UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

DISEÑO DE UN SISTEMA DE CONTROL DE ILUMINACIÓN MEDIANTE MICROCONTROLADOR DE BAJO COSTE Y BUS DALