIDTH**/2
- #define **BUTTON_ONE_X_POS** **BUTTON_NUMBER_WIDTH**/2 + **BUTTON_NUMBER_WIDTH**
- #define **BUTTON_TWO_X_POS** **BUTTON_NUMBER_WIDTH**/2 + 2 ∗ **BUTTON_NUMBER_WIDTH**
- #define **BUTTON_THREE_X_POS** **BUTTON_NUMBER_WIDTH**/2 + 3 ∗ **BUTTON_NUMBER_WIDTH**
- #define **BUTTON_FOUR_X_POS** **BUTTON_NUMBER_WIDTH**/2 + 4 ∗ **BUTTON_NUMBER_WIDTH**
- #define **BUTTON_FIVE_X_POS** **BUTTON_NUMBER_WIDTH**/2 + 5 ∗ **BUTTON_NUMBER_WIDTH**

- #define **BUTTON_SIX_X_POS** **BUTTON_NUMBER_WIDTH**/2 + 6 ∗ **BUTTON_NUMBER_WIDTH**
- #define **BUTTON_SEVEN_X_POS** **BUTTON_NUMBER_WIDTH**/2 + 7 ∗ **BUTTON_NUMBER_WIDTH**
- #define **BUTTON_EIGHT_X_POS** **BUTTON_NUMBER_WIDTH**/2 + 8 ∗ **BUTTON_NUMBER_WIDTH**
- #define **BUTTON_NINE_X_POS** **BUTTON_NUMBER_WIDTH**/2 + 9 ∗ **BUTTON_NUMBER_WIDTH**
- #define **BUTTON_NUMBER_Y_POS** **SCREEN_Y_PIXELS** - 3∗ **BUTTON_NUMBER_HEIGHT**/2
- #define **SLIDER_WIDTH** 200
- #define **SLIDER_HEIGHT** 30
- #define **SLIDER_X_POS** 20
- #define **SLIDER_Y_POS** **SCREEN_Y_PIXELS** / 2
- #define **SLIDER_RADIUS** 5
- #define **MEAS_HEIGHT** 30
- #define **MEAS_READY_WIDTH** 30
- #define **MEAS_READY_HEIGHT** 15
- #define **MEAS_READY_X_POS** 10+ **MEAS_READY_WIDTH**/2
- #define **MEAS_READY_Y_POS** **STATUSBAR_HEIGHT**+ **MEAS_HEIGHT**/2
- #define **GRAPH_X_TICKS** 9
- #define **GRAPH_Y_TICKS** 7
- #define **GRAPH_TICKS_LENGTH** 6
- #define **GRAPH_TOP_MARGIN** 10
- #define **GRAPH_AREA_WIDTH** **SCREEN_X_PIXELS**
- #define **GRAPH_AREA_HEIGHT** 120
- #define **GRAPH_GRID_WIDTH** 176
- #define **GRAPH_GRID_SIDE_MARGIN** (**GRAPH_AREA_WIDTH**- **GRAPH_GRID_WIDTH**)/2
- #define **GRAPH_GRID_HEIGHT** 84
- #define **GRAPH_GRID_BOTTOM_MARGIN** **GRAPH_AREA_HEIGHT**- **GRAPH_TOP_MARGIN**- **GRAPH_GRID_HEIGHT**
- #define **GRAPH_LINE_X_SEP** **GRAPH_GRID_WIDTH**/(**GRAPH_X_TICKS**-1)
- #define **GRAPH_LINE_Y_SEP** **GRAPH_GRID_HEIGHT**/(**GRAPH_Y_TICKS**-1)
- #define **GRAPH_FIRST_POS** 50
- #define **GRAPH_SECOND_POS** **GRAPH_FIRST_POS**+ **GRAPH_AREA_HEIGHT**
- #define **DIAGRAMS_GUI_H**

Enumerations

- enum **screens** {
  MAIN , **CONNECTIVITY** , **MEASURE** , **MEASURE_HISTORIC** ,
  BATTERY , **SETTINGS** , **HELP** , **SHUTDOWN_CONFIRM** ,
  MEMORY , **ENGINEER_MODE** , **DISPLAY_** }

Variables

- const uint32_t **LOOP_TICKS** = 50 / portTICK_PERIOD_MS

8.6.1 Detailed Description

Header file for the **GUI** (p. 19) class.

This file contains the declaration of the **GUI** (p. 19) class, which manages the graphical user interface for a TFT display.

8.6.2 Macro Definition Documentation

8.6.2.1 BAT_CH_ICON_X_POS

```
#define BAT_CH_ICON_X_POS 157
```

8.6.2.2 BAT_CH_ICON_Y_POS

```
#define BAT_CH_ICON_Y_POS 2
```

8.6.2.3 BAT_LEVEL_ICON_X_POS

```
#define BAT_LEVEL_ICON_X_POS 205
```

8.6.2.4 BAT_LEVEL_ICON_Y_POS

```
#define BAT_LEVEL_ICON_Y_POS 4
```

8.6.2.5 BT_ICON_X_POS

```
#define BT_ICON_X_POS 142
```

8.6.2.6 BT_ICON_Y_POS

```
#define BT_ICON_Y_POS 3
```

8.6.2.7 BUTTON_BACK_HEIGHT

```
#define BUTTON_BACK_HEIGHT 30
```

8.6.2.8 BUTTON_BACK_WIDTH

```
#define BUTTON_BACK_WIDTH 80
```

8.6.2.9 BUTTON_BACK_X_POS

```
#define BUTTON_BACK_X_POS  BUTTON_BACK_WIDTH/2
```

8.6.2.10 BUTTON_BACK_Y_POS

```
#define BUTTON_BACK_Y_POS  SCREEN_Y_PIXELS - BUTTON_BACK_HEIGHT/2
```

8.6.2.11 BUTTON_CENTER_HEIGHT

```
#define BUTTON_CENTER_HEIGHT 40
```

8.6.2.12 BUTTON_CENTER_MARGIN

```
#define BUTTON_CENTER_MARGIN 10
```

8.6.2.13 BUTTON_CENTER_WIDTH

```
#define BUTTON_CENTER_WIDTH 220
```

8.6.2.14 BUTTON_CENTER_X_POS

```
#define BUTTON_CENTER_X_POS  SCREEN_X_PIXELS/2
```

8.6.2.15 BUTTON_CENTER_Y_POS_0

```
#define BUTTON_CENTER_Y_POS_0 120 -  BUTTON_CENTER_HEIGHT/2 -  BUTTON_CENTER_MARGIN
```

8.6.2.16 BUTTON_CENTER_Y_POS_1

```
#define BUTTON_CENTER_Y_POS_1 120 +  BUTTON_CENTER_HEIGHT/2
```

8.6.2.17 BUTTON_CENTER_Y_POS_2

```
#define BUTTON_CENTER_Y_POS_2  BUTTON_CENTER_Y_POS_1 +  BUTTON_CENTER_HEIGHT +  BUTTON_↩
CENTER_MARGIN
```

8.6.2.18 BUTTON_CENTER_Y_POS_3

```
#define BUTTON_CENTER_Y_POS_3  BUTTON_CENTER_Y_POS_2 +  BUTTON_CENTER_HEIGHT +  BUTTON_↩
CENTER_MARGIN
```

8.6.2.19 BUTTON_CENTER_Y_POS_4

```
#define BUTTON_CENTER_Y_POS_4  BUTTON_CENTER_Y_POS_3 +  BUTTON_CENTER_HEIGHT +  BUTTON_↩
CENTER_MARGIN
```

8.6.2.20 BUTTON_CONNECTIVITY_HEIGHT

```
#define BUTTON_CONNECTIVITY_HEIGHT 40
```

8.6.2.21 BUTTON_CONNECTIVITY_WIDTH

```
#define BUTTON_CONNECTIVITY_WIDTH 100
```

8.6.2.22 BUTTON_CONNECTIVITY_X_POS1

```
#define BUTTON_CONNECTIVITY_X_POS1  SCREEN_X_PIXELS/4
```

8.6.2.23 BUTTON_CONNECTIVITY_X_POS2

```
#define BUTTON_CONNECTIVITY_X_POS2 3* SCREEN_X_PIXELS/4
```

8.6.2.24 BUTTON_EIGHT_X_POS

```
#define BUTTON_EIGHT_X_POS  BUTTON_NUMBER_WIDTH/2 + 8 *  BUTTON_NUMBER_WIDTH
```

8.6.2.25 BUTTON_FIVE_X_POS

```
#define BUTTON_FIVE_X_POS  BUTTON_NUMBER_WIDTH/2 + 5 *  BUTTON_NUMBER_WIDTH
```

8.6.2.26 BUTTON_FOUR_X_POS

```
#define BUTTON_FOUR_X_POS  BUTTON_NUMBER_WIDTH/2 + 4 *  BUTTON_NUMBER_WIDTH
```

8.6.2.27 BUTTON_NINE_X_POS

```
#define BUTTON_NINE_X_POS  BUTTON_NUMBER_WIDTH/2 + 9 *  BUTTON_NUMBER_WIDTH
```

8.6.2.28 BUTTON_NUMBER_HEIGHT

```
#define BUTTON_NUMBER_HEIGHT 30
```

8.6.2.29 BUTTON_NUMBER_WIDTH

```
#define BUTTON_NUMBER_WIDTH 24
```

8.6.2.30 BUTTON_NUMBER_Y_POS

```
#define BUTTON_NUMBER_Y_POS  SCREEN_Y_PIXELS - 3* BUTTON_NUMBER_HEIGHT/2
```

8.6.2.31 BUTTON_ONE_X_POS

```
#define BUTTON_ONE_X_POS  BUTTON_NUMBER_WIDTH/2 +  BUTTON_NUMBER_WIDTH
```

8.6.2.32 BUTTON_SEVEN_X_POS

```
#define BUTTON_SEVEN_X_POS  BUTTON_NUMBER_WIDTH/2 + 7 *  BUTTON_NUMBER_WIDTH
```

8.6.2.33 BUTTON_SIX_X_POS

```
#define BUTTON_SIX_X_POS  BUTTON_NUMBER_WIDTH/2 + 6 *  BUTTON_NUMBER_WIDTH
```

8.6.2.34 BUTTON_START_HEIGHT

```
#define BUTTON_START_HEIGHT 30
```

8.6.2.35 BUTTON_START_WIDTH

```
#define BUTTON_START_WIDTH 80
```

8.6.2.36 BUTTON_START_X_POS

```
#define BUTTON_START_X_POS  BUTTON_START_WIDTH*2 +  BUTTON_START_WIDTH/2
```

8.6.2.37 BUTTON_START_Y_POS

```
#define BUTTON_START_Y_POS  SCREEN_Y_PIXELS -  BUTTON_START_HEIGHT/2
```

8.6.2.38 BUTTON_THREE_X_POS

```
#define BUTTON_THREE_X_POS  BUTTON_NUMBER_WIDTH/2 + 3 *  BUTTON_NUMBER_WIDTH
```

8.6.2.39 BUTTON_TWO_X_POS

```
#define BUTTON_TWO_X_POS  BUTTON_NUMBER_WIDTH/2 + 2 *  BUTTON_NUMBER_WIDTH
```

8.6.2.40 BUTTON_ZERO_X_POS

```
#define BUTTON_ZERO_X_POS  BUTTON_NUMBER_WIDTH/2
```

8.6.2.41 CALIBRATION_FILE

```
#define CALIBRATION_FILE "/TouchCalData3"
```

8.6.2.42 DIAGRAMS_GUI_H

```
#define DIAGRAMS_GUI_H
```

8.6.2.43 GRAPH_AREA_HEIGHT

```
#define GRAPH_AREA_HEIGHT 120
```

8.6.2.44 GRAPH_AREA_WIDTH

```
#define GRAPH_AREA_WIDTH  SCREEN_X_PIXELS
```

8.6.2.45 GRAPH_FIRST_POS

```
#define GRAPH_FIRST_POS 50
```

8.6.2.46 GRAPH_GRID_BOTTOM_MARGIN

```
#define GRAPH_GRID_BOTTOM_MARGIN  GRAPH_AREA_HEIGHT- GRAPH_TOP_MARGIN- GRAPH_GRID_HEIGHT
```

8.6.2.47 GRAPH_GRID_HEIGHT

```
#define GRAPH_GRID_HEIGHT 84
```

8.6.2.48 GRAPH_GRID_SIDE_MARGIN

```
#define GRAPH_GRID_SIDE_MARGIN ( GRAPH_AREA_WIDTH- GRAPH_GRID_WIDTH)/2
```

8.6.2.49 GRAPH_GRID_WIDTH

```
#define GRAPH_GRID_WIDTH 176
```

8.6.2.50 GRAPH_LINE_X_SEP

```
#define GRAPH_LINE_X_SEP  GRAPH_GRID_WIDTH/( GRAPH_X_TICKS-1)
```

8.6.2.51 GRAPH_LINE_Y_SEP

```
#define GRAPH_LINE_Y_SEP  GRAPH_GRID_HEIGHT/( GRAPH_Y_TICKS-1)
```

8.6.2.52 GRAPH_SECOND_POS

```
#define GRAPH_SECOND_POS  GRAPH_FIRST_POS+ GRAPH_AREA_HEIGHT
```

8.6.2.53 GRAPH_TICKS_LENGTH

```
#define GRAPH_TICKS_LENGTH 6
```

8.6.2.54 GRAPH_TOP_MARGIN

```
#define GRAPH_TOP_MARGIN 10
```

8.6.2.55 GRAPH_X_TICKS

```
#define GRAPH_X_TICKS 9
```

8.6.2.56 GRAPH_Y_TICKS

```
#define GRAPH_Y_TICKS 7
```

8.6.2.57 HOME_GRANASAT_X_POS

```
#define HOME_GRANASAT_X_POS  SCREEN_X_PIXELS/2
```

8.6.2.58 HOME_GRANASAT_Y_POS

```
#define HOME_GRANASAT_Y_POS  SCREEN_Y_PIXELS/2
```

8.6.2.59 MEAS_HEIGHT

```
#define MEAS_HEIGHT 30
```

8.6.2.60 MEAS_READY_HEIGHT

```
#define MEAS_READY_HEIGHT 15
```

8.6.2.61 MEAS_READY_WIDTH

```
#define MEAS_READY_WIDTH 30
```

8.6.2.62 MEAS_READY_X_POS

```
#define MEAS_READY_X_POS 10+ MEAS_READY_WIDTH/2
```

8.6.2.63 MEAS_READY_Y_POS

```
#define MEAS_READY_Y_POS  STATUSBAR_HEIGHT+ MEAS_HEIGHT/2
```

8.6.2.64 REPEAT_CAL

```
#define REPEAT_CAL false
```

8.6.2.65 SCREEN_ROTATION

```
#define SCREEN_ROTATION 0
```

8.6.2.66 SCREEN_X_PIXELS

```
#define SCREEN_X_PIXELS 240
```

8.6.2.67 SCREEN_Y_PIXELS

```
#define SCREEN_Y_PIXELS 320
```

8.6.2.68 SLIDER_HEIGHT

```
#define SLIDER_HEIGHT 30
```

8.6.2.69 SLIDER_RADIUS

```
#define SLIDER_RADIUS 5
```

8.6.2.70 SLIDER_WIDTH

```
#define SLIDER_WIDTH 200
```

8.6.2.71 SLIDER_X_POS

```
#define SLIDER_X_POS 20
```

8.6.2.72 SLIDER_Y_POS

```
#define SLIDER_Y_POS  SCREEN_Y_PIXELS / 2
```

8.6.2.73 STATUSBAR_HEIGHT

```
#define STATUSBAR_HEIGHT 20
```

8.6.2.74 TITLE_CENTER_X_POS

```
#define TITLE_CENTER_X_POS  SCREEN_X_PIXELS/2
```

8.6.2.75 TITLE_CENTER_Y_POS

```
#define TITLE_CENTER_Y_POS 30
```

8.6.2.76 TITLE_RIGHT_MARGIN

```
#define TITLE_RIGHT_MARGIN 20
```

8.6.2.77 TITLE_RIGHT_X_POS

```
#define TITLE_RIGHT_X_POS  SCREEN_X_PIXELS - TITLE_RIGHT_MARGIN
```

8.6.2.78 WIFI_ICON_X_POS

```
#define WIFI_ICON_X_POS 120
```

8.6.2.79 WIFI_ICON_Y_POS

```
#define WIFI_ICON_Y_POS 2
```

8.6.3 Enumeration Type Documentation

8.6.3.1 screens

```
enum screens
```

Enumerator

MAIN	
CONNECTIVITY	
MEASURE	
MEASURE_HISTORIC	
BATTERY	
SETTINGS	
HELP	
SHUTDOWN_CONFIRM	
MEMORY	
ENGINEER_MODE	
DISPLAY_	

8.6.4 Variable Documentation

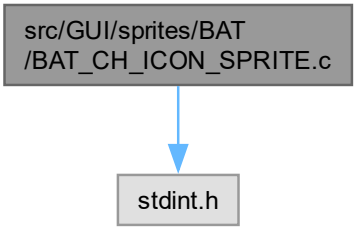
8.6.4.1 LOOP_TICKS

```
const uint32_t LOOP_TICKS = 50 / portTICK_PERIOD_MS
```

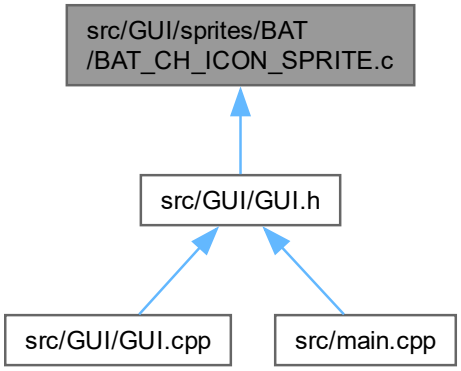
8.7 src/GUI/sprites/BAT/BAT_CH_ICON_SPRITE.c File Reference

```
#include <stdint.h>
```

Include dependency graph for BAT_CH_ICON_SPRITE.c:



This graph shows which files directly or indirectly include this file:



Macros

- #define **BAT_CH_ICON_WIDTH** 7
- #define **BAT_CH_ICON_HEIGHT** 16

Variables

- const uint8_t **BAT_CH_ICON_SPRITE** [ ]

8.7.1 Macro Definition Documentation

8.7.1.1 BAT_CH_ICON_HEIGHT

```
#define BAT_CH_ICON_HEIGHT 16
```

8.7.1.2 BAT_CH_ICON_WIDTH

```
#define BAT_CH_ICON_WIDTH 7
```

8.7.2 Variable Documentation

8.7.2.1 BAT_CH_ICON_SPRITE

```
const uint8_t BAT_CH_ICON_SPRITE[]
```

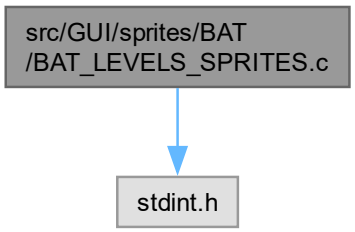
Initial value:

```
= {
     0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x6d,0xdb,0xdb,0x92,0x00,0x00
    ,0x00,0xdb,0xff,0xff,0x6d,0x00,0x00
    ,0x00,0xff,0xff,0xff,0x00,0x00,0x00
    ,0x24,0xff,0xff,0xdb,0xb6,0xdb,0x6d
    ,0x92,0xff,0xff,0xff,0xff,0xff,0x00
    ,0xb6,0xff,0xff,0xff,0xff,0x92,0x00
    ,0x6d,0x6d,0x00,0xff,0xff,0x00,0x00
    ,0x00,0x00,0x49,0xff,0xb6,0x00,0x00
    ,0x00,0x00,0x92,0xff,0x24,0x00,0x00
    ,0x00,0x00,0xdb,0xb6,0x00,0x00,0x00
    ,0x00,0x00,0xdb,0x49,0x00,0x00,0x00
    ,0x00,0x00,0x6d,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00
}
```

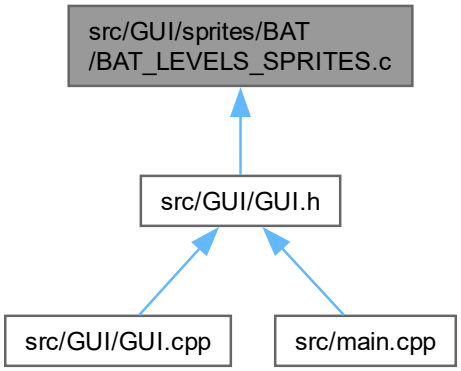
8.8 src/GUI/sprites/BAT/BAT_LEVELS_SPRITES.c File Reference

```
#include <stdint.h>
```

Include dependency graph for BAT_LEVELS_SPRITES.c:



This graph shows which files directly or indirectly include this file:



Macros

- #define **BAT_LEVEL_ICON_WIDTH** 25
- #define **BAT_LEVEL_ICON_HEIGHT** 12
- #define **BAT_LEVEL_HIGH** 75
- #define **BAT_LEVEL_OKAY** 50
- #define **BAT_LEVEL_MED** 25

Variables

- const uint8_t **BAT_LEVEL_HIGH_ICON_SPRITE** [ ]
- const uint8_t **BAT_LEVEL_OKAY_ICON_SPRITE** [ ]
- const uint8_t **BAT_LEVEL_MED_ICON_SPRITE** [ ]
- const uint8_t **BAT_LEVEL_LOW_ICON_SPRITE** [ ]

8.8.1 Macro Definition Documentation

8.8.1.1 BAT_LEVEL_HIGH

```
#define BAT_LEVEL_HIGH 75
```

8.8.1.2 BAT_LEVEL_ICON_HEIGHT

```
#define BAT_LEVEL_ICON_HEIGHT 12
```

8.8.1.3 BAT_LEVEL_ICON_WIDTH

```
#define BAT_LEVEL_ICON_WIDTH 25
```

8.8.1.4 BAT_LEVEL_MED

```
#define BAT_LEVEL_MED 25
```

8.8.1.5 BAT_LEVEL_OKAY

```
#define BAT_LEVEL_OKAY 50
```

8.8.2 Variable Documentation

8.8.2.1 BAT_LEVEL_HIGH_ICON_SPRITE

```
const uint8_t BAT_LEVEL_HIGH_ICON_SPRITE[]
```

Initial value:

```
= {
     0x00,0x00,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,(
    ,0x00,0x00,0xff,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,(
    ,0x00,0x00,0xff,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0xff,0xff,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0xff,0xff,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0x00,0x00,0xff,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0x00,0x00,0xff,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,(
    ,0x00,0x00,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,(
}
```

8.8.2.2 BAT_LEVEL_LOW_ICON_SPRITE

```
const uint8_t BAT_LEVEL_LOW_ICON_SPRITE[]
```

Initial value:

```
= {
     0x00,0x00,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,(
    ,0x00,0x00,0xff,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,(
    ,0x00,0x00,0xff,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0xe0,0xe0,0xe0,0xe0,0x00,(
    ,0xff,0xff,0xff,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0xe0,0xe0,0xe0,0xe0,0x00,(
    ,0xff,0x00,0xff,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0xe0,0xe0,0xe0,0xe0,0x00,(
    ,0xff,0x00,0xff,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0xe0,0xe0,0xe0,0xe0,0x00,(
    ,0xff,0x00,0xff,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0xe0,0xe0,0xe0,0xe0,0x00,(
    ,0xff,0x00,0xff,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0xe0,0xe0,0xe0,0xe0,0x00,(
    ,0xff,0xff,0xff,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0xe0,0xe0,0xe0,0xe0,0x00,(
    ,0x00,0x00,0xff,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0x40,0x40,0x40,0x40,0x00,0xe0,0xe0,0xe0,0xe0,0x00,(
    ,0x00,0x00,0xff,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,(
    ,0x00,0x00,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,(
}
```

8.8.2.3 BAT_LEVEL_MED_ICON_SPRITE

```
const uint8_t BAT_LEVEL_MED_ICON_SPRITE[]
```

Initial value:

```
= {
     0x00,0x00,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,(
    ,0x00,0x00,0xff,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,(
    ,0x00,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0xff,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0xff,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0x00,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0x00,0x00,0xff,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,(
    ,0x00,0x00,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,(
}
```

8.8.2.4 BAT_LEVEL_OKAY_ICON_SPRITE

```
const uint8_t BAT_LEVEL_OKAY_ICON_SPRITE[]
```

Initial value:

```
= {
     0x00,0x00,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,(
    ,0x00,0x00,0xff,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,(
    ,0x00,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0xff,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0xff,0xff,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0x00,0x00,0xff,0x00,0x08,0x08,0x08,0x08,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,0x18,0x18,0x18,0x18,0x00,(
    ,0x00,0x00,0xff,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,(
    ,0x00,0x00,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,(
}
```

8.9 src/GUI/sprites/BRIGHTNESS/empty_sun.h File Reference

Macros

- #define **LOGO_WIDTH** 30
- #define **LOGO_HEIGHT** 30

Variables

- const unsigned short empty_sun[2450] **PROGMEM**

8.9.1 Macro Definition Documentation

8.9.1.1 LOGO_HEIGHT

```
#define LOGO_HEIGHT 30
```

8.9.1.2 LOGO_WIDTH

```
#define LOGO_WIDTH 30
```

8.9.2 Variable Documentation

8.9.2.1 PROGMEM

```
const unsigned short empty_sun [2450] PROGMEM
```

8.10 src/GUI/sprites/BRIGHTNESS/full_sun.h File Reference

Variables

- const unsigned short full_sun[2450] **PROGMEM**

8.10.1 Variable Documentation

8.10.1.1 PROGMEM

```
const unsigned short full_sun [2450] PROGMEM
```

Initial value:

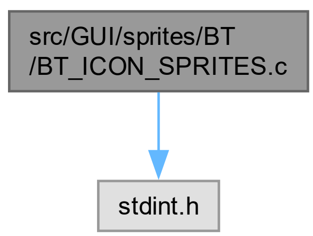
```
={0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0x0000, 0x0000, 0x0000, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0x0000, 0x0000, 0x0000,
      0x0000, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0x0000, 0x0000, 0x0000, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0x0000, 0x0000, 0x0000,
      0x0000, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0x0000,
      0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0x0000, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0000, 0x0000, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
```

```
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0x0000, 0x0000, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff,
      0xffff, 0xffff, 0xffff, 0xffff, 0xffff}
```

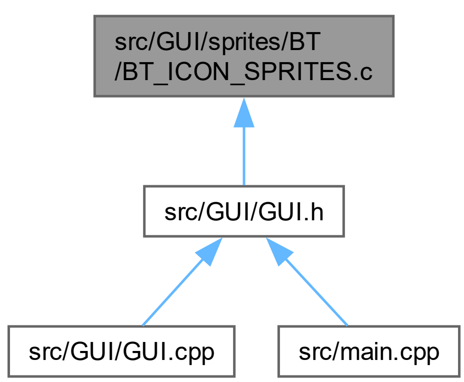
8.11 src/GUI/sprites/BT/BT_ICON_SPRITES.c File Reference

```
#include <stdint.h>
```

Include dependency graph for BT_ICON_SPRITES.c:



This graph shows which files directly or indirectly include this file:



Macros

- #define **BT_ICON_WIDTH** 9
- #define **BT_ICON_HEIGHT** 14
- #define **BT_ICON_DISCONNECTED_HEIGHT** 16
- #define **BT_ICON_DISCONNECTED_WIDTH** 16

Variables

- const uint8_t **BT_ICON_SPRITE** [ ]
- const uint8_t **BT_ICON_DISCONNECTED** [ ]

8.11.1 Macro Definition Documentation

8.11.1.1 BT_ICON_DISCONNECTED_HEIGHT

```
#define BT_ICON_DISCONNECTED_HEIGHT 16
```

8.11.1.2 BT_ICON_DISCONNECTED_WIDTH

```
#define BT_ICON_DISCONNECTED_WIDTH 16
```

8.11.1.3 BT_ICON_HEIGHT

```
#define BT_ICON_HEIGHT 14
```

8.11.1.4 BT_ICON_WIDTH

```
#define BT_ICON_WIDTH 9
```

8.11.2 Variable Documentation

8.11.2.1 BT_ICON_DISCONNECTED

```
const uint8_t BT_ICON_DISCONNECTED[ ]
```

Initial value:

```
= {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xb6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0xff, 0xff, 0xdb, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x00, 0xb6, 0x24, 0xdb, 0xff, 0x6d, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x00, 0x24, 0x00, 0x00, 0x92, 0xff, 0xb6, 0x00, 0x00, 0x00,
  0x00, 0x00, 0xb6, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0xdb, 0xff, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x24, 0xb6, 0x6d, 0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x92, 0x24, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x92, 0xb6, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x6d, 0xb6, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x6d, 0xff, 0xdb, 0x6d, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x92, 0xff, 0x92, 0xff, 0x92, 0x92, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x24, 0xdb, 0xff, 0x6d, 0x00, 0xff, 0x00, 0x6d, 0xb6, 0x24, 0x00, 0x00, 0x24, 0x00,
  0x00, 0x00, 0xff, 0xdb, 0x24, 0x00, 0x00, 0xff, 0x00, 0x00, 0x24, 0xb6, 0x49, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x6d, 0x00, 0x00, 0x00, 0x00, 0xff, 0x00, 0x00, 0x92, 0xff, 0x92, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0x24, 0xdb, 0xff, 0x6d, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xdb, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xb6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
}
```

8.11.2.2 BT_ICON_SPRITE

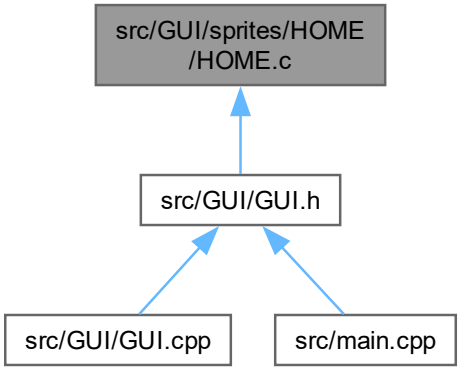
```
const uint8_t BT_ICON_SPRITE[]
```

Initial value:

```
= {
      0x00,0x00,0x00,0x00,0xb6,0x49,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0xff,0xff,0x6d,0x00,0x00
    ,0x00,0x00,0x00,0x00,0xff,0x6d,0xff,0xb6,0x00
    ,0x92,0x00,0x00,0x00,0xff,0x00,0x24,0xdb,0xdb
    ,0xb6,0xdb,0x49,0x00,0xff,0x00,0x49,0xdb,0xb6
    ,0x00,0x92,0xff,0x92,0xff,0x92,0xff,0x92,0x00
    ,0x00,0x00,0x49,0xdb,0xff,0xdb,0x49,0x00,0x00
    ,0x00,0x00,0x49,0xdb,0xff,0xdb,0x49,0x00,0x00
    ,0x00,0x92,0xff,0x92,0xff,0x92,0xff,0x92,0x00
    ,0xb6,0xdb,0x49,0x00,0xff,0x00,0x49,0xdb,0xb6
    ,0x92,0x00,0x00,0x00,0xff,0x00,0x24,0xdb,0xdb
    ,0x00,0x00,0x00,0x00,0xff,0x6d,0xff,0xb6,0x00
    ,0x00,0x00,0x00,0x00,0xff,0xff,0x6d,0x00,0x00
    ,0x00,0x00,0x00,0x00,0xb6,0x49,0x00,0x00,0x00
}
```

8.12 src/GUI/sprites/HOME/HOME.c File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define **HOME_GRANASAT_WIDTH** 100
- #define **HOME_GRANASAT_HEIGHT** 100

Variables

- const unsigned short **logo_granasat** [54000]

8.12.1 Macro Definition Documentation

8.12.1.1 HOME_GRANASAT_HEIGHT

```
#define HOME_GRANASAT_HEIGHT 100
```

8.12.1.2 HOME_GRANASAT_WIDTH

```
#define HOME_GRANASAT_WIDTH 100
```

8.12.2 Variable Documentation

8.12.2.1 logo_granasat

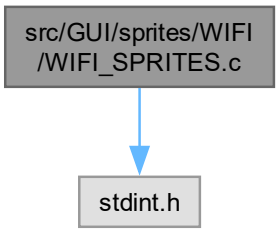
```
const unsigned short logo_granasat[54000]
```

8.13 src/GUI/sprites/WIFI/WIFI_ICON_DISCONNECTED.h File Reference

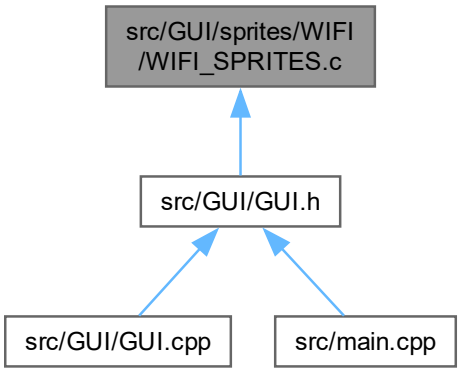
8.14 src/GUI/sprites/WIFI/WIFI_SPRITES.c File Reference

```
#include <stdint.h>
```

Include dependency graph for WIFI_SPRITES.c:



This graph shows which files directly or indirectly include this file:



Macros

- #define **WIFI_ICON_WIDTH** 16
- #define **WIFI_ICON_HEIGHT** 16
- #define **WIFI_RSSI_GOOD** -67
- #define **WIFI_RSSI_OKAY** -70

Variables

- const uint8_t **WIFI_GOOD_ICON_SPRITE** [ ]
- const uint8_t **WIFI_OKAY_ICON_SPRITE** [ ]
- const uint8_t **WIFI_BAD_ICON_SPRITE** [ ]
- const uint8_t **WIFI_ICON_DISCONNECTED** [ ]

8.14.1 Macro Definition Documentation

8.14.1.1 WIFI_ICON_HEIGHT

```
#define WIFI_ICON_HEIGHT 16
```

8.14.1.2 WIFI_ICON_WIDTH

```
#define WIFI_ICON_WIDTH 16
```

8.14.1.3 WIFI_RSSI_GOOD

```
#define WIFI_RSSI_GOOD -67
```

8.14.1.4 WIFI_RSSI_OKAY

```
#define WIFI_RSSI_OKAY -70
```

8.14.2 Variable Documentation

8.14.2.1 WIFI_BAD_ICON_SPRITE

```
const uint8_t WIFI_BAD_ICON_SPRITE[]
```

Initial value:

```
= {
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0x24,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x92,0xff,0xff,0xff,0xff,0x92,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0xdb,0xdb,0x6d,0x6d,0xdb,0xdb,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0xff,0xff,0x24,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0xdb,0xdb,0x24,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0x24,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
}
```

8.14.2.2 WIFI_GOOD_ICON_SPRITE

```
const uint8_t WIFI_GOOD_ICON_SPRITE[]
```

Initial value:

```
= {
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x49,0x6d,0x92,0x92,0x6d,0x49,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x24,0x92,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0x92,0x24,0x00,0x00
   ,0x00,0x6d,0xff,0xff,0xb6,0x49,0x24,0x24,0x24,0x24,0x49,0xb6,0xff,0xff,0x6d,0x00
   ,0x92,0xff,0xdb,0x49,0x00,0x24,0x49,0x6d,0x6d,0x49,0x24,0x00,0x49,0xdb,0xff,0x92
   ,0x00,0x6d,0x00,0x24,0x92,0xff,0xff,0xff,0xff,0xff,0xff,0x92,0x24,0x00,0x6d,0x00
   ,0x00,0x00,0x49,0xff,0xff,0xdb,0x6d,0x24,0x24,0x6d,0xdb,0xff,0xff,0x49,0x00,0x00
   ,0x00,0x00,0x00,0xb6,0x49,0x00,0x00,0x24,0x24,0x00,0x00,0x49,0xb6,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x92,0xff,0xff,0xff,0xff,0x92,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0xdb,0xdb,0x6d,0x6d,0xdb,0xdb,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0xff,0xff,0x24,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0xdb,0xdb,0x24,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0x24,0x00,0x00,0x00,0x00,0x00,0x00,0x00
   ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
}
```

8.14.2.3 WIFI_ICON_DISCONNECTED

```
const uint8_t WIFI_ICON_DISCONNECTED[]
```

Initial value:

```
= {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x24, 0x24, 0x6d, 0x92, 0x92, 0x6d, 0x49, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x24, 0x24, 0x24, 0x92, 0xff, 0xff, 0xff, 0xff, 0xff, 0x92, 0x24, 0x00, 0x00,
  0x00, 0x6d, 0x6d, 0x24, 0x24, 0x24, 0x24, 0x24, 0x24, 0x24, 0x49, 0xb6, 0xff, 0xff, 0x6d, 0x00,
  0x92, 0xff, 0xdb, 0x24, 0x00, 0x00, 0x00, 0x24, 0x24, 0x24, 0x24, 0x00, 0x49, 0xdb, 0xff, 0x92,
  0x00, 0x6d, 0x00, 0x24, 0x92, 0x49, 0x24, 0x24, 0x24, 0x24, 0x92, 0x92, 0x24, 0x00, 0x6d, 0x00,
  0x00, 0x00, 0x49, 0xff, 0xff, 0xb6, 0x24, 0x24, 0x24, 0x24, 0x24, 0x6d, 0xdb, 0x24, 0x00, 0x00,
  0x00, 0x00, 0x00, 0xb6, 0x49, 0x00, 0x24, 0x24, 0x24, 0x00, 0x00, 0x24, 0x24, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x92, 0xff, 0xff, 0xff, 0x49, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xdb, 0xdb, 0x6d, 0x6d, 0xb6, 0x6d, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0xff, 0xff, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0xdb, 0xdb, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
}
```

8.14.2.4 WIFI_OKAY_ICON_SPRITE

```
const uint8_t WIFI_OKAY_ICON_SPRITE[]
```

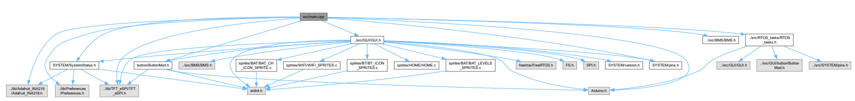
Initial value:

```
= {
     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0x6d,0x6d,0x24,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x24,0x92,0xff,0xff,0xff,0xff,0xff,0xff,0x92,0x24,0x00,0x00,0x00
    ,0x00,0x00,0x49,0xff,0xff,0xdb,0x6d,0x24,0x24,0x6d,0xdb,0xff,0xff,0x49,0x00,0x00
    ,0x00,0x00,0x00,0xb6,0x49,0x00,0x00,0x24,0x24,0x00,0x00,0x49,0xb6,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x92,0xff,0xff,0xff,0xff,0x92,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0xdb,0xdb,0x6d,0x6d,0xdb,0xdb,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0xff,0xff,0x24,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0xdb,0xdb,0x24,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0x24,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
}
```

8.15 src/main.cpp File Reference

```
#include <Arduino.h>
#include "../lib/Adafruit_INA219/Adafruit_INA219.h"
#include "../lib/Preferences/Preferences.h"
#include "../lib/TFT_eSPI/TFT_eSPI.h"
#include "../src/GUI/GUI.h"
#include "../src/GUI/button/ButtonMod.h"
#include "../src/SYSTEM/pins.h"
#include "../src/SYSTEM/SystemStatus.h"
#include "../src/BMS/BMS.h"
#include "../src/RTOS_tasks/RTOS_tasks.h"
```

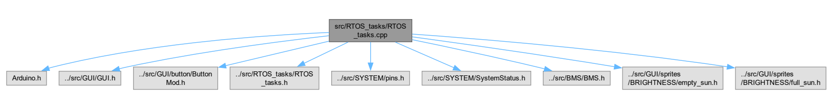
Include dependency graph for main.cpp:



8.16 src/RTOS_tasks/RTOS_tasks.cpp File Reference

```
#include <Arduino.h>
#include "../src/GUI/GUI.h"
#include "../src/GUI/button/ButtonMod.h"
#include "../src/RTOS_tasks/RTOS_tasks.h"
#include "../src/SYSTEM/pins.h"
#include "../src/SYSTEM/SystemStatus.h"
#include "../src/BMS/BMS.h"
#include "../src/GUI/sprites/BRIGHTNESS/empty_sun.h"
#include "../src/GUI/sprites/BRIGHTNESS/full_sun.h"
```

Include dependency graph for RTOS_tasks.cpp:



Variables

- uint16_t **value_brightness**
- static struct **SystemStatus SYSTEM_STATUS**

  Global system status variable.
- static SemaphoreHandle_t **MUTEX_SYS_STATUS** = xSemaphoreCreateMutex()

  Mutex for protecting access to the system status.
- static SemaphoreHandle_t **MUTEX_SPI** = xSemaphoreCreateMutex()

  Mutex for protecting access to the SPI bus.
- xTaskHandle **screenManager_TaskHandler**

  Task handler for screenManager task.
- xTaskHandle **refreshTouch_TaskHandler**

  Task handler for statusBarRefresher task.
- static SemaphoreHandle_t **MUTEX_TOUCH_POS** = xSemaphoreCreateMutex()

  Structure for touch position detection.
- static enum **screens SCREEN_ID** = **screens::MAIN**

  Enum for screen IDs.
- static struct **touch_pos TOUCH_POS**
- const uint32_t **STATUS_BAR_TICKS** = 5000 / portTICK_PERIOD_MS
- xTaskHandle **statusBarRefresher_TaskHandler**
- xTaskHandle **monitorBattery_TaskHandler**
- xTaskHandle **OnOffManager_TaskHandler**
- **GUI gui**

8.16.1 Variable Documentation

8.16.1.1 gui

```
GUI gui
```

Initial value:

```
= GUI(&SYSTEM_STATUS, &MUTEX_SYS_STATUS,
              &TOUCH_POS, &MUTEX_TOUCH_POS,
              &SCREEN_ID,

              &MUTEX_SPI)
```

8.16.1.2 monitorBattery_TaskHandler

```
xTaskHandle monitorBattery_TaskHandler
```

8.16.1.3 MUTEX_SPI

```
SemaphoreHandle_t MUTEX_SPI = xSemaphoreCreateMutex()  [static]
```

Mutex for protecting access to the SPI bus.

8.16.1.4 MUTEX_SYS_STATUS

```
SemaphoreHandle_t MUTEX_SYS_STATUS = xSemaphoreCreateMutex()  [static]
```

Mutex for protecting access to the system status.

8.16.1.5 MUTEX_TOUCH_POS

```
SemaphoreHandle_t MUTEX_TOUCH_POS = xSemaphoreCreateMutex()  [static]
```

Structure for touch position detection.

8.16.1.6 OnOffManager_TaskHandler

```
xTaskHandle OnOffManager_TaskHandler
```

8.16.1.7 refreshTouch_TaskHandler

```
xTaskHandle refreshTouch_TaskHandler
```

Task handler for statusBarRefresher task.

8.16.1.8 SCREEN_ID

```
enum  screens SCREEN_ID =  screens::MAIN  [static]
```

Enum for screen IDs.

8.16.1.9 screenManager_TaskHandler

```
xTaskHandle screenManager_TaskHandler
```

Task handler for screenManager task.

8.16.1.10 STATUS_BAR_TICKS

```
const uint32_t STATUS_BAR_TICKS = 5000 / portTICK_PERIOD_MS
```

8.16.1.11 statusBarRefresher_TaskHandler

```
xTaskHandle statusBarRefresher_TaskHandler
```

8.16.1.12 SYSTEM_STATUS

```
struct  SystemStatus SYSTEM_STATUS  [static]
```

Global system status variable.

8.16.1.13 TOUCH_POS

```
struct touch_pos TOUCH_POS [static]
```

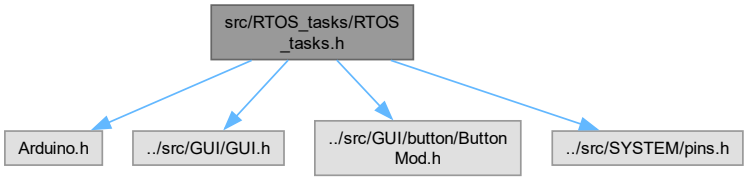
8.16.1.14 value_brightness

```
uint16_t value_brightness
```

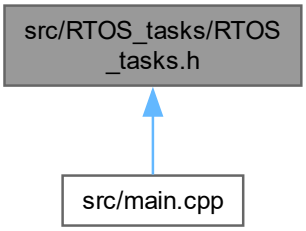
8.17 src/RTOS_tasks/RTOS_tasks.h File Reference

```
#include <Arduino.h>
#include "../src/GUI/GUI.h"
#include "../src/GUI/button/ButtonMod.h"
#include "../src/SYSTEM/pins.h"
```

Include dependency graph for RTOS_tasks.h:



This graph shows which files directly or indirectly include this file:



Functions

- void **refreshTouch** (void *parameters)

  Function decclaration drom the .cpp.
- void **statusBarRefresher** (void *parameters)
- void **screenManager** (void *parameters)
- void **monitorBattery** (void *parameters)
- void **turn_on_off** ()
- void **OnOffManager** (void *parameters)

Variables

- xTaskHandle **screenManager_TaskHandler**

  Task handler for screenManager task.
- xTaskHandle **refreshTouch_TaskHandler**

  Task handler for statusBarRefresher task.
- xTaskHandle **statusBarRefresher_TaskHandler**
- xTaskHandle **monitorBattery_TaskHandler**
- xTaskHandle **OnOffManager_TaskHandler**
- **GUI gui**

8.17.1 Function Documentation

8.17.1.1 monitorBattery()

```
void monitorBattery (
            void * parameters)  [extern]
```

8.17.1.2 OnOffManager()

```
void OnOffManager (
            void * parameters)  [extern]
```

8.17.1.3 refreshTouch()

```
void refreshTouch (
            void * parameters)  [extern]
```

Function decclaration drom the .cpp.

8.17.1.4 screenManager()

```
void screenManager (
            void * parameters)  [extern]
```

8.17.1.5 statusBarRefresher()

```
void statusBarRefresher (
            void * parameters)  [extern]
```

8.17.1.6 turn_on_off()

```
void turn_on_off ()  [extern]
```

8.17.2 Variable Documentation

8.17.2.1 gui

```
GUI gui  [extern]
```

8.17.2.2 monitorBattery_TaskHandler

```
xTaskHandle monitorBattery_TaskHandler  [extern]
```

8.17.2.3 OnOffManager_TaskHandler

```
xTaskHandle OnOffManager_TaskHandler  [extern]
```

8.17.2.4 refreshTouch_TaskHandler

```
xTaskHandle refreshTouch_TaskHandler  [extern]
```

Task handler for statusBarRefresher task.

8.17.2.5 screenManager_TaskHandler

```
xTaskHandle screenManager_TaskHandler  [extern]
```

Task handler for screenManager task.

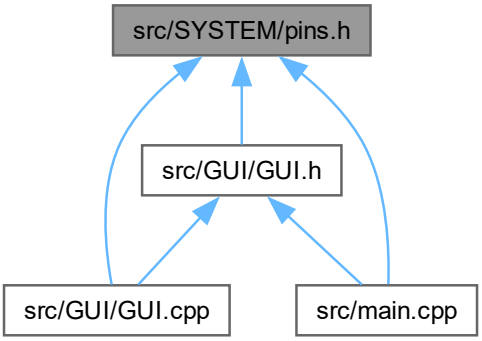
8.17.2.6 statusBarRefresher_TaskHandler

```
xTaskHandle statusBarRefresher_TaskHandler  [extern]
```

8.18 src/SYSTEM/pins.h File Reference

All pins definitions of the device.

This graph shows which files directly or indirectly include this file:



Macros

- #define **TFT_MISO** 1

  Pin for the Master Input Slave Output.
- #define **TFT_MOSI** 2

  Pin fot the Master Output Slave Input.
- #define **TFT_SCLK** 1

  Pin for clock SPI.
- #define **TFT_CS** 1

  Pin for the Chip Select screen.
- #define **TFT_DC**

  Pin for data command.
- #define **TFT_RST**

  Pin for reset screen.
- #define **TOUCH_CS** 1

  Pin for chip select from the touch IC.
- #define **PIN_SD_CS**

  Pin for chip select from the SD.
- #define **SPI_FREQUENCY** 80000000

  SPI frequency.
- #define **SPI_READ_FREQUENCY** 20000000

  SPI read frequency.
- #define **SPI_TOUCH_FREQUENCY** 2500000

  SPI touch frequency.
- #define **PIN_EMITTER** 3

  Emitter analog signal from ESP32 through BNC connector.
- #define **PIN_RECEIVER** 3

  Receiver analog signal from ESP32 through BNC connector.
- #define **PIN_THUMB_SW_CCW** 2

  Pin from the thumb button counter clockwise.
- #define **PIN_THUMB_SW_PUSH** 1

  Pin from the thumb button to push the center like a taditional button.
- #define **PIN_THUMB_SW_CW** 1

  Pin from the thumb button clockwise.
- #define **PIN_GPIO0**

  General purpouse input output 0 to flashing system.
- #define **PIN_LED** 1

  Brightness control through the N-Mosfet integrated on the module ILI9341.
- #define **PIN_TEST** 17
- #define **PIN_I2C_SCL** 2

  I2C pin with clock signal.
- #define **PIN_I2C_SDA** 2

  I2C pin with data transfer.
- #define **PIN_POWER_SW_USER** 3

  Defined like a input pin to read the activity user.
- #define **PIN_POWER_SW_HOLD** 2

  Defined like a output pin to hold a high logic value and power on the enable buck.

8.18.1 Detailed Description

All pins definitions of the device.

Version

1.4

Date

2024-06-27

Version

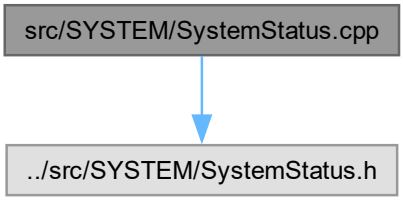
1.3

8.19 src/SYSTEM/SystemStatus.cpp File Reference

Systemstatus on the device.

```
#include "../src/SYSTEM/SystemStatus.h"
```

Include dependency graph for SystemStatus.cpp:



Variables

- Adafruit_INA219 **ina219**

  INA219 sensor object for measuring voltage and current.
- TFT_eSPI **_tft** = TFT_eSPI()

  TFT display object for rendering graphics and text on the screen.
- Preferences **preferences**

  Preferences object for managing non-volatile storage.

8.19.1 Detailed Description

Systemstatus on the device.

Implementation file for system status management.

This file contain the functions used and pinned to core on **main.cpp** (p. 100) to manage how the device works in the both cores of the ESP32. Here you can see the RTOS functionos like a deterministic system deterministic system with full knowledge of what is going to happen in our equipment.

8.19.2 Libraries

Arduino.h> // Arduino Framework

// Hardware-specific library.

// LCD user interface

// system status variable

// Hardware-specific library.

The necessary libraries for the program to function correctly are included in the directory src.

This file includes the necessary headers and initializes global objects used for interfacing with the INA219 sensor, TFT display, and for managing non-volatile storage (NVS) using the Preferences library.

8.19.3 Variable Documentation

8.19.3.1 _tft

```
TFT_eSPI _tft = TFT_eSPI()
```

TFT display object for rendering graphics and text on the screen.

TFT object from the TFT_eSPI library.

The `TFT_eSPI` object is used to control the TFT display, enabling the drawing of graphics, text, and other visual elements on the screen. Object for managing and rendering on the TFT display.

8.19.3.2 ina219

```
Adafruit_INA219 ina219
```

INA219 sensor object for measuring voltage and current.

ina219 object from the INA210 library

< Includes the header file for system status management.

The `Adafruit_INA219` object is used to interface with the INA219 sensor, which is capable of measuring current, voltage, and power consumption. Object for interacting with the INA219 sensor.

8.19.3.3 preferences

```
Preferences preferences
```

Preferences object for managing non-volatile storage.

Preferences object from the Preferences library.

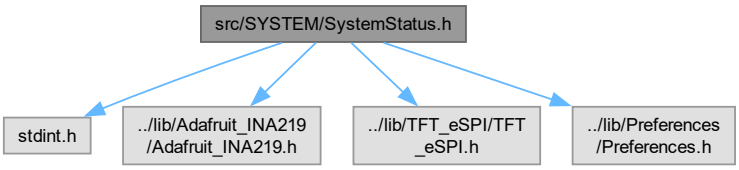
The `Preferences` object provides functionality for storing and retrieving data from non-volatile storage (NVS), ensuring that data persists across power cycles and resets. Object for accessing and managing non-volatile storage.

8.20 src/SYSTEM/SystemStatus.h File Reference

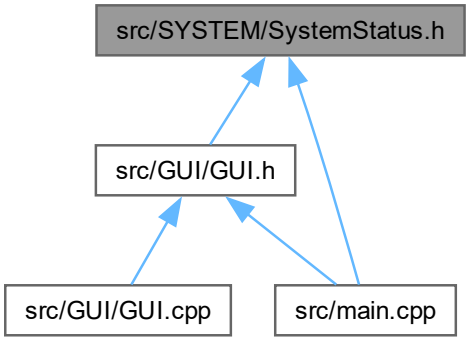
Implementation file for constum configuration module functions.

```
#include <stdint.h>
#include "../lib/Adafruit_INA219/Adafruit_INA219.h"
#include "../lib/TFT_eSPI/TFT_eSPI.h"
#include "../lib/Preferences/Preferences.h"
```

Include dependency graph for SystemStatus.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct **SystemStatus**

  Structure to hold various system status information.

Variables

- TFT_eSPI **_tft**

  TFT object from the TFT_eSPI library.
- Adafruit_INA219 **ina219**

  ina219 object from the INA210 library
- Preferences **preferences**

  Preferences object from the Preferences library.

8.20.1 Detailed Description

Implementation file for constum configuration module functions.

On this file has been declared the mutex and the variables used in the system, also the boolean variables used like flags to advice free utilities in the system.

8.20.2 Libraries

```
@include <stdint.h>
@include "../src/SYSTEM/SystemStatus.cpp"  // system status variable
```

The necessary libraries for the program to function correctly are included.

Warning

The mutexes created are critical to the proper functioning of the computer, if any of these are not released after a task takes them, it may trigger an error for the user and the computer will reboot.

Returns

Returns free mutexes once tasks stop taking them

8.20.3 Variable Documentation

8.20.3.1 _tft

```
TFT_eSPI _tft  [extern]
```

TFT object from the TFT_eSPI library.

TFT object from the TFT_eSPI library.

The `TFT_eSPI` object is used to control the TFT display, enabling the drawing of graphics, text, and other visual elements on the screen. Object for managing and rendering on the TFT display.

8.20.3.2 ina219

```
Adafruit_INA219 ina219  [extern]
```

ina219 object from the INA210 library

ina219 object from the INA210 library

< Includes the header file for system status management.

The `Adafruit_INA219` object is used to interface with the INA219 sensor, which is capable of measuring current, voltage, and power consumption. Object for interacting with the INA219 sensor.

8.20.3.3 preferences

```
Preferences preferences  [extern]
```

Preferences object from the Preferences library.

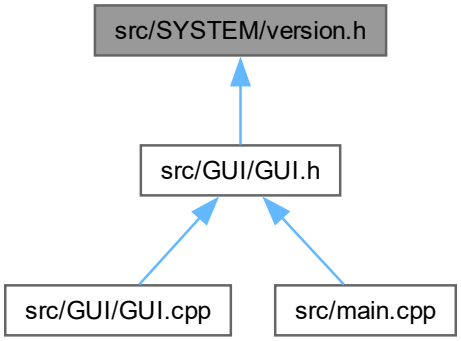
Preferences object from the Preferences library.

The `Preferences` object provides functionality for storing and retrieving data from non-volatile storage (NVS), ensuring that data persists across power cycles and resets. Object for accessing and managing non-volatile storage.

8.21 src/SYSTEM/version.h File Reference

Version of the firmware.

This graph shows which files directly or indirectly include this file:



Macros

- #define **TIK_FW_VERSION** "V1.2-20240731"

8.21.1 Detailed Description

Version of the firmware.

Version

1.0

Date

2024-06-27

Copyright

Copyright (c) 2024

8.21.2 Macro Definition Documentation

8.21.2.1 TIK_FW_VERSION

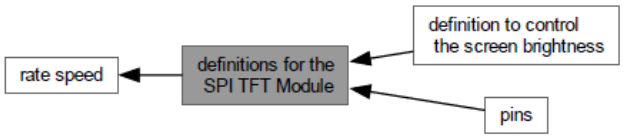
```
#define TIK_FW_VERSION "V1.2-20240731"
```

Index

6.1.2 definitions for the SPI TFT Module

Pins connected to the device for use the SPI protocol. Is needed for use the screen, touch chip and SD capabilities.

Collaboration diagram for definitions for the SPI TFT Module:



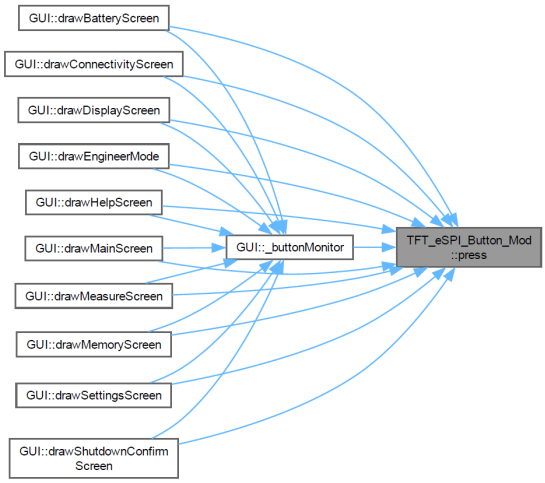
Topics

- **definition to control the screen brightness**
  Pin to adjust the brightness of the display thanks to the mosfet n that is integrated in the ili9341 module.
- **pins**
  These pins manage the on/off of the device, reading the user activity and hold the enable buck with the 3.3V to supply the system.

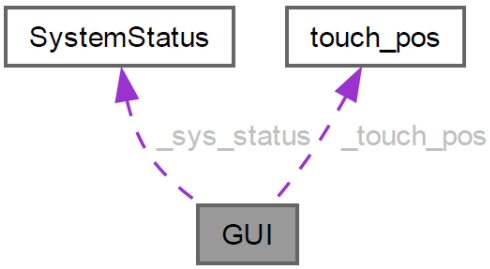
Macros

- #define **TFT_MISO** 1
  Pin for the Master Input Slave Output.
- #define **TFT_MOSI** 2
  Pin fot the Master Output Slave Input.
- #define **TFT_SCLK** 1
  Pin for clock SPI.
- #define **TFT_CS** 1
  Pin for the Chip Select screen.
- #define **TFT_DC**
  Pin for data command.
- #define **TFT_RST**
  Pin for reset screen.
- #define **TOUCH_CS** 1

(a) *PDF generado por Doxygen 1.11.0*



(a) *Ejemplo de gráfico de dependencia generado con Graphviz en Doxygen 1.11.0*



(b) *Ejemplo de gráfico de colaboración generado con Graphviz en Doxygen 1.11.0*

Apéndice E

Instrumentación electrónica

E.1. Power Analyzer N6705A

El N6705A Power Analyzer de Keysight es un equipo que punciona como salida y entrada de potencia en cada uno de sus 4 fuentes. Combina las funcionalidades de fuente de alimentación programable, multímetro, osciloscopio y generador de señales en un solo instrumento, facilitándonos el análisis de consumo de energía.

El equipo cuenta con una pantalla a color en la cual podemos ver las 4 fuentes de las que dispone y de la potencia que está entregando o recibiendo. Para recibir o emitir datos soporta control remoto vía USB, LAN, o GPIB. Para comunicarme con el equipo usé el puerto USB A de mi ordenador y el puerto USB tipo B del Power Analyzer. Busco leer datos tanto del meter view como del data logger (meter view es la opción de multímetro y data logger la de osciloscopio figura E.2). Para ello se emplea la librería PyVisa y los comandosde cada equipo especificados en su manual de usuario en algunos de los apéndices (power analyzer Appendix C).

Figura E.1 – *Power Analalyzer N6705A de los laboratorios de GranaSAT.*

Estos equipos suelen contar con el estandar SCPI que nos permite constrolar los instrumentos de medición a distancia. Nos permiten medir voltaje y corriente en el canal que seyleccionemos, establecer límites de protección si en alguna fuente estamos alcanzando valores peligrosos además de reconocer el instrumento y que el mismo nos mande el nombre del equipo (entre otras muchas funcionalidades). Matizar en este última que, cuando un laboratorio está bien instrumentalizado, los equipos se controlan totalmente desde el ordenador de mando y así podemos ajustar a números precisos deseados en vez de usar ruletas analógicas imprecisas, por ello, hay veces que necesitamos requerir el nombre del equipo al que en ese momento estamos conectados. Estos comandos y programación ha sido ejecutada mediante el lenguaje de programación python, y ademas he usado pandas tener un dataframe y amnipular los dato más fácilmente. Por otro lado, la librería time para medir tiempos de ejecución.

```
1
2     # Este código de comunicación con el equipo Power Analyzer N6705A ha
  sido implementado
3     # en un primer momento por Irene Gil Martín y en segundo lugar por Juan
  Manuel Martín Lucena Curso 2023/2024.
4
5     import pyvisa
6     import numpy as np
7     import pandas as pd
8     import visa, time
9     import time
10
11    begin = time.time()
12
13    debug_ = 1 # 0 si no quiero mostrar datos en la terminal, 1 si quiero
  mostrarlos
14
15    resource_manager = pyvisa.ResourceManager() # incio objeto que nos
  muestra las librerías de los sistemas conectados
16
17    def List_Resources(resource_manager_object):
18        connected_resources = resource_manager_object.list_resources() #
  obtengo los equipos conectados al PC
19
20        if debug_:
21            print("Connected Resources:",connected_resources) # muestro los
  equipos conectados si la variable debug
22
23        return connected_resources
24
25    def reset_device(instrument):
26        instrument.write("*rst; status:preset; *cls") #Empieza y resetea el
  equipo para eliminar valores anteriores
27
28        if debug_:
29            print("Reset and Clear Device")
30
31    def setPowerSupplyOutput(instrument,channel,status): # Selecciono el
  instrumento, salida y estado de la fuente
32        if status:
33            instrument.write("OUTPut ON,(@" + str(channel) + ")\n")
34            if debug_:
35                print("Channel " + str(channel) + " On")
36        else:
37            instrument.write("OUTPut OFF,(@" + str(channel) + ")\n")
38            if debug_:
39                print("Channel " + str(channel) + " Off")
40
41    def setVoltage(instrument,channel,voltage): # Escojo el canal y el
  voltaje de salida
42        instrument.write("VOLTage %.2f,(@%d)\n"%(voltage,channel,))
43        if debug_:
44                print("Setting channel" + str(channel) + " voltage with " +
  str(voltage) + " V")
45
46
47    def setCurrent(instrument,channel,current): # Escojo el canal y la
```

```
intensidad de salida
48          instrument.write("CURR %.2f,(@%d)\n"%(current,channel,))
49          if debug_:
50                  print("Setting channel" + str(channel) + " current with " +
str(current) + " A")
51
52      #Values output: Estso son los valores mñaximos de nuestro equipo en
particular del laboratorio de GranaSAT
53      # Max values for each generator Output_channel//(0-VMax)/(0-Imax)
54      # 1/(0-20V)/(0-15A) 2/(0-60V)/(0-1,6A)
55
56      output_channel = 1 # Escojo uno de los cuatro canales a usar
57      set_voltage = 0 # Selecciono voltaje de salida
58      set_current = 0 # Selecciono intensidad de salida
59
60      connected_resources = List_Resources(resource_manager) # listing the
resources
61
62      inst = resource_manager.open_resource(connected_resources[0]) # Device
connected
63
64      #inst.write('*RST') #Reset previous values
65
66      setPowerSupplyOutput(inst,output_channel,True) #Comandos para poner
tensión y alimentación en el equipo
67      setVoltage(inst,output_channel,set_voltage)
68      setCurrent(inst,output_channel,set_current)
69
70
71      print(inst.query('*IDN?')) # Pedimos al equipo que se identifique
72
73      number_measures = 5 # Indicamos cuantas medidas queremos que haga el
multímetro
74
75      # Definimos listas vacías donde irán las mediciones del multímetro
76      lista_volts = []
77      lista_amps = []
78      lista_time = []
79
80      def read_measurements(inst, output_channel, number_measures): # Función
para rellenar las anteriores listas con las mediciones
81
82          for n in range(number_measures):
83              time.sleep(10)
84
85              try:
86                  voltage = inst.query_ascii_values(f'MEAS:VOLT? (@ {
output_channel})')
87                  lista_volts.append(voltage) # añadimos a la lista la medició
n de tensión que acabamos de hacer
88
89                  current = inst.query_ascii_values(f'MEAS:CURR? (@ {
output_channel})')
90                  lista_amps.append(current )# añadimos a la lista la medición
 de tensión que acabamos de hacer
91
92                  finish = time.time()
```

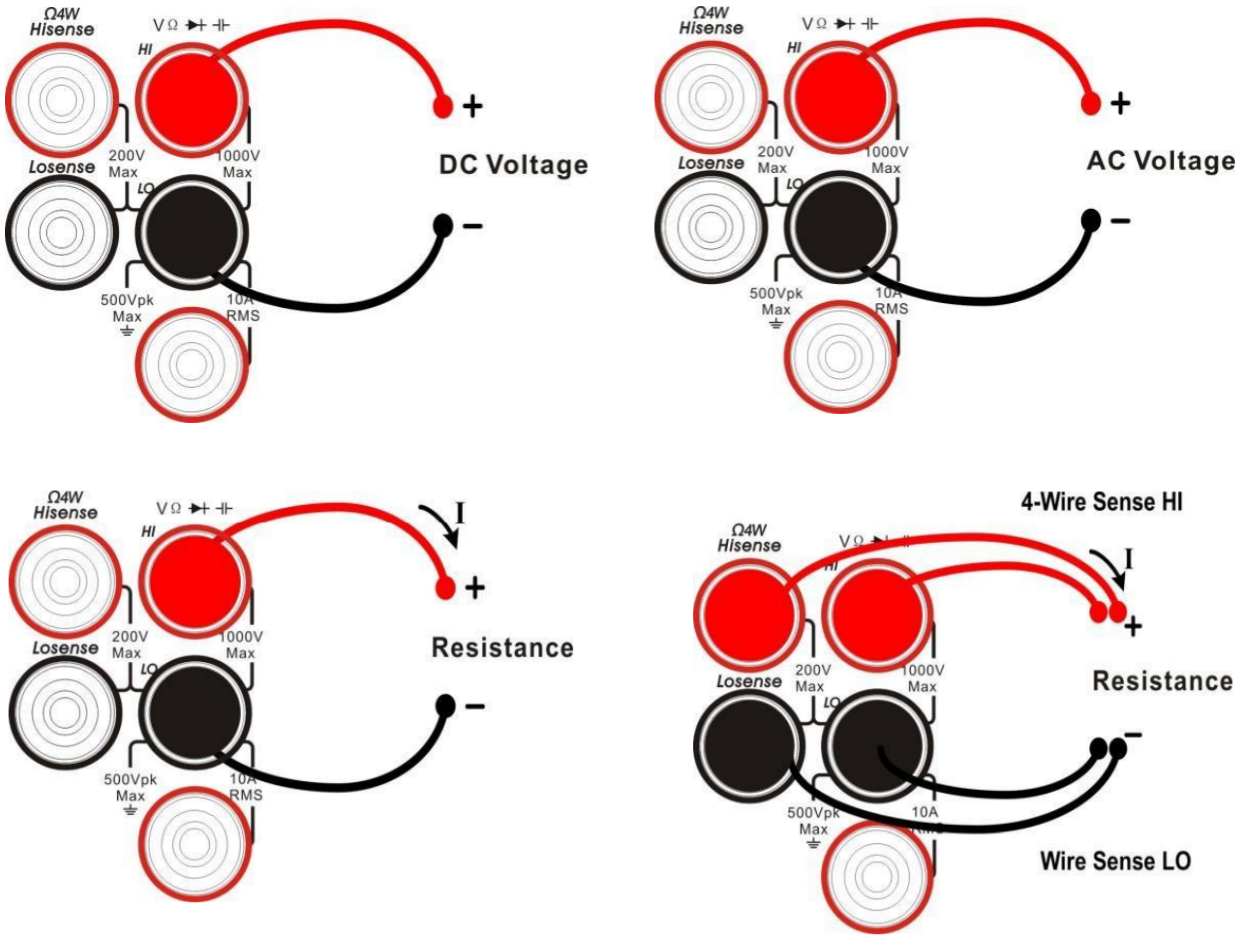
```
93                  lista_time.append(finish - begin) # añadimos a la lista el
    tiempo que ha llevado en hacer la medición
94
95                  #print(f'Measure in volts: {voltage}')
96                  #print(f'Measure in amps: {current}')
97
98              except:
99                  print('No measure available') # Si el equipo no consigue
    hacer la medición devolvemos que no hay medidas disponibles
100
101         return lista_volts, lista_amps, lista_time # Devolvemos las listas
    ya rellenas
102
103     read_measurements(inst, output_channel, number_measures)
104
105     columns_measures = []
106
107     def gen_columns(number_measures): # Leemos el número de columnas, que
    indica el número de mediciones realizadas
108         columns_measures = [f'Measure{i}' for i in range(0, number_measures)
    ]
109         return columns_measures
110
111
112
113     df = pd.DataFrame((lista_volts, lista_amps, lista_time), index = (['V'],
     ['I'], ['T']), columns = (gen_columns(number_measures))) # Creamos un
    dataframe con la librería pandas para obtener la tabla de mediciones.
114
115     print(df)
116
117
118     #voltage, current = response.strip().split(',')  # Divide la respuesta
    en tensión y corriente
119
120
121     #Limitaciones de tension y corriente
122     #inst.write(f'CURR:PRO:STAT .5, (@ {output_channel})')
123     #inst.write(f'VOLT:PROT:STAT .5, (@ {output_channel})')
124
125     #Measurements of data logger with N6705B
126
127
128     #inst.write("SENS:FUNC:VOLT ON,(@1)")
129     ##inst.write("SENS:FUNC:CURR ON,(@1)") Cant read 2 values at same time
130     #inst.write("SENS:SWE:TINT .001,(@1)")
131     #inst.write("SENS:SWE:POIN 5000,(@1)")
132     #inst.write("INIT:ACQ (@1)")
133     #inst.write("TRIG:ACQ (@1)")
134     #x=inst.query_ascii_values("FETC:ARR:VOLT? (@1)")
135     #print (x)
```

Listado E.1 – *Código en Python*

El equipo de GranaSAT como he comentado cuenta con 4 módulos de fuente de alimentación. Una de ellas con tensión de 0-20V y 0-15A que da una una potencia total de 300 W, y dos de ellas cuentan con tensiones de 0-60V y intensidad de 0-1.6V que hacen una potencia 96W. La cuarta fuente de alimentación no contaba en el laboratorio con las especificaciones. Como se puede ver en la imagen E.4

E.2. SDM3065X Series Digital Multimeter

El multímetro digital de Siglent SDM3065X es el usado en el laboratorio de GranaSAT cuando se necesita una alta resolución, en este caso el equipo cuenta con 6 1/2 de dígitos. El equipo va más allá de medir voltaje, corriente, resistencia, capacitancia, freccuencia, temperatura y comprobar continuidad además de diodos. Cuenta con una conectividad de USB, LAN y GPIB. Además cuenta con la funcion de true RMS para señales periódicas alternas. Todas las funcionalidades vienen explicadas yd etalladas en su manual de usuario [17].



La medición de resistencias de bajo valor está determinada por el número de cables que se usen. Se pueden usar 2, 3 y 4 cables consiguiendo que las resistencias R_w propias de los cables queden en paralelo disminuyendo así su valor predominando el valor de la resistencia a medir deseada como se puede ver ne las figuras E.6.

E.2.1. Medición en 2 cables

En la medición de 2 cables, la resistencia total medida R_{total} incluye la resistencia del cable R_w de ambos cables:

$$R_{\text{total}} = R + 2R_w$$

E.2.2. Medición en 3 cables

En la medición de 3 cables, se reduce parcialmente el error debido a la resistencia del cable, pero todavía se incluye una parte de R_w:

$$R_{\text{total}} = R + R_w + \frac{R_w}{2}$$

E.2.3. Medición en 4 cables

En la medición de 4 cables, la resistencia del cable R_w se cancela completamente, por lo que la resistencia total medida es solo la resistencia R:

$$R_{\text{total}} = R + \frac{Rw}{2} + \frac{Rw}{2}$$

E.3. POWER SUPPLY FAC-363B

La fuente de tensión FAC 363 B [51] es la utilizada en el día a día y que estaba presente en todas las mesas del laboratorio de GranaSAT. Contiene tres fuentes de alimentación independientes. La primera y única usada tiene una tensión ajustable de 0 a 30 V con una corriente máxima de 2 A. La segunda es fija de -15 V y +15 V con un máximo de corriente de 0.5 A. La terdera y última es fija de 5 V y de hasta 1 A que simula un puerto USB.

Característica	Especificación
Salida 0-30 V	
Tensión máxima de salida	30 V
Intensidad máxima de salida	2 A
Resistencia interna	6 mohms a 1 kHz, 10 mohms a 10 kHz
Regulación de carga (0 a 100 %)	0,05 % + 2 mV
Regulación de red (± 10 %)	0,05 % + 2 mV
Tiempo de recuperación (I de 10 a 100 %)	<50 us
Ruido y zumbido	500 uV rms
Salida 5 V	
Intensidad máxima de salida	1 A
Regulación de carga (0 a 100 %)	<1,5 %
Regulación de red (± 10 %)	<1 %
Ruido y zumbido	<2 mV rms
Salida ±15 V	
Intensidad máxima de salida	0,5 A
Regulación de carga (0 a 100 %)	<1,5 %
Regulación de red (± 10 %)	<1 %
Ruido y zumbido	<2 mV rms
Alimentación	
Tensión de red AC	110,125,220,230,240 V ± 10 % / 50 - 60 Hz
Consumo	120 W
Temperatura Ambiente Máxima	
Temperatura máxima	40 ºC
Características Mecánicas	
Dimensiones	A. 230 x Al. 145 x Pr. 290 mm
Peso	6 kg

E.4. KEITHLEY 220 PROGRAMABLE CURRENT SOURCE

Keithley 220 ofrece un amplio rango de corriente de salida, desde 100 picoamperios a como máximo 100 miliamperios. La precisión es una de las características del equipo, permitiendo porporcionar una intensidad fiable y estable además de proporcionar bajo ruido. El equipo puede ser controlado mediante un USB y un puerto GPIB.

Las fuentes de corriente son equipos más difíciles de ver y que en el grado no había manipulado. Funciona igual que una fuente de tensión pero con la corriente. En las fuentes de tensión ajustables como la FAC-363B descrita anteriormente, cuando provocamos un cortocircuito entre sus bornes podemos ajustar la máxima corriente que aplica la fuente al detectar el corto. En el caso de una fuente de intensidad, nosotros ajustamos la intensida deseada y en el cortocircuito, ajustamos la tensión máxima entre sus bornes. Esta fuente trabaja con un rango de 1.995nA a 101.00mA con resolución de intensidad de 1nA a menos de 100uA+-3 % con una tensión límite de 1V a 105V con resolución de 1V. Fácilmente con un boton de start y stop podemos controlar el inicio o parada de la corriente.

Las fuentes de intensidad son muy utilizadas en:

- Experimentación con semiconductores: Muy usada en pruebas de dispositivos como diodos y transistores, donde se necesita alta precisión de la corriente.
- Medición y caracterización de materiales: Utilizada para aplicar corrientes precisas en experimentos de resistencia eléctrica y estudios de materiales.
- Calibración de instrumentos: Debido a su alta precisión, es utilizada para calibrar otros equipos de medición.

Por este tercer motivo ha sido el objetivo de usar la fuente de corriente. Como se ha descrito en el capìtulo 2, se ha necesitado calibrar el INA219 que es el que se encarga de la monitorización de la batería. Para ello se necesitó ajustar el valor de registro cal_value de la librería descrito en el capítulo 4.

E.5. Osciloscopios usados: HAMEG HMO1024 y TEKTRONIK TDS 1002

El osciloscopio TEKTRONIK TDS 1002 es un dispositivo de medición digital presente en el laboratorio de GranaSAT, al igual que HAMEG HMO1024. Sus principales características son:

- Canales: 2
- Ancho de banda: 60 MHz
- Frecuencia de muestreo: 1 GS/s (giga muestras por segundo)
- Profundidad de memoria: 2.5 k puntos por canal
- Pantalla: LCD de 5.7 pulgadas en blanco y negro
- Conectividad: Puertos USB para almacenamiento de datos y conexión a PC

El osciloscopio HAMEG HMO1024 es un dispositivo más avanzado con características que lo hacen adecuado para aplicaciones más exigentes. En concreto, este fué el usado en el capítulo 2, para comprobar que los buses SPI, tenían la frecuencia suficiente para comunicarnos con ellos desde el ADC. Algunas de sus principales características son:

- Canales: 4
- Ancho de banda: 200 MHz
- Frecuencia de muestreo: 2 GS/s
- Profundidad de memoria: 1 M puntos por canal
- Pantalla: LCD a color de 6.5 pulgadas
- Conectividad: Puertos USB, LAN y opcional GPIB

A continuación, se presenta una lista de las diferencias más destacadas entre los dos osciloscopios:

- Canales: El TDS 1002 tiene 2 canales, mientras que el HMO1024 tiene 4 canales.
- Ancho de banda: El TDS 1002 tiene un ancho de banda de 60 MHz, mientras que el HMO1024 ofrece 200 MHz.
- Frecuencia de muestreo: El TDS 1002 tiene una frecuencia de muestreo de 1 GS/s, en comparación con los 2 GS/s del HMO1024.
- Profundidad de memoria: El TDS 1002 tiene una profundidad de memoria de 2.5 k puntos por canal, mientras que el HMO1024 tiene 1 M puntos por canal.

- Pantalla: El TDS 1002 tiene una pantalla LCD en blanco y negro de 5.7 pulgadas, mientras que el HMO1024 tiene una pantalla LCD a color de 6.5 pulgadas.
- Conectividad: El TDS 1002 cuenta con puertos USB, mientras que el HMO1024 tiene USB, LAN y opcional GPIB.

E.6. Cargador de baterías ALC 8500

El ALC8500 es un cargador de baterías avanzado diseñado para satisfacer las necesidades de carga de una amplia variedad de baterías recargables. Es ideal para aplicaciones en las que se requiere un control preciso y eficiente del proceso de carga, siendo utilizado en sectores como la automoción, electrónica y aplicaciones industriales. En mi caso ha sido usado para conocer el estado de la batería y obtener las curvas de carga y descarga que se ven en la sección 3.5.

E.6.1. Características destacadas

- Compatibilidad de baterías: Capaz de cargar baterías de NiMH, NiCd, Li-Ion, LiPo, y Pb.
- Capacidad de carga: Admite baterías con capacidad desde 100 mAh hasta 10,000 mAh.
- Corriente de carga: Ajustable entre 0.1 A y 10 A.
- Pantalla: Pantalla LCD para la visualización de parámetros como voltaje, corriente, tiempo de carga, y estado de la batería.
- Modos de carga: Múltiples modos de carga, incluyendo carga rápida, carga balanceada, y ciclo de descarga-recarga.
- Funciones de seguridad: Protección contra sobrecarga, sobrecalentamiento, cortocircuito y polaridad inversa.
- Conectividad: Puertos USB para actualización de firmware y conexión a PC para monitoreo en tiempo real.
- Enfriamiento: Sistema de enfriamiento activo con ventilador controlado por temperatura para evitar el sobrecalentamiento durante el uso intensivo.

Figura E.2 – *Meter view de Power Analyzer N6705A.* ??

Figura E.3 – *Data logger view de Power Analyzer N6705A.* ??

Figura E.4 – *Módulos de Power Analyzer N6705A. [16]*

Figura E.5 – *Multímetro digital Siglent SDM3065X de los laboratorios de GranaSAT*

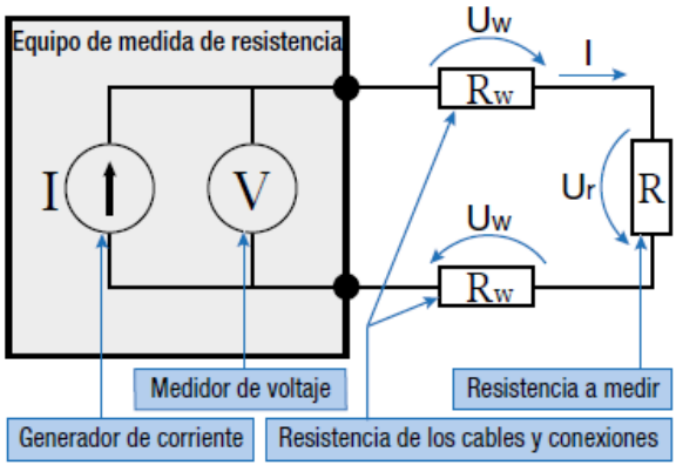


Figura E.6 – *Esquemático con dos cables de medición con multímetro. [17]*

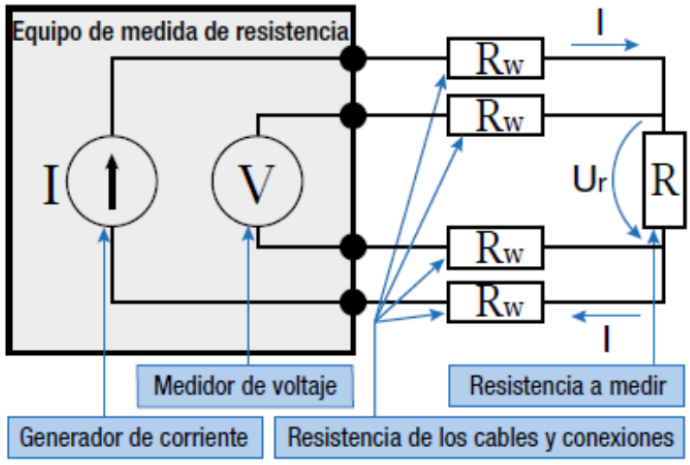


Figura E.7 – *Esquemático con cuatro cables con multímetro. [17]*

Figura E.8 – *Fuente de tensión FAC 363 B de los laboratorios de GranaSAT*

Figura E.9 – *Fuente de corriente Keithley 220 de los laboratorios de GranaSAT*

Figura E.10 – *Osciloscopio TEKTRONIC TDS 1002*

Figura E.11 – *OsciloscopiocHAMEG HMO1024*

Figura E.12 – *Cargados y descargador de baterías ALC8500 de los laboratorios de GranaSAT*

Apéndice F

Presentación sobre RTOS

A continuación se incluyen las diapositivas usadas para realizar una presentación del mencionado sistema operativo. Este sistema operativo es muy novedoso y en mi experiencia en ningún grado de la UGR se imparte materia sobre él. El tutor Andrés, me pidió que explicase a los compañeros del laboratorio de GranaSAT que explicase en que consiste y como resultado está la mencionada exposición.

INTRODUCTION TO RTOS

A brief introduction to RTOS as part of my final degree project at TIK Group Development.

By Juan Manuel Martín Lucena

Student of the industrial electronic engineering at the University of Granada 2024

Contents

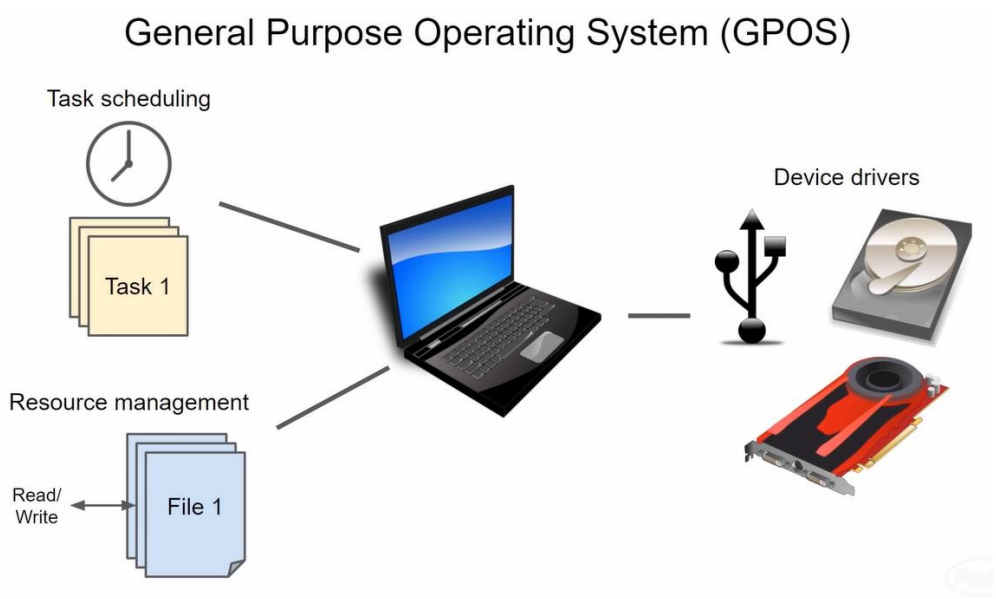
1. What is Real-Time Operating System (RTOS) ?
2. Getting started with FreeRTOS
3. Task scheduling
4. Memory Management
5. Queue
6. Mutex
7. Semaphore

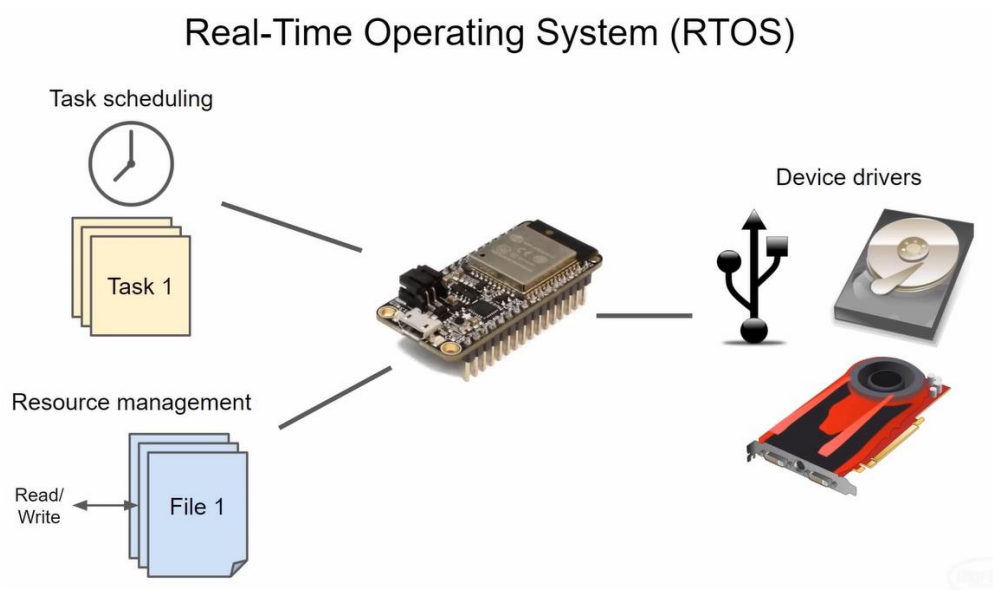
03/04/2024

8. Software timer
9. Hardware interrupts
10. Deadlocks and Starvation
11. Priority inversion
12. Multicore Systems

??

1. What is Real-Time Operating System (RTOS)?





The basic difference of using a **GPOS** or an **RTOS** lies in the nature of the system i.e whether the system is ***time critical*** or not.

RTOS tend to be lighter in size and complexity and ***deterministic*** while **GPOS** is ***no deterministic*** and experiences variable latencies due to background processes and system services.

1. RTOS utilities

```
## Starting application at 0x4010100000 ...

 _________            _________
 \77777777\          /77777777/
  \77777777\        /77777777/
   \77777777\      /77777777/
    \77777777\    /77777777/
     \77777777\   \7777777/
      \77777777\   \77777/              VxWorks 7 SMP 64-bit
       \77777777\   \777/
        \77777777\   \7/       Core Kernel version: 1.0.0.0
         \77777777\   -        Build date: May 30 2014 10:51:05
          \77777777\
           \7777777/           Copyright Wind River Systems, Inc.
            \77777/   -                   1984-2014
             \777/   /7\
              \7/   /777\
               -   -------

                   Board: Wind River Dev Kit MP8
               CPU Count: 8
          OS Memory Size: 1899MB
        ED&R Policy Mode: Deployed


Adding 5290 symbols for standalone.

[vxWorks]# i

  NAME         TID          PRI   STATUS        PC          ERRNO  CPU #
---------- ---------------- --- ---------- ---------------- ------- -----
tJobTask    40104cdbc0        0 PEND       401020c83c             0     -
tExcTask    40102a073c        0 PEND       401020c83c             0     -
tLogTask    40104d01d8        0 PEND       401020b0f0             0     -
tShell0     40105c1d30        1 READY      4010215e08             0     0
ipcom_tick> 401057a990       20 PEND       401020c83c             0     -
tVxdbgTask  401057dc20       25 PEND       401020c83c             0     -
tNet0       40104d3b78       50 PEND       401020c2b4             0     -
ipcom_sysl> 40104c9810       50 PEND       401020d3d4             0     -
tNetConf    40105a6e40       50 PEND       401020c83c             0     -
miiBusMoni> 40104d5e08      252 DELAY      4010215640             0     -
ipcom_egd   4010583c20      255 DELAY      4010215640             0     -
tIdleTask0  40102a2fb0      287 READY      401020c004             0     -
tIdleTask1  40102a7220      287 READY      401020c00c             0     1
tIdleTask2  40102ab490      287 READY      401020c004             0     2
tIdleTask3  40102afb20      287 READY      401020c004             0     3
tIdleTask4  40102b1700      287 READY      401020c004             0     4
tIdleTask5  40102b2440      287 READY      401020c004             0     5
tIdleTask6  40102a4620      287 READY      401020c004             0     6
tIdleTask7  40102a4860      287 READY      401020c004             0     7
[vxWorks]#
```

RAD750

The RAD750

General information	
Launched	2001
Designed by	IBM
Common manufacturer(s)	BAE
Performance	
Max. CPU clock rate	110 MHz to 200 MHz
Cache	
L1 cache	32 KB instruction + 32 KB data
Architecture and classification	
Application	Radiation-hardened
Technology node	250 nm to 150 nm
Microarchitecture	PowerPC 750
Instruction set	PowerPC v.1.1
Physical specifications	
Cores	1

Rover Curiosity, Launched on November 26, 2011

Developed by Wind River System for embedded systems with fast responses in the order of microseconds.

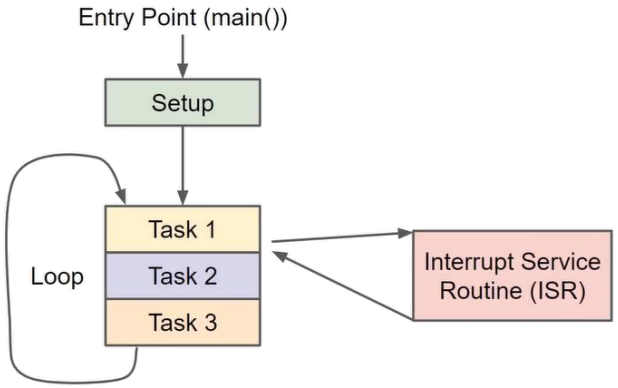
Customer adaptable to 32-bit and 64-bit CPUs and manufacturers such as Intel on ARM architectures or IBM with their respective Power PC architecture.

The hardware used for example is the RAD750 V2 which operates at MHz and is powered by 1.9/3.3V and has 10.4 million transistors, a performance that is surpassed by a mobile processor such as the A13, however, these processors demonstrate great **reliability.**

In the curiosity where the RAD750 is implemented, it is doubly implemented to always ensure the operation of the on-board computer.

2. Getting started with FreeRTOS

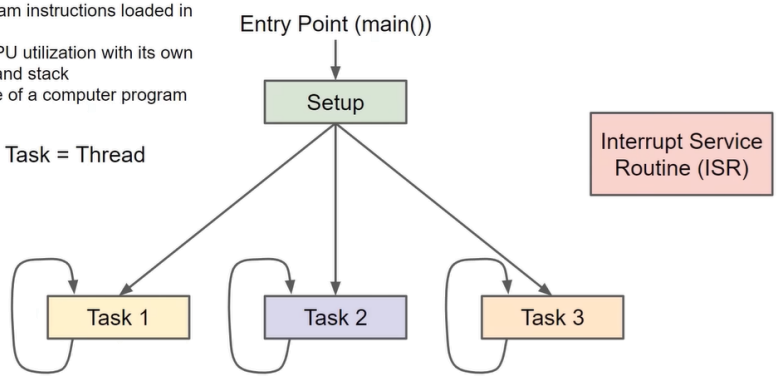
Super Loop



RTOS

- **Task**: set of program instructions loaded in memory
- **Thread**: unit of CPU utilization with its own program counter and stack
- **Process**: instance of a computer program

FreeRTOS: Task = Thread



In the super Loop we have the great disadvantage of **not moving to a subsequent task without first passing through the previous one,** while in RTOS we have the ability to **run n tasks with n cores at the same time.**

In RTOS when we find an **ISR** all tasks are interrupted without continuing until it is solved, storing the state of that task when it has been **interrupted to later continue from the point where the execution was left (TCB).**

2. Getting started with FreeRTOS in ESP32

```
xTaskCreatePinnedToCore(
                    Task1code,   /* Task function. */
                    "Task1",     /* name of task. */
                    10000,       /* Stack size of task */
                    NULL,        /* parameter of the task */
                    1,           /* priority of the task */
                    &Task1,      /* Task handle to keep track of created task */
                    0);          /* pin task to core 0 */
```

As first function of the freeRTOS library, I introduce this one with which we model the characteristics of the task to use. In it **we fix a task to a core**, however, we can also **not give affinity to the task and it can be executed in any core.**

It stands out stack size in number of bytes that at most can be **520 kB and must be taken into account when handling pointers.**

- The priority ranges **from 0 with the lowest priority to 24 with the highest priority.**
- As pointers we find the task parameter which is added **if we want to pass a pointer to work with in our external task**, and **the second is the handler that we use in the task.**
- NULL means that we do not pass anything

ESP-WROOM-32

- 240 MHz (dual core)
- 4 MB flash
- 520 kB RAM

2. Tasks behaviour

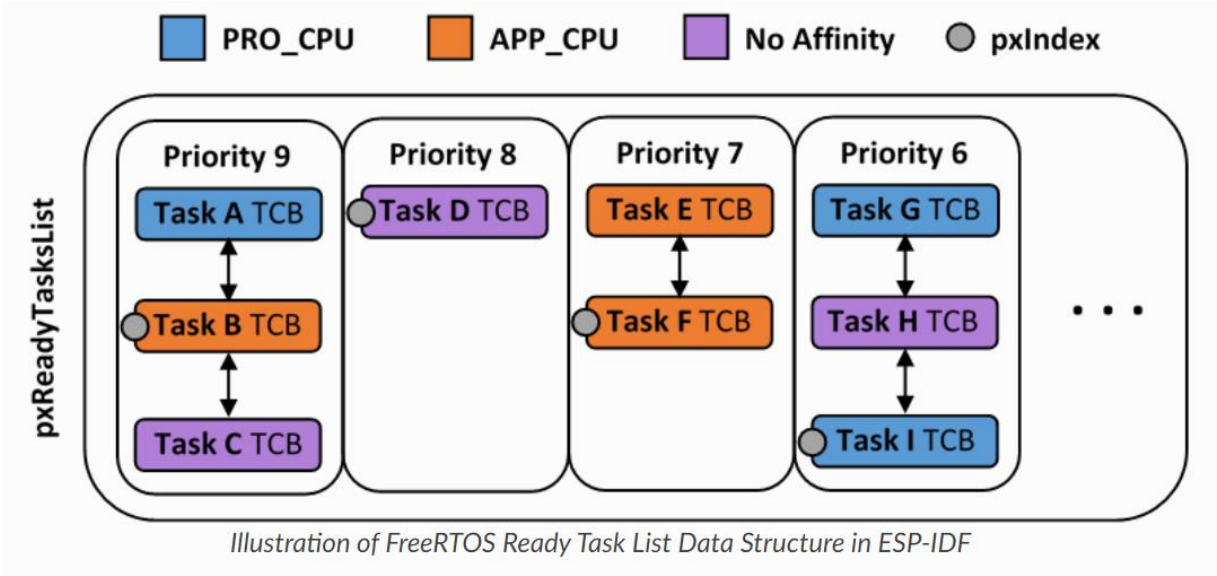


Illustration of FreeRTOS Ready Task List Data Structure in ESP-IDF

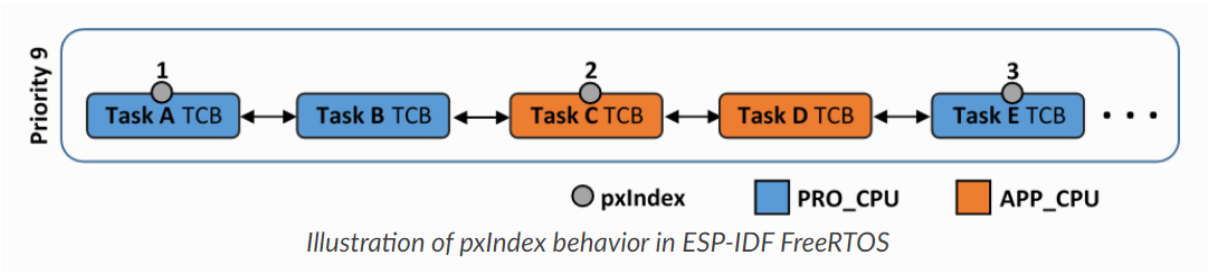
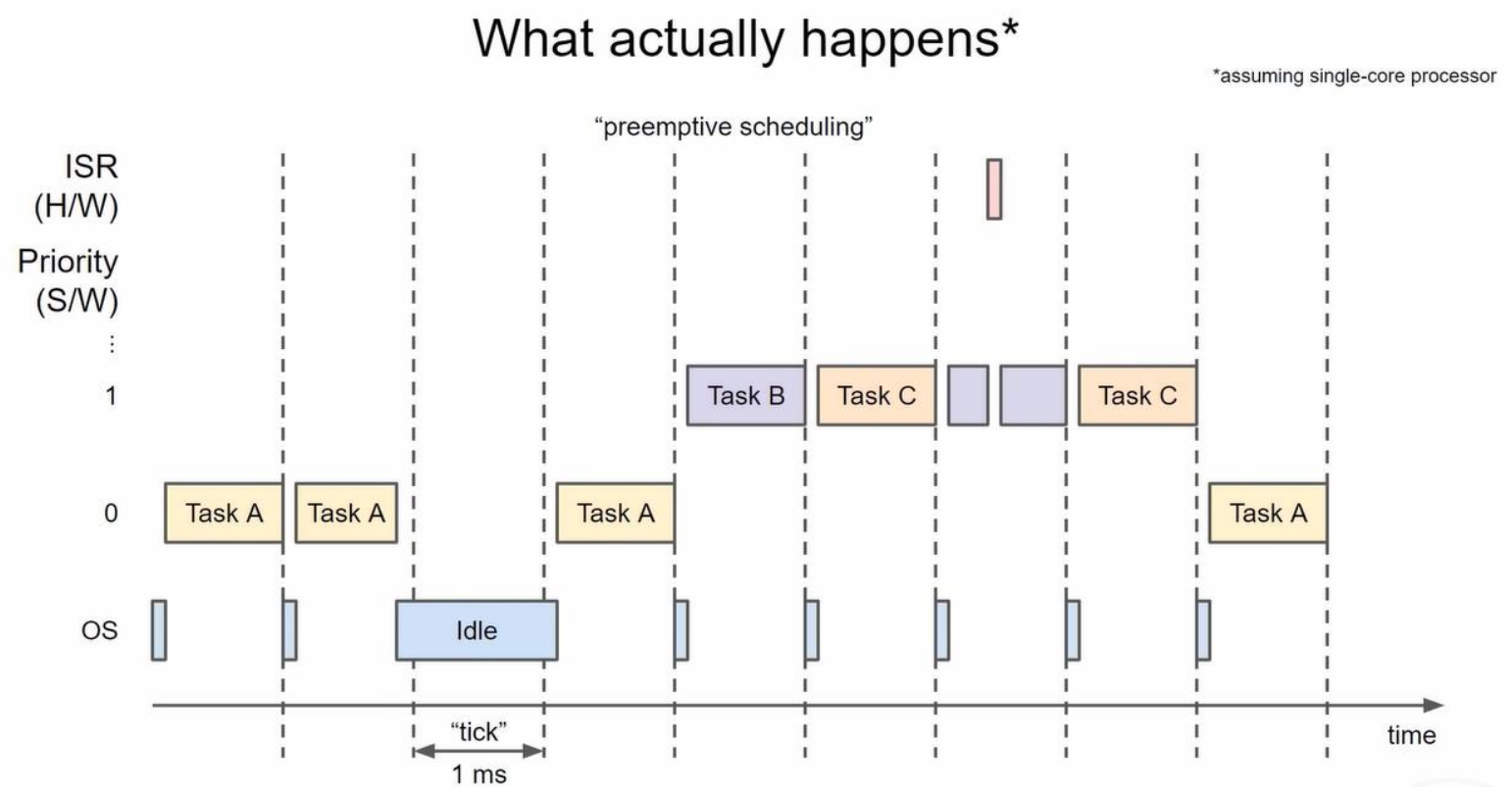


Illustration of pxIndex behavior in ESP-IDF FreeRTOS

With the difference of colours we can see to which cores have been assigned the different tasks and even not define core and that is located according to the priority.

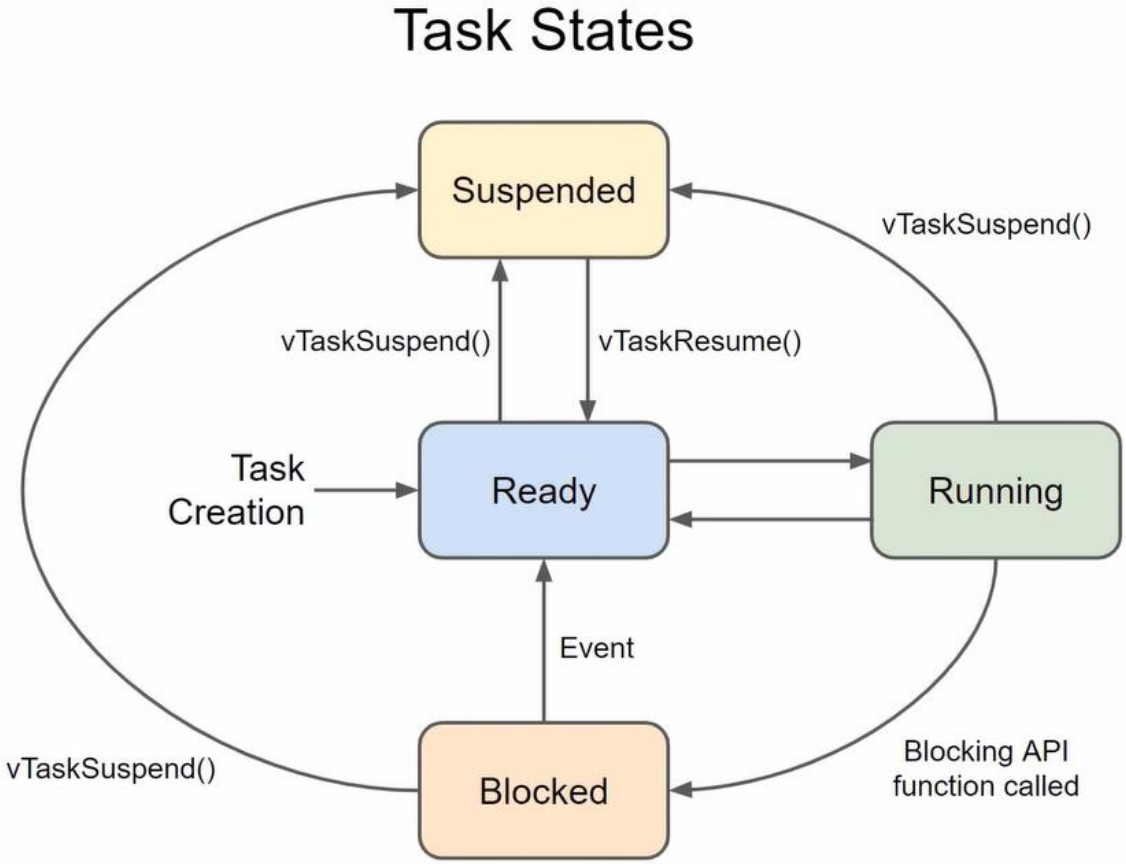
The **pxIndex determines the order of execution of the tasks** and in the illustration below we see for the same priority**, if we find two tasks of the same core consecutively we have jump** of the tasks B and D since freeRTOS works like this.

3. Task scheduling



When we **have a task with a higher priority such as tasks B and C with respect to A**, the processor executes them consecutively. When an ISR is encountered, **no task is executed until the ISR is solved.** In FreeRTOS, the default time slice is 1 ms, and a time slice is known as a "tick." A hardware timer is configured to create an interrupt every 1 ms.

3. Task states



Tasks have states that we manipulate at our convenience and **tell the processor how to work with them.**

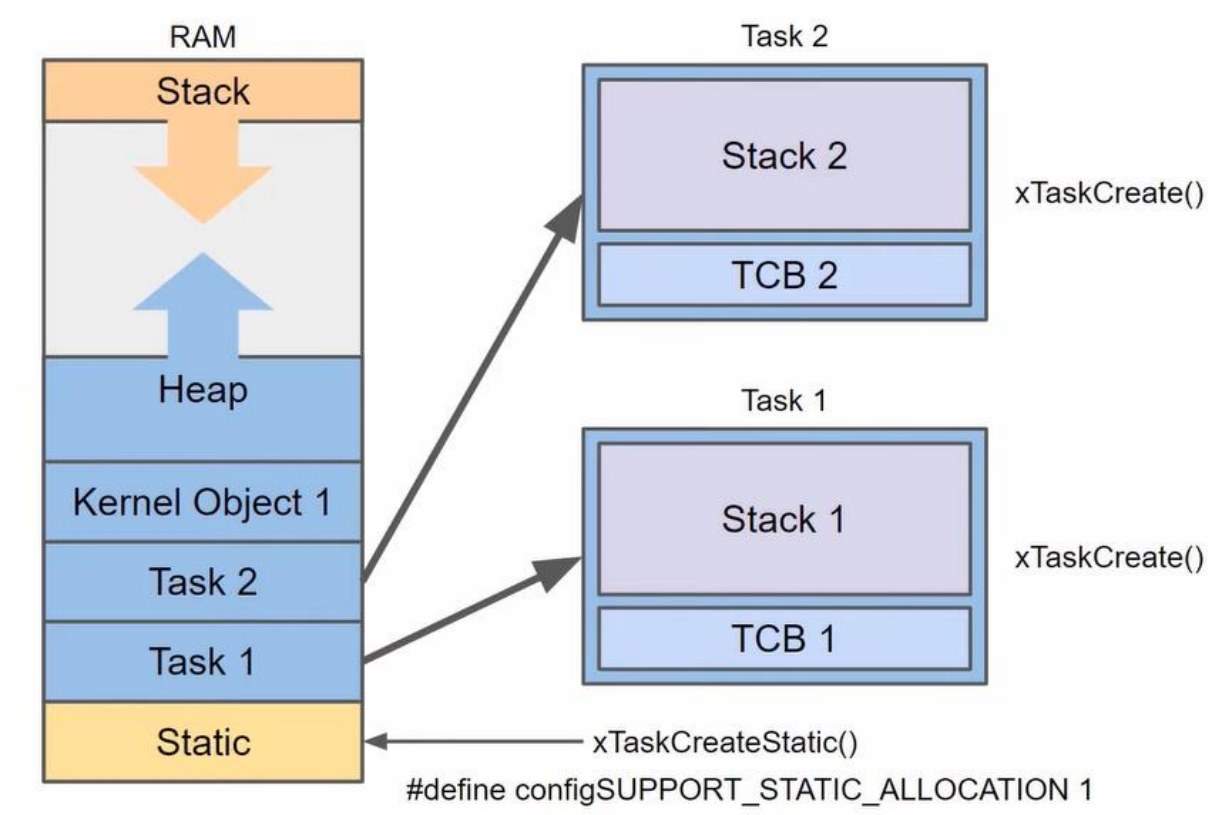
When we create a task**, by default it is placed in the Ready state.** When the priority allows it, it goes to running.

When **a task is actually executing** it is said to be in the **Running state**. It is currently utilising the processor. If the processor on which the RTOS is running only has a single core then there can only be one task in the Running state at any given time.

The **Suspend state** is a **control by API calls** which we manually manipulate to move it to ready.

The **Blocked state** is only accessed via a delay function **vTaskDelay() and automatically goes to ready or we indicate its suspension.** Tasks can also block to wait for queue, semaphore, event group, notification or semaphore event

4. RTOS Memory Management



We use **RAM** memory because it is the non-volatile memory that allows fast access (not DMA).

This memory is divided into automatic, dynamic and static memory. **The tasks in freeRTOS use the dynamic memory known as heap,** where inside we have stack to store variables and the **TCB that contains information about the state of the task.** In general, the pointers are the ones stored in the heap **and if they are not released** there can be overflow **with the automatic stack** where global variables are stored.

Finally, we have the **static stack whose storage does not vary** and is used in case of storing **variables with vital information.**

4. Memory management: Example 1 (a)

```
1   //Usamos solo un nucleo para la demo
2   #if CONFIG_FREERTOS_UNICORE
3     static const BaseType_t app_cpu = 0;
4   #else
5     static const BaseType_t app_cpu = 1;
6   #endif
7
8   void testTask(void *parameter){
9     while (1)
10    {
11      int a = 1;
12      int b[100]; //Matriz de 100 elementos
13
14      for (int i = 0; i<100; i++) {
15        b[i] = a + 1;
16      }
17      Serial.println(b[0]); //Leemos la primera posición por ejemplo
```

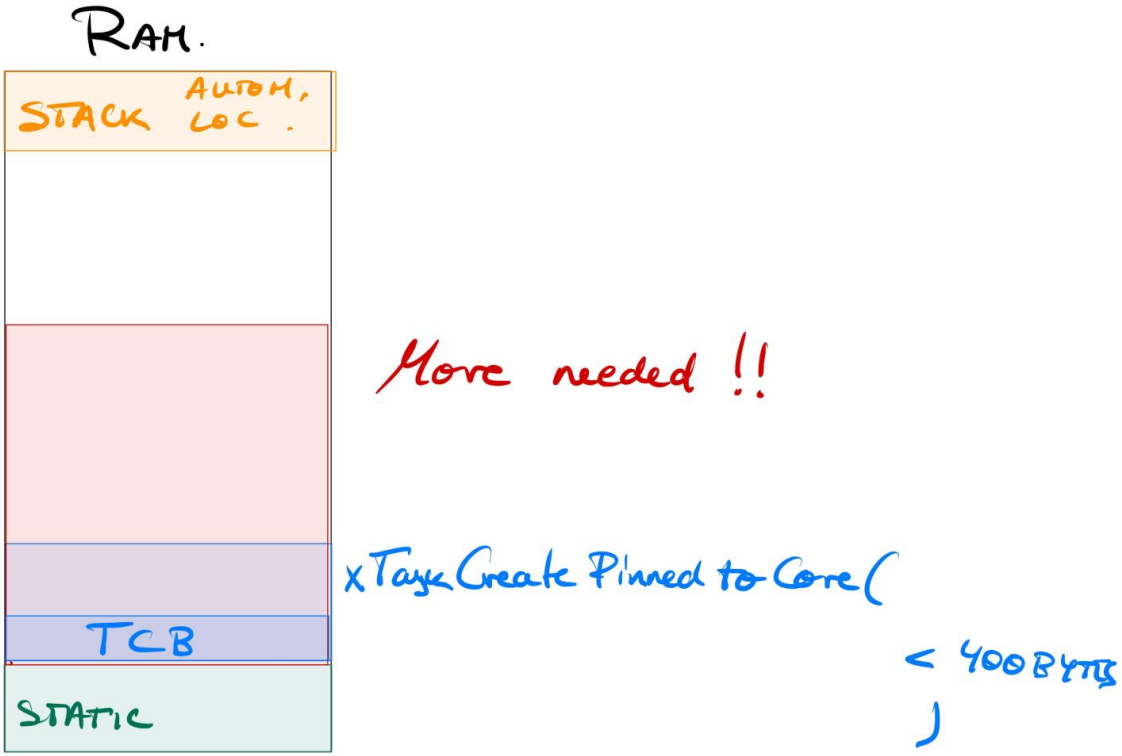
```
53  void setup(){
54    Serial.begin(115200);
55    vTaskDelay(1000 / portTICK_PERIOD_MS);
56    Serial.println();
57    Serial.println("---FreeRTOS Memory Demo---");
58
59    xTaskCreatePinnedToCore(testTask,
60                            "Test Task",
61                            400,
62                            NULL,
63                            1,
64                            NULL,
65                            app_cpu);
66  }
67
68  void loop(){
69  }
```

In this code we create an array of 100 elements where we store 400 integers which would mean that **we need 400 bytes no?**

```
---FreeRTOS Memory Demo---

***ERROR*** A stack overflow in task Test Task has been detected.


Backtrace: 0x4008340d:0x3ffb8680 0x40088035:0x3ffb86a0 0x4008af05:0x3ffb86c0
0x40089933:0x3ffb8740 0x40088144:0x3ffb8770 0x400880f4:0x00000000 |<-CORRUPTED
```

There is not enough space allocated for this task !

4. Memory management: Example 1 (a)



What is happening is that **we are requiring more storage than is allocated to perform that task, therefore we are exceeding the space and we can overwrite other tasks** and have serious memory problems. Therefore, working with **dynamic memories is very delicate.**

Nor does it help us as we see only 400 bytes of storage since the **TCB also requires space** and therefore we must allocate the memory space freely.

4. Memory management: Explain example 1 (b)

```
23  //DEvuelve los bytes libres del heap antes de asignar un almacenamiento dinámico en el HEAP
24  Serial.print("Heap antes del almacenamiento del puntero (bytes)");
25  Serial.println(xPortGetFreeHeapSize());
26
27  //Asigna un bloque de memoria en el Heap de 1024 bytes por 4 bytes de cada entero
28  int *ptr = (int*)pvPortMalloc(1024 * sizeof(int));
29
30  //Evitar el desbordamiento del HEAP por el almacenamiento del puntero
31  if (ptr==NULL){
32    Serial.print("No hay espacio de almacenamiento en HEAP");
33  }
34
35  else {
36    //Hacemos algo con la memoria dinámica asignada, por ejemplo:
37    for(int i = 0; i<100; i++){
38      ptr[i] = 1;
39    }
40  }
41
42  //Memoria libre despues de la asigancion dinámica
43  Serial.print("Heap después del almacenamiento (bytes)");
44  Serial.println(xPortGetFreeHeapSize());
```

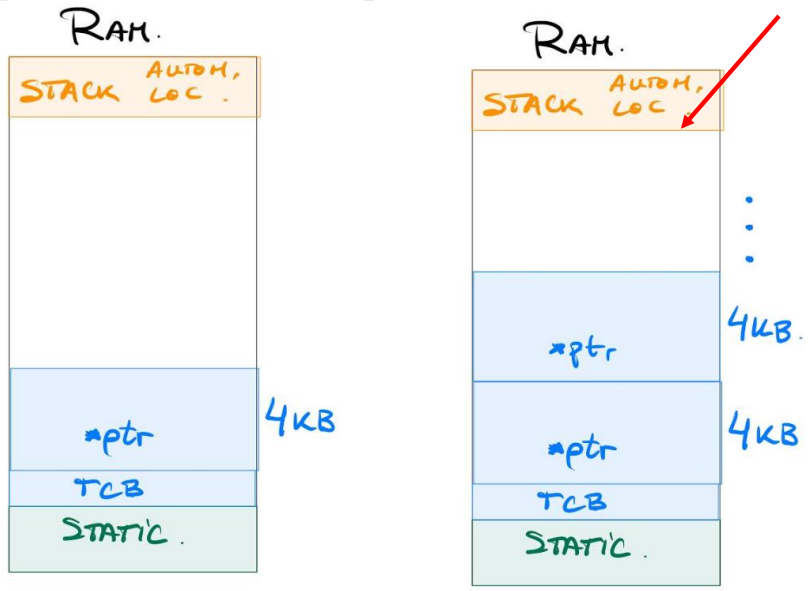
```
xTaskCreatePinnedToCore(testTask,
                        "Test Task",
                        1024*5,
                        NULL,
                        1,
                        NULL,
                        app_cpu);
```

We increase the memory up to 5 kB therefore every time the code is executed, we will not have problems due to not having given enough space to execute this task.

We now introduce a pointer with a space of 4 kB (1024 bytes * 4 bytes each integer) and see what happens if it is not freed.

4. Memory management: Explain example 1 (b)

Every time the code is executed, we see how the **free heap space decreases.** If you have enough time, there **will be a moment when we have a memory overflow** and end up having overwriting problems.



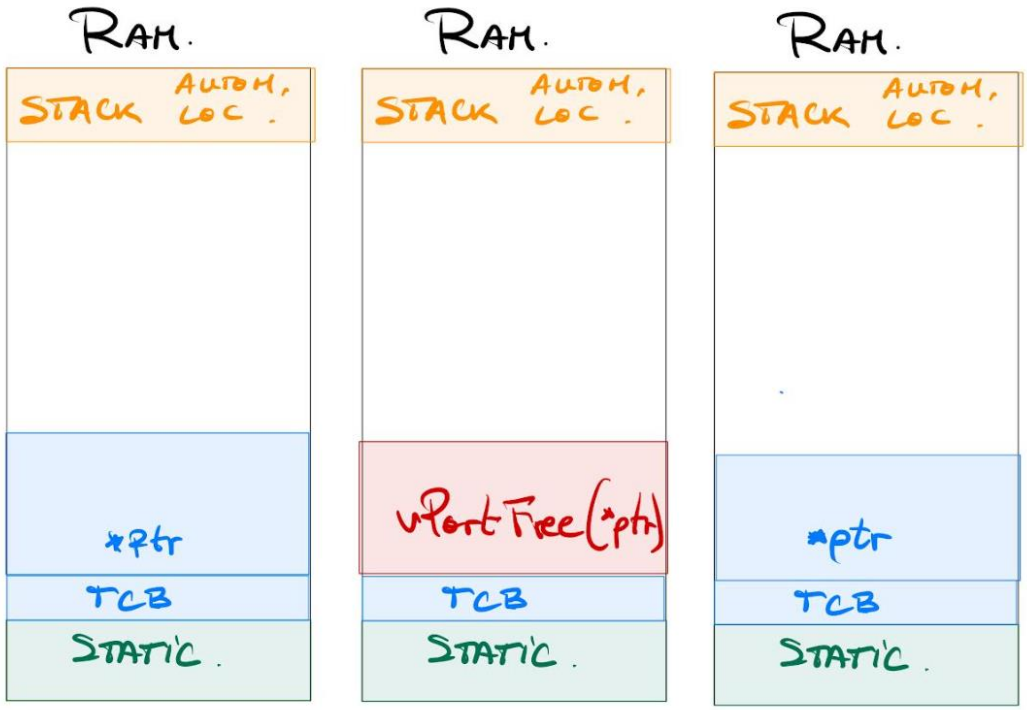

```
---FreeRTOS Memory Demo---
2
Heap antes del almacenamiento del puntero (bytes)273856
Heap después del almacenamiento (bytes)269744
2
Heap antes del almacenamiento del puntero (bytes)269744
Heap después del almacenamiento (bytes)265632
2
Heap antes del almacenamiento del puntero (bytes)265632
Heap después del almacenamiento (bytes)261520
```

273856 – 269744 ≅ 4kB

4. Memory management: Example 1 (b)

```
47
48      vPortFree(ptr); //Liberamos la memoria dinámica después de cada uso
49  }
```

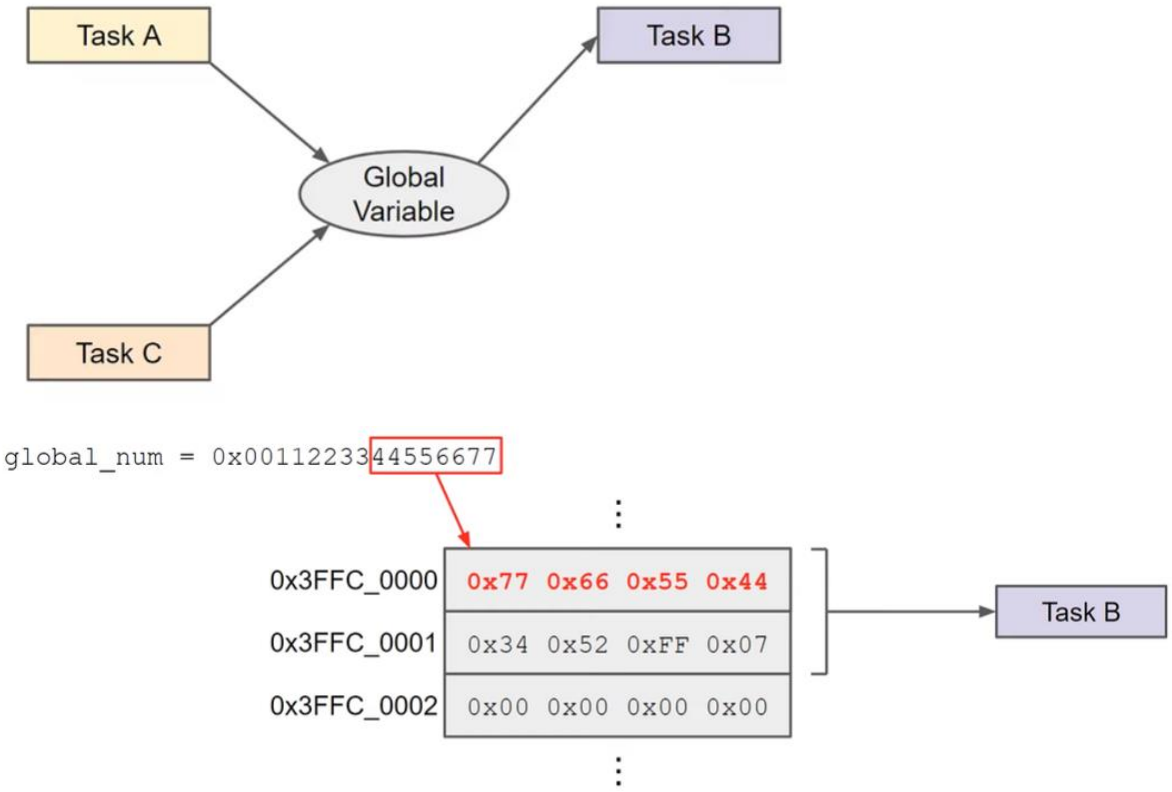
When we include the function to release the assigned pointer at the end of the code, **we do not have problems with unnecessary memory leaks.**




```
---FreeRTOS Memory Demo---
2
Heap antes del almacenamiento del puntero (bytes)273736
Heap después del almacenamiento (bytes)269624
2
Heap antes del almacenamiento del puntero (bytes)273736
Heap después del almacenamiento (bytes)269624
```

As we see we have the same space before and after using the pointer for our task.

5. Queue

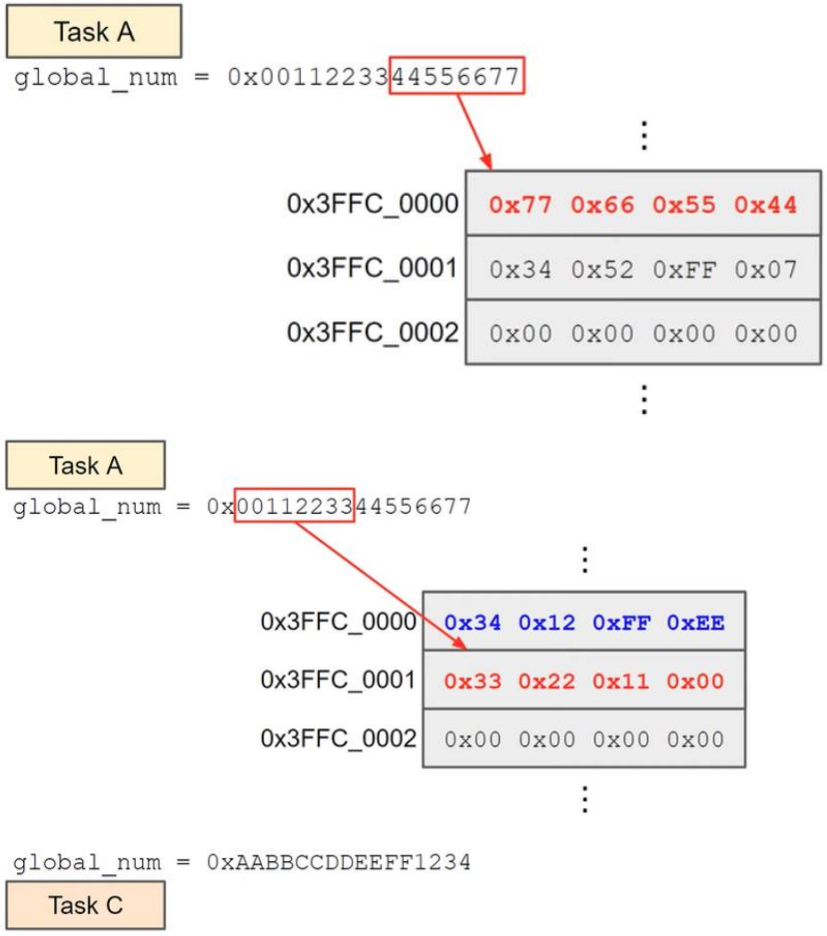


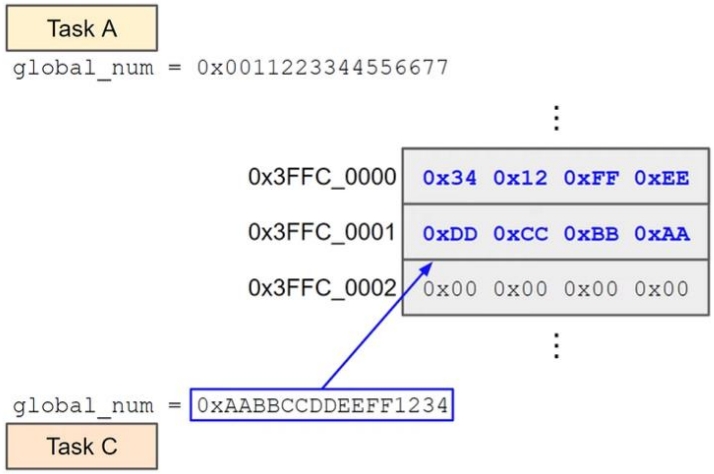
A global variable can be written and read by several tasks, and this can result, in this case, **task A and C modifying the variable and task B reading it incompletely.**

In the case of the ESP32 SoC (system on chip) chip we have **Little-endian (appendix A)** in which 32-bit data is stored in 32 bits and therefore a 64-bit number needs to be divided by 2.

The first half would be stored, and the **second half would not be stored,** since each integer needs 4 bits and if we multiply there are 8 integers, therefore 32 bits.

5. Queue

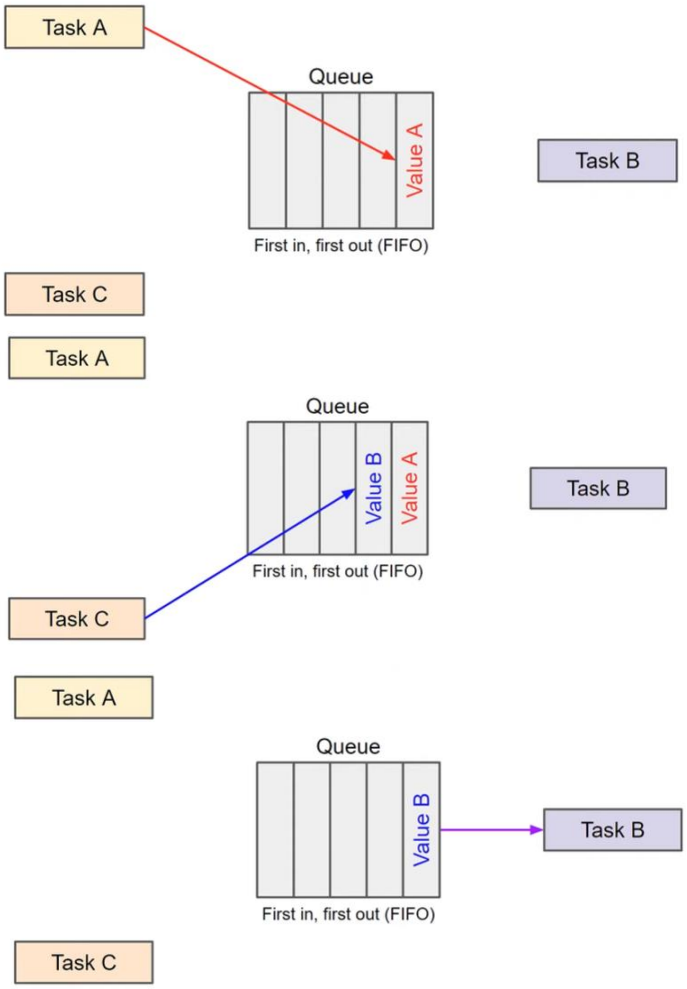




In this case we see **how the outputs of the tasks are mixed.** **Task A** writes and without task C finishing, **it starts writing** its output in the global variable that will later be read by task B **with altered information.**

To avoid this, i am going to discuss **the mutex and semaphore functions.**

5. Queue



We are looking to pass a queue where messages are not interrupted like passing a note in school.

The operation of receiving and taking values follows the philosophy that the **first value that arrives is the first to leave** and therefore as **a handicap we cannot read values that have been stored recently if there are others that have arrived previously** and have priority to be read.

We specify how many values can be stored in the queue, as well as when we want to send and receive value from the queue,

We may encounter a read **error if there are no more values in the queue** and we are afraid of read requests.

5. Queue: Example 1 (a)

We define the length of the list and the handler needed for the queue to handle the data.

```
//Settings
static const int msg_queue_len = 5; //Máximo números de valores que la pila puede contener

//Manejador de cola en esencia un puntero que usamos cada vez que enviamos o recibimos en la cola un valor nuevo
static QueueHandle_t msg_queue;
```

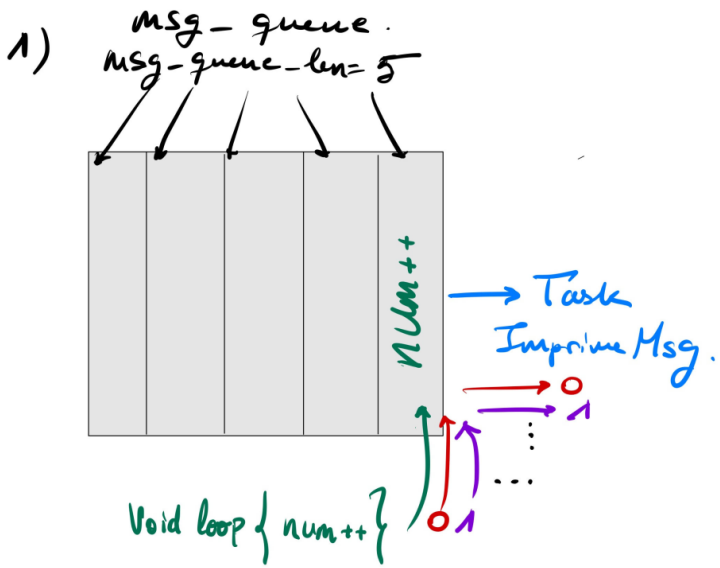
We create a function with higher priority which prints (if possible after one attempt in a loop) **the number sent from the number_increment function.**

```
void ImprimeMsg(void *parameters){

  int item; //Número que la pila va a almacenar

  while(1){

    if (xQueueReceive(msg_queue, (void*)&item, 0) == pdTRUE) {
      // La función no esperará ni un ciclo de instrucción
      // si no puede realizar la adición del item no la hará
      Serial.println(item);
    }

    vTaskDelay(1000 / portTICK_PERIOD_MS);
  }
}
```

```
void number_increment(){
    static int num = 0;

  //Si por algún motivo hay 1 ticks de intento de enviar el mensaje
  // y no es posible, se envía una señal de llenado.
  if (xQueueSend(msg_queue,(void*)&num, 1) != pdTRUE) {
    Serial.println("Queue full!");
  }
  num++;

  vTaskDelay(1000 / portTICK_PERIOD_MS);
}
```

We create **a function that constantly increments the value of a variable and is sent to the queue** if possible after a cycle attempt, if not possible we print Queue full!

5. Queue: Example 1 (a)




```
---Free RTOS Queue Demo--
0
1
2
3
4
5
6
7
8
9
```

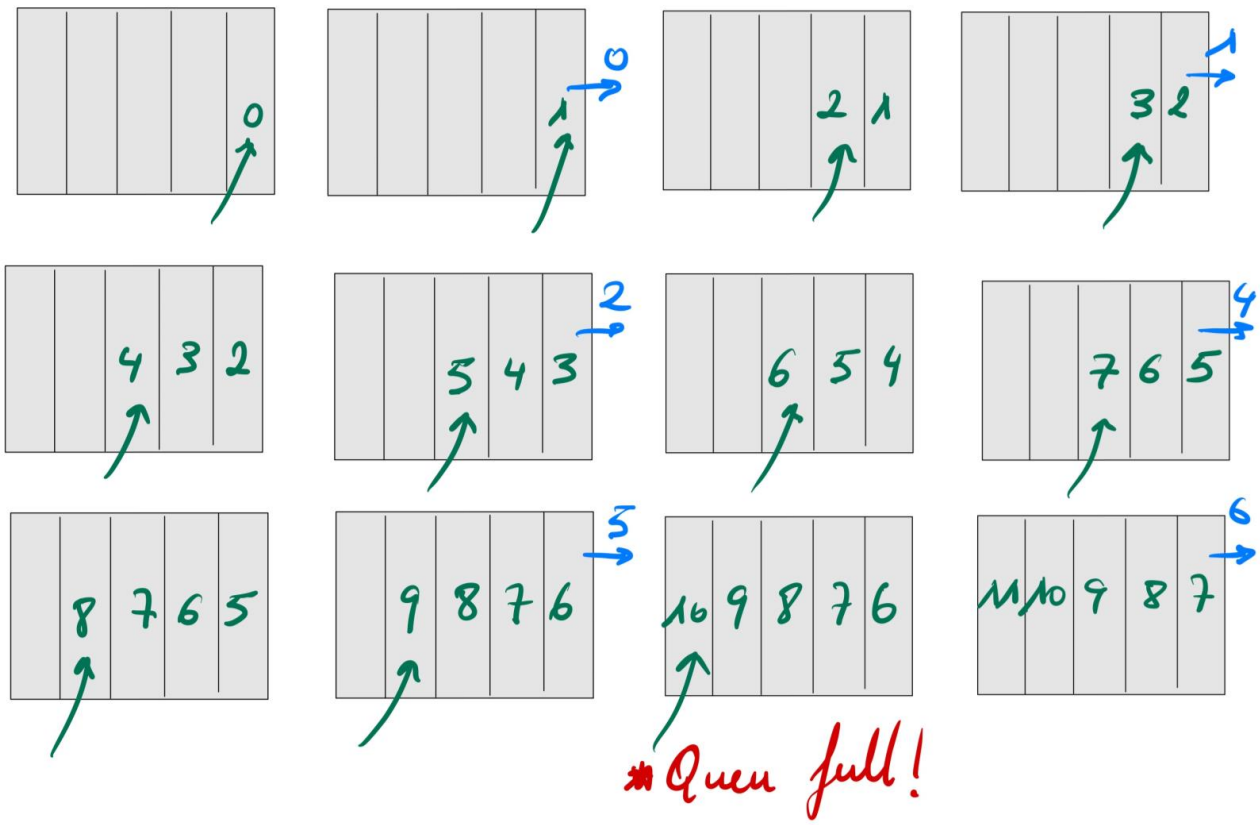
If the waiting time to increment the number and the waiting time to take the number from the queue is the same, we will **not** encounter the error of overfilling the queue and therefore we receive and write values synchronously.

5. Queue: Example 1(b)

```
55  void number_increment(){
56      static int num = 0;
57
58    //Si por algún motivo hay 1 ticks de intento de enviar el mensaje
59    // y no es posible, se envía una señal de llenado.
60    if (xQueueSend(msg_queue,(void*)&num, 1) != pdTRUE) {
61      Serial.println("Queue full!");
62    }
63    num++;
64
65    vTaskDelay(500 / portTICK_PERIOD_MS);
66  }
67
```

Escribimos al doble de lo que leemos!

```
---Free RTOS Queue Demo--
0
1
2
3
4
5
Queue full!
6
```



As we can see we decrease the sending of the incremented number to 500 ms and take the number from the queue after 1000 ms, therefore, as we can see in the figure above, **we are writing at twice what we are able to read and therefore fill the queue in a moment.**

6. Mutex

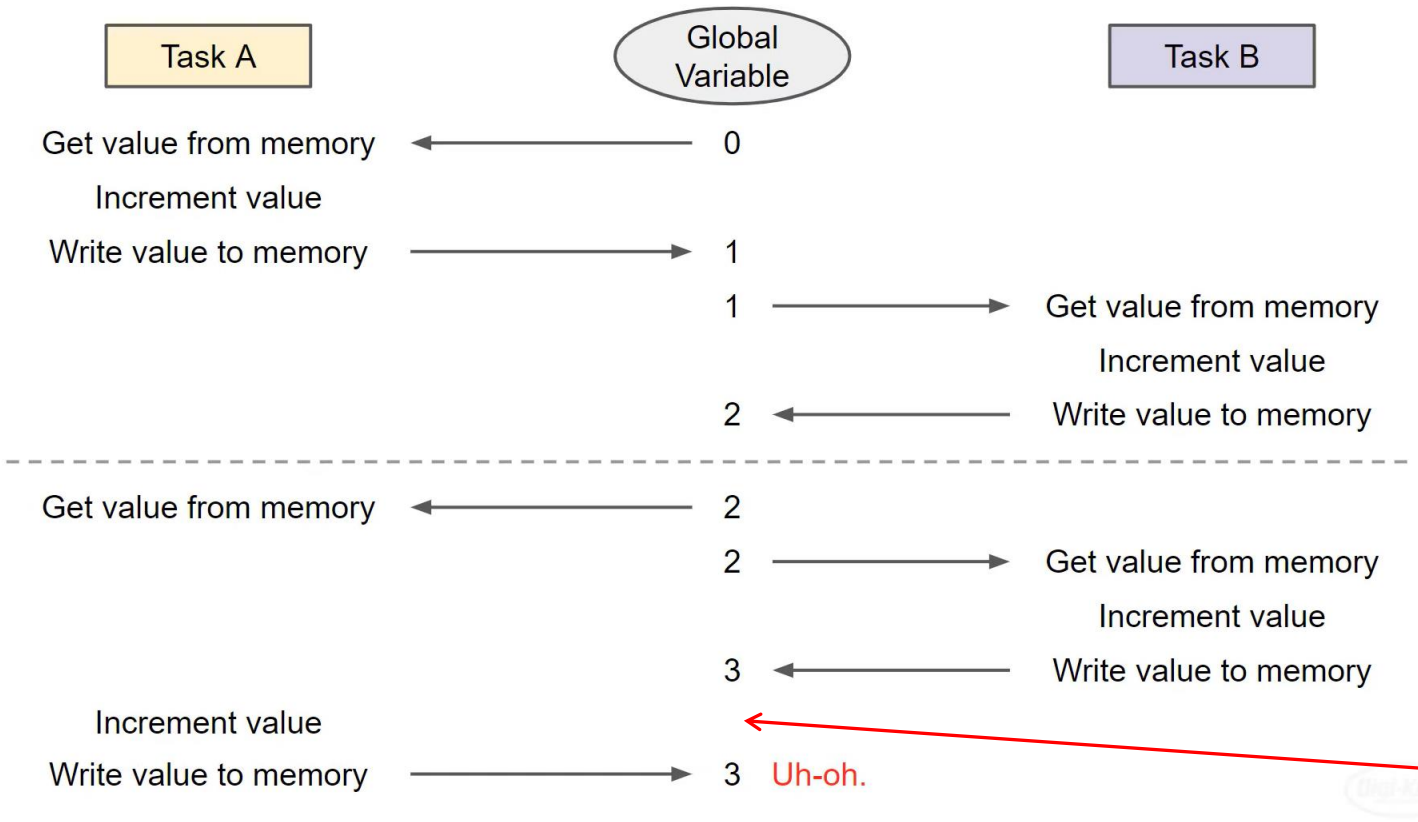
Protecting Shared Resources and Synchronizing Threads

- Queue: pass messages (data) between threads
- Lock: allows only one thread to enter the “locked” section of code
- Mutex (MUTual EXclusion): Like a lock, but system wide (shared by multiple processes)
- Semaphore: allows multiple threads to enter a critical section of code

As we have seen in the handling of data with **concurrent programming using queues, we need some synchronisation in order to have an efficient code.** At this point we will describe the mutual exclusion tools (Mutex) for which we will also describe the semaphores in the use of a shared resource.
This is done **to avoid race conditions** in which performance is lost due to slow performance of a task in modifying a shared resource.

The concept of avoiding such critical sections or race conditions is called mutual exclusion.

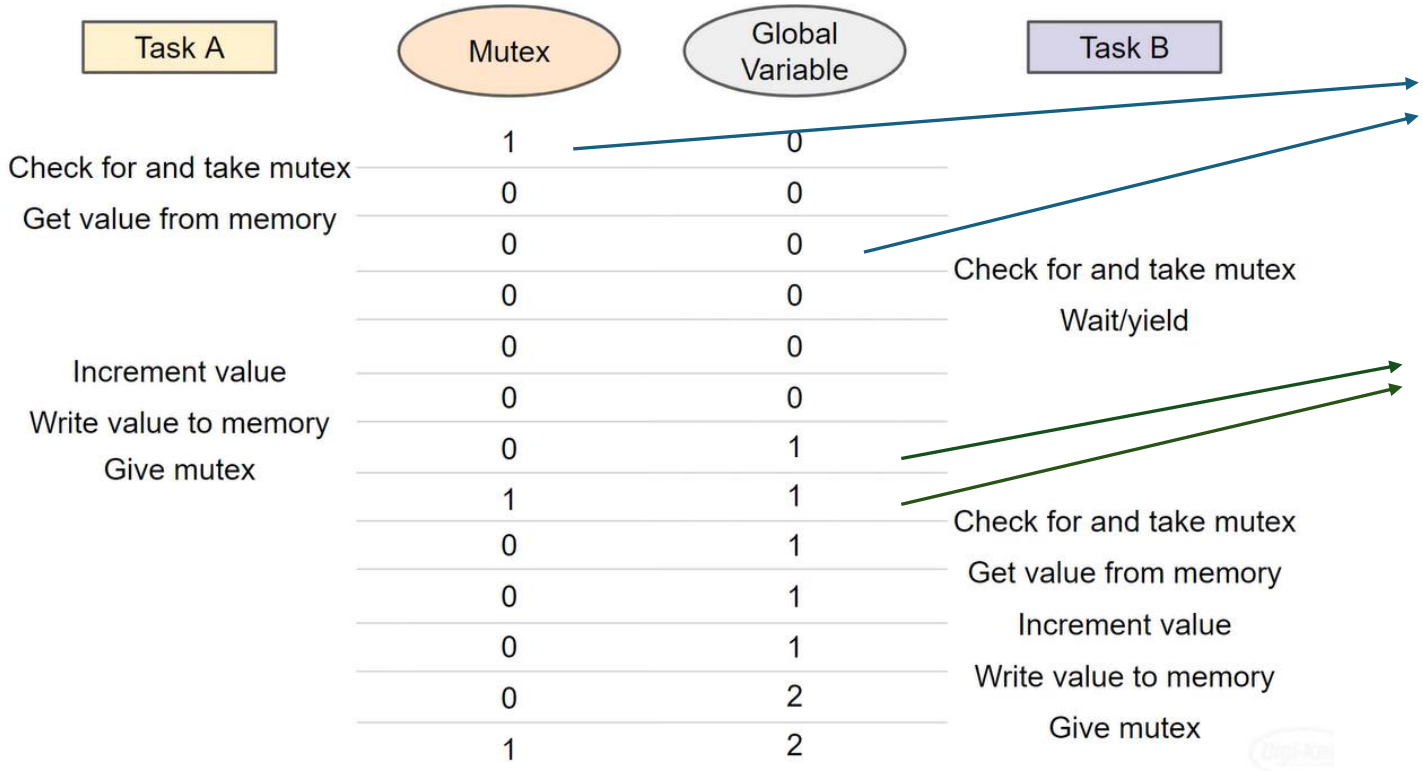
6. Mutex



We find a race condition at the end when two tasks are in charge of incrementing a global variable and at a given time task A receives the value and task B is preempted **before task A returns the value.**

6. Mutex

We create a shared variable (MUTEX) which we can understand as a key in a cafeteria that is on the bar and the customer must take it to go to the toilet. **If the key is not there, it means that someone is in the restroom and until he/she is back in his/her place, he/she cannot go to the restroom.** Therefore, the **users would be the tasks** and the **MUTEX is the boolean variable key** that indicates if it is free or another task is using it.



Before performing the task, it is checked if the mutex is 1, if it is 1, the task is executed (task A), if it is 0, the task waits until it is 1 again (Task B).

When task A, which has taken the mutex, finishes, it returns the mutex and task B, which was waiting for it, takes it and executes the global variable increment.

6. Mutex: Example 1

```
// Globals

// Variable global a la que la tarea va a acceder
static int shared_var = 0;

// Variable de 1 o 0 que cambia de valor cuando la tarea la toma
static SemaphoreHandle_t mutex;
```

In this example, we create the shared variable and the Mutex as globals which will be accessed by the tasks.
We declare two tasks with the same priority that take the mutex and increment the global variable and print the respective message **for each function (Task 1 or 2) and the incremented value of the global variable** to the terminal.

```
//*****************************************************************
// Tasks

// tarea que incrementa una variable local
void incTask1(void *parameters) {

  while (1) {

    //Usamos un mutex creado anteriormente en la pila estática
    if (xSemaphoreTake(mutex, 0) == pdTRUE) {

      // Aumentamos una variable de la tarea y lo igualamos a la global
      shared_var++;

      //Imprimimmos por el terminal
      Serial.println(shared_var);
      Serial.println("Tarea 1");

      // Dejamos libre el mutex para que otra tarea pueda acceder a él.
      xSemaphoreGive(mutex);
    }

    vTaskDelay(1000 / portTICK_PERIOD_MS);
  }
}
```

```
void incTask2(void *parameters) {

  int local_var2;

  while (1) {

    //Usamos un mutex creado anteriormente en la pila estática
    if (xSemaphoreTake(mutex, 0) == pdTRUE) {

      // Aumentamos una variable de la tarea y lo igualamos a la global
      shared_var++;

      //Imprimimmos por el terminal
      Serial.println(shared_var);
      Serial.println("Tarea 2");

      // Dejamos libre el mutex para que otra tarea pueda acceder a él.
      xSemaphoreGive(mutex);
    }

    vTaskDelay(1000 / portTICK_PERIOD_MS);
  }
}
```

```
---FreeRTOS Race Condition Demo---
1
Tarea 1
2
Tarea 2
3
Tarea 1
4
Tarea 2
```

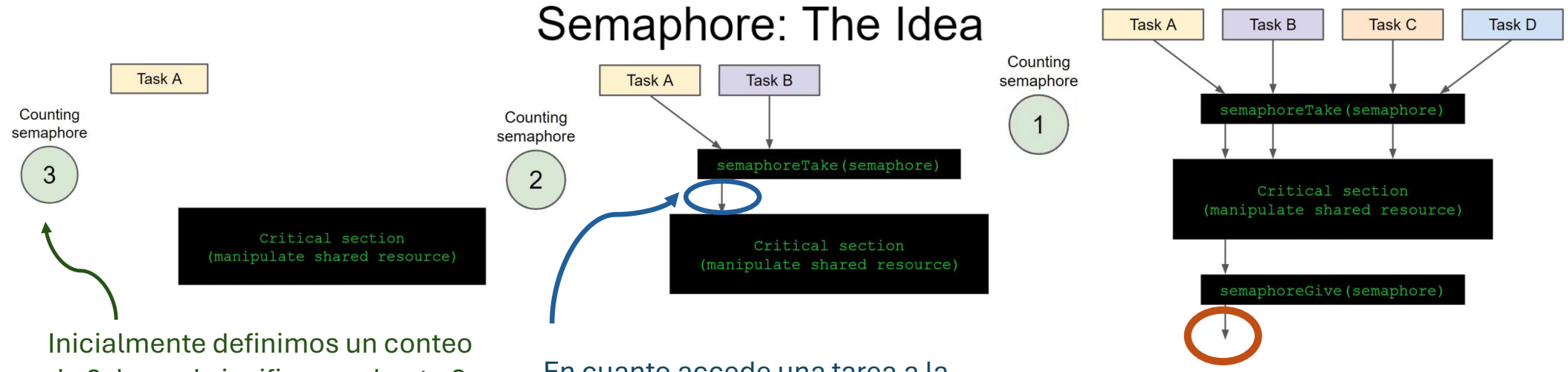
Vemos por el monitor como una y otra tarea se alternan y aumentan la variable global cada vez que una tarea la toma.

7. Semaphores: The Idea

To understand semaphores, we could see it as the analogy above where there **are several keys and several bathrooms, however, we would have the problem that you do not know which bathroom is occupied and which is free before entering with the key** (any key opens any bathroom) so it is not an optimal choice.

Semaphores can take several positive values while **mutexes only use 1 and 0 and therefore**, we could interpret this as a generality with respect to the latter.

The task verifies that the semaphore count is positive and if it is, it accesses the critical section.



Inicialmente definimos un conteo de 3, lo cual significa que hasta 3 tareas distintas pueden acceder a la sección crítica.

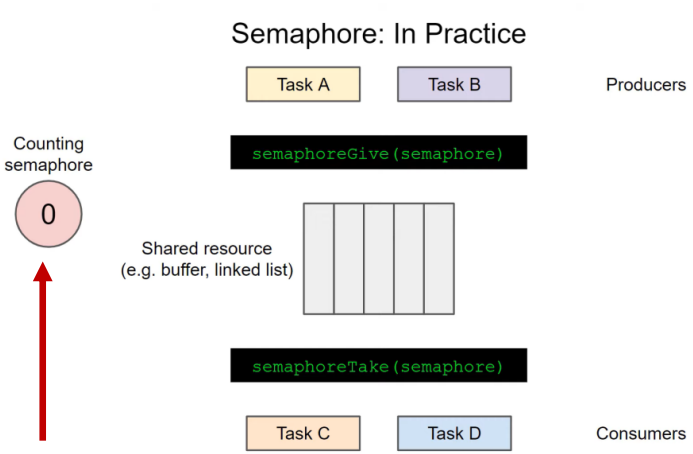
En cuanto accede una tarea a la sección crítica el conteo del semáforo disminuye.

Si tres tareas a la vez acceden, hasta que una no deja el semáforo otra tarea no puede entrar.

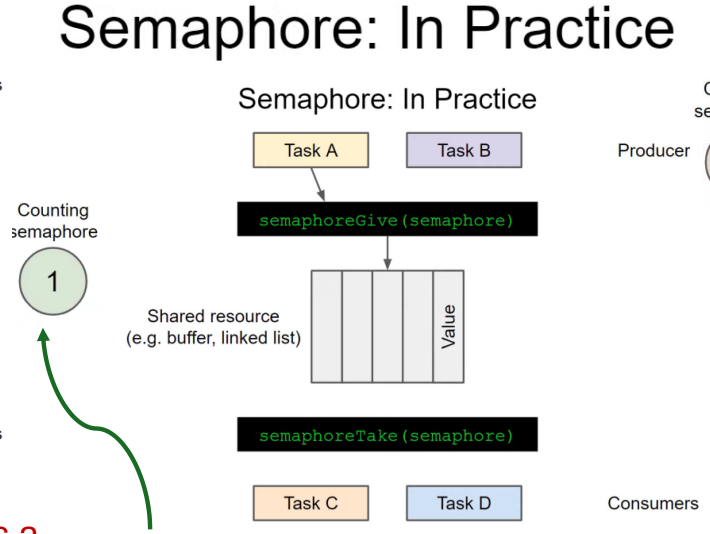
7. Semaphores: In Practice

They are often explained in this way, but it is wrong because we would not be protecting the threads, and **we would not be controlling access to the shared resource.**

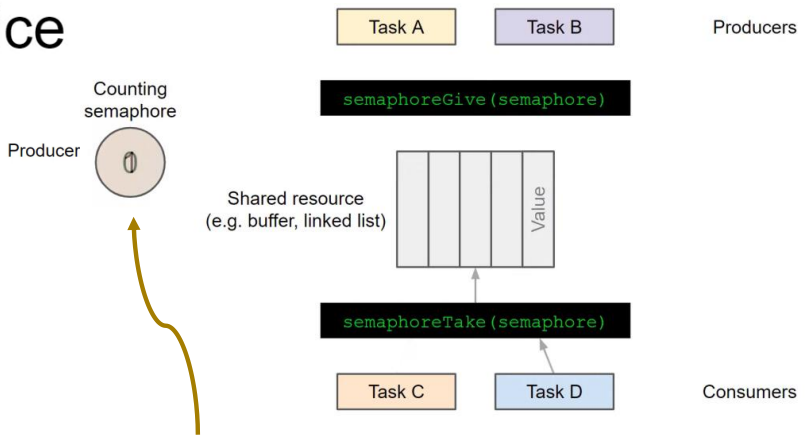
In practice and **the best way to implement it is the idea of using a producer and consumer design**, where there are tasks that produce shared resources and others that take and delete them. The shared resources will normally be enlarged (as buffer or additional lists).



Initially the traffic light count is 0 and reading is not allowed until a task enters a value in the shared resource.



Once the task accesses the shared variable, the count increases.



Once the value is positive, the consuming task takes the value and returns the count to 0.

7. Semaphore: Differences

Mutex

```
// Task 1
semaphoreTake(mutex)
// Use shared resource
semaphoreGive(mutex)


// Task 2
semaphoreTake(mutex)
// Use shared resource
semaphoreGive(mutex)
```

Semaphore

```
// Task 1 (producer)
// Add something to
// shared resource
semaphoreGive(semaphore)


// Task 2 (consumer)
semaphoreTake(mutex)
// Remove something from
// shared resource
```

In the MUTEX it implies ownership of the task because while **working with a shared resource it takes the value of the MUTEX and does not return it until it is done with it.**

Semaphores do not imply ownership as they only modify **the count values by altering a shared variable.**

7. Semaphore: Example 1

We define a traffic light count of at most 5 tasks accessing a shared variable.

In addition, a message structure and a traffic light handler that will store the traffic light count.

```
static const int num_tasks = 5; // Definimos el número de tareas que se crearán

// Example struct for passing a string as a parameter
// Se define una estructura de mensaje que pasaremos a las tareas como parámetro
typedef struct Message {
  char body[20];
  uint8_t len;
} Message;

// Globals
static SemaphoreHandle_t sem_params; // Conteo que dsiminuirá cuando las tareas accedan al recurso compartido
```

```
void myTask(void *parameters) {

  // Copiamos el mensaje creado que es un puntero como parámetro en la variable local
  // para la tarea
  Message msg = *(Message *)parameters;

  // Incrementamos el valor para indicar que se ha leído el mensaje
  xSemaphoreGive(sem_params);

  // Imprimimos el mensaje leído que ha incrementado el conteo del semáforo
  Serial.print("Recibimos: ");
  Serial.print(msg.body);
  Serial.print(" | Longitud: ");
  Serial.println(msg.len);

  // Esperamos y eliminamos la tarea librando la memoria de la tarea
  vTaskDelay(1000 / portTICK_PERIOD_MS);
  vTaskDelete(NULL);
}
```

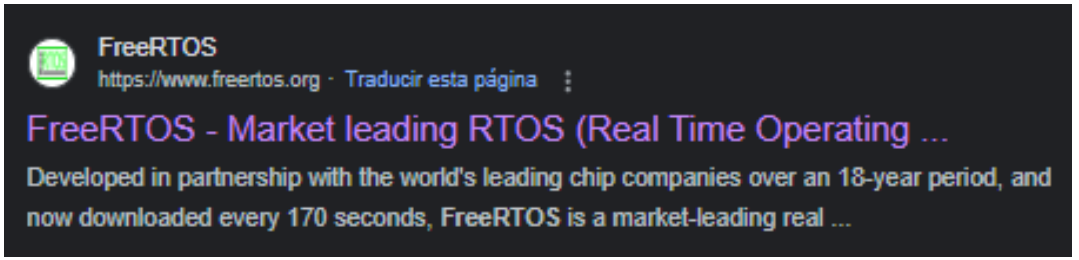
```
// Bucle que incrementa el número de tareas
for (int i = 0; i < num_tasks; i++) {

  // Generamos un número distinto por tarea que accederá al
  // recurso compartido común de mensaje
  sprintf(task_name, "Task %i", i);

  // Empieza la tarea y le pasamos el mensaje común a todas las tareas
  xTaskCreatePinnedToCore(myTask,
                          task_name,
                          1024,
                          (void *)&msg,
                          1,
                          NULL,
                          app_cpu);
}
```

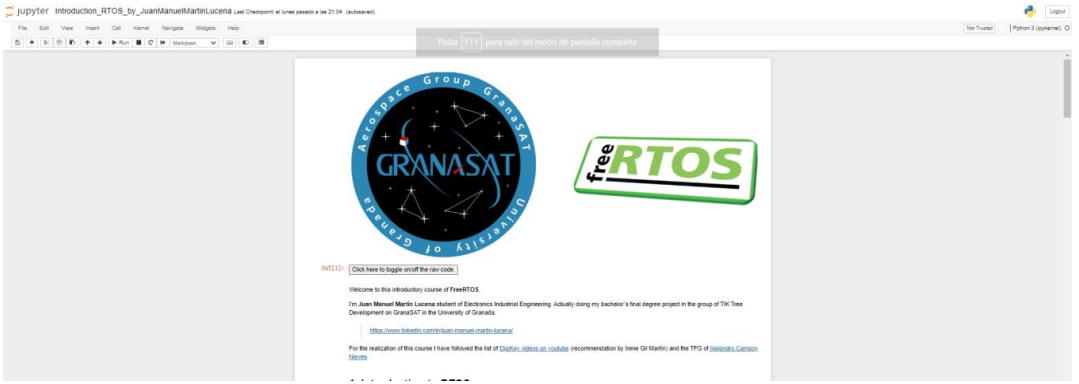
```
---FreeRTOS Counting Semaphore Demo---
Recibimos: Texto leido | Longitud: 11
Recibimos: Texto leido | Longitud: 11
Recibimos: Texto leido | Longitud: 11
Recibimos: Texto leido | Longitud: Recibimos: Texto leido | Longitud: 11
11
Todas las tareas han accedido al mensaje
```

The 5 tasks do the same thing, store a message and print it until the count reaches the indicated maximum of 5 and therefore, no more tasks can access the message to take it which is our shared resource.

Pending contents

8. Software timer
9. Hardware interrupts
10. Deadlocks and Starvation
11. Priority inversión
12. Multicore Systems





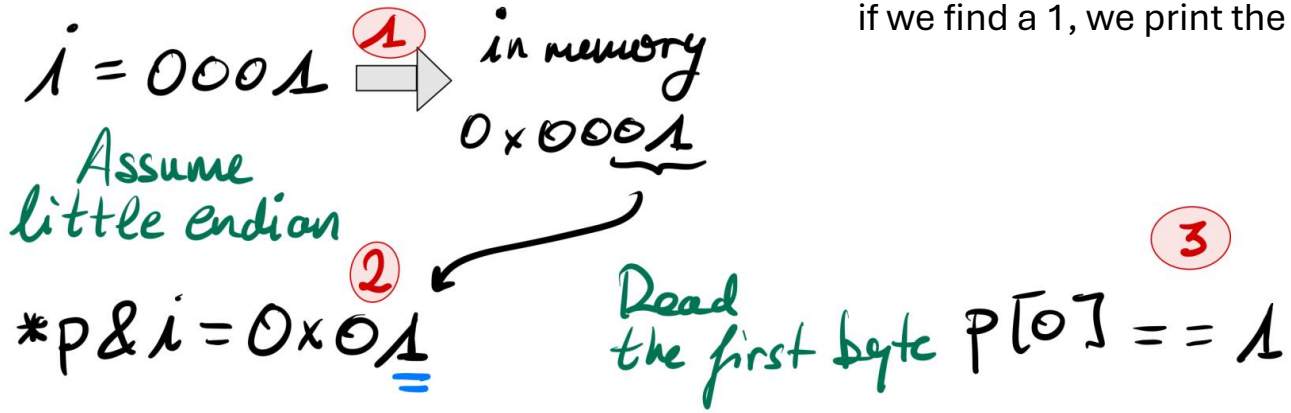
Appendix A: Big endian and Little endian

```c
#include <stdio.h>
#include <stdint.h>

int main(void)
{
   int16_t i = 1;
   int8_t *p = (int8_t *) &i;

   if (p[0] == 1) printf("Little endian\n");
   else printf("Big endian\n");
}
```

Motorola, for example, has followed the big endian standard, while Intel follows the little endian standard, among others.

In a few lines of code in C++, if we do not know what type of machine it is, we can find out in this way:

1 - We declare a 16-bit variable with a capacity of 4 integers.
2 - We create an 8-bit pointer and therefore 2 integers to which the memory address of i is assigned.
3 – We compare its initial position and read from memory, if we find a 1, we print the storage criterion in question.



Bibliografía

[1] E. Ministerio de Agricultura, Pesca y Alimentación, "Anuario de estadística del ministerio de agricultura, pesca y alimentación," 2021. https://www.mapa.gob.es/es/estadistica/temas/publicaciones/anuario-de-estadistica/default.aspx.

[2] SpringerOpen, "Un método de eco de impacto para detectar cavidades entre losas de vías ferroviarias y cimientos de suelo," 2021. https://jeas.springeropen.com/articles/10.1186/s44147-021-00008-w.

[3] ResearchGate, "Analysis of acoustic modes in optical waveguides," 2021. https://www.researchgate.net/figure/Longitudinal-wave-and-Shear-wave_fig3_266373533.

[4] Fakopp, "Fakopp microsecond timer," 2024. https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.dagasl.es%2Fpdf%2Fmicrosecondtimer.pdf&psig=AOvVaw1dSzAFu5hdaSLeTkpB8iNd&ust=1725559768458000&source=images&cd=vfe&opi=89978449&ved=0CBcQjhxqFwoTCKjU1pnxqYgDFQAAAAAdAAAAABAE.

[5] F. Bt., "Ms timer," 2023. https://fakopp.com/es/product/mstimer/.

[6] J. del Pino Mena, "Versión 0.6," 2022. Versión previa en la biblioteca UGR: https://digibug.ugr.es/handle/10481/77129?show=full.

[7] D. Incorporated, "Ap3429 datasheet," 2020. https://pdf1.alldatasheet.com/datasheet-pdf/view/1110453/DIODES/AP3429.html.

[8] A. Devices, "Introduction to the spi interface," 2018. https://www.analog.com/media/en/analog-dialogue/volume-52/number-3/introduction-to-spi-interface.pdf.

[9] S. Electronics, "Usb type-c datasheet," n.d. https://cdn.sparkfun.com/assets/e/b/4/f/7/USB-C_Datasheet.pdf.

[10] T. Instruments, "End equipment solutions guide," 2022. https://www.ti.com/lit/sg/sszb130d/sszb130d.pdf?ts=1724159049956&ref_url=https%253A%252F%252Fwww.youtube.com%252F.

[11] KNSCHA, "Rvt1000uf10v34rv0081 aluminium electrolytic capacitor," 2024. https://www.lcsc.com/product-detail/Aluminum-Electrolytic-Capacitors-SMD_KNSCHA-RVT1000UF10V34RV0081_C2904875.html.

[12] T. Instruments, "Ina219 datasheet," 2019. https://www.ti.com/lit/ds/symlink/ina219.pdf.

[13] Koney, "Informe mercado energético enero 2024," 2024. https://konery.com/informe-mercado-energetico-enero-2024/.

[14] N. Audio, "Transformada de fourier en señales de audio," 2020. https://www.nti-audio.com/es/servicio/conocimientos/transformacion-rapida-de-fourier-fft.

[15] R. M. J, "Ruiz m j 2017 software: Fourier synthetizer," 2017. https://mjtruiz.com/ped/fourier/.

[16] K. technologies, "Keysight n6705dc power analyzer," 2023. https://www.mouser.com/pdfdocs/2ugN6705-90001.pdf.

[17] S. technologies, "Sdm3065x series digital multimeter," 2021. https://int.siglent.com/u_file/download/22_03_29/SDM3065X_Usermanual_E02B.pdf.

[18] P. Manoonpong, "Lithium polymer information : Lithium polymer batteries,s," 2018. https://www.manoonpong.com/Other/main_page=page_2.pdf.

[19] LWFF, "Life wood for future web.." Accedido: 1 de junio, 2024.

[20] L. P. Database., "Life wood for future: Recovery of granada-vega poplar groves to boost biodiversity and long-term carbon capture through structural bioproducts.." Accedido: 1 de junio, 2024.

[21] E. comission, "Recovery of granada-vega poplar groves to boost biodiversity and long-term carbon capture through structural bioproducts," 2021. https://webgate.ec.europa.eu/life/publicWebsite/project/LIFE20-CCM-ES-001656/recovery-of-granada-vega-poplar-groves-to-boost-biodiversity-and-long-term-carbon-captur

[22] +IQAir, "Calidad del aire en granada," 2024. https://www.iqair.com/es/spain/andalucia/granada.

[23] C. oficial de aparejadores y arquitectos técnicos de Granada, "Contaminación y calidad del aire en el Área metropolitana de granada. parte 1," 2021. https://www.coaatgr.es/contaminacion-y-calidad-del-aire-en-el-area-metropolitana-de-granada-parte-1/.

[24] Propolus, "Beneficios ambientales de las plantaciones y bosques de chopos gestionados," 2021. https://propopulus.eu/es/beneficios-ambientales-de-las-plantaciones-y-bosques-de-chopos-gestionad#:~:text=Los%20chopos%20depuran%20el%20agua,el%20uso%20de%20la%20tierra.

[25] E. español, "La madera de chopo y el bosque urbano salvarán granada, la ciudad con la peor calidad del aire," 2022. https://www.elespanol.com/enclave-ods/historias/20220502/madera-bosque-urbano-salvaran-granada-ciudad-calidad/668433270_0.html#:~:text=La%20deforestaci%C3%B3n%20en%20la%20zona,sustituida%20por%20cultivos%20agr%C3%ADcolas%20intensivos.

[26] E.-L. Home, "Consolidated text: Directive 2010/31/eu of the european parliament and of the council of 19 may 2010 on the energy performance of buildings (recast)," 2010. https://eur-lex.europa.eu/eli/dir/2010/31/2021-01-01.

[27] E.-L. Home, "Consolidated text: Directive 2012/27/eu of the european parliament and of the council of 25 october 2012 on energy efficiency, amending directives 2009/125/ec and 2010/30/eu and repealing directives 2004/8/ec and 2006/32/ec (text with eea relevance)text with eea relevance," 2012. https://eur-lex.europa.eu/eli/dir/2012/27/2021-01-01.

[28] C. E. y Social Europeo, "Dictamen del comité económico y social europeo sobre la «construcción en madera para reducir el co2 en el sector de la construcción»," 2023. https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:52022AE6006.

[29] M. Adrados, "El chopo," 2021. https://www.maderasadrados.com/el-chopo.html.

[30] BOE, "Boe nº 179 de 27 de julio de 1992," 1992. https://www.boe.es/boe/dias/1992/07/27/pdfs/BOE-S-1992-179.pdf.

[31] BOE, "Boe nº 179 de 14 de marzo de 2003," 2003. https://www.boe.es/eli/es/res/2003/03/04/21/dof/spa/pdf.

[32] I. F. Europeo, "Construcción con madera: Sostenible y renovable," 2021. https://propopulus.eu/es/trazando-un-futuro-mas-verde-ipc27-y-el-papel-de-los-arboles-de-rapido-crecimiento/.

[33] O. de las Naciones Unidas para la Alimentación y la Agricultura (FAO), "Árboles de rápido crecimiento y su papel en la silvicultura y agroforestería," 2019. https://www.fao.org/forestry/ipc/.

[34] O. de las Naciones Unidas para la Alimentación y la Agricultura (FAO), "Comisión internacional del chopo (ipc)," 2020. https://www.fao.org/home/es.

[35] E. Ministerio de Ciencia e Innovación, "Materiales compuestos innovadores para la industria de la construcción: Proyecto compop," 2017. http://compop.ugr.es/.

[36] D. Incorporated, "Datasheet ap61100/ap61102," 2022. https://www.diodes.com/assets/Datasheets/AP61100-AP61102.pdf.

[37] "Mlp604060 1500mah 3.7v datasheet." https://download.mikroe.com/documents/datasheets/MLP604060%201500mAh%203.7V.pdf, 2021. Consultado el 15 de agosto de 2024.

[38] MikroElektronika, "Li-polymer battery 3.7v 1500mah." https://www.mikroe.com/li-polymer-battery-37v-1500mah?srsltid=AfmBOorNMPfer1lmULEeJQXBQXcnszU69NTPkoPl92J6x3ofyS0YMDA5.

[39] AliExpress, "803450 batería de litio de 1800mah," 2024. https://es.aliexpress.com/item/1005002621704252.html.

[40] E. Commission, "European union common charger directive," 2022. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32022L2380.

[41] S. Hymel, "Course introduction to rtos," 2024. https://www.youtube.com/watch?v=F321087yYy4&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz.

[42] E. confidencial, "¿cuántos zurdos hay en el planeta? este estudio desvela el porcentaje real," 2022. https://www.elconfidencial.com/tecnologia/ciencia/2020-04-06/cuantos-zurdos-planeta-mayor-estudio-porcentaje_2535896/.

[43] E. de Posgrado de la Universidad de Granada, "Fp053: Gestión de proyectos software," 2024. https://escuelaposgrado.ugr.es/static/EP_Management/*/showCard/23/FP/053.

[44] UGR, "Prototipado y test electrónicos," 2023. https://www.ugr.es/estudiantes/grados/grado-ingenieria-electronica-industrial/prototipado-test-electronicos.

[45] Espresif, "Esp32 series datasheet version 4.6," 2024. https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf.

[46] talent.com, "Salario medio para ingeniero electronico en españa, 2024," 2024. https://es.talent.com/salary?job=ingeniero+electronico#:~:text=El%20salario%20ingeniero%20electronico%20promedio,hasta%20%E2%82%AC%2038.000%20al%20a%C3%B1o.

[47] M. de Seguridad social, "Bases y tipo de cotización, 2024," 2024. https://es.talent.com/salary?job=ingeniero+electronico#:~:text=El%20salario%20ingeniero%20electronico%20promedio,hasta%20%E2%82%AC%2038.000%20al%20a%C3%B1o.

[48] glassdoor, "Sueldos para el puesto de ingeniero industrial junior en españa," 2024. https://www.glassdoor.es/Sueldos/ingeniero-industrial-junior-sueldo-SRCH_KO0,27.htm#:~:text=%C2%BFCu%C3%A1nto%20gana%20un%20Ingeniero%20industrial,junior%20es%20de%20%E2%82%AC25.269%20.

[49] R. Scheimbler, "Fft on esp32," 2024. https://github.com/fakufaku/esp32-fft.

[50] doxygen, “Doxygen official page,” 2019. https://www.doxygen.nl/manual/starting.html.

[51] PROMAX, “Fuente de alimentacion fac-363b,” 2024. https://int.siglent.com/u_file/download/22_03_29/SDM3065X_Usermanual_E02B.pdf.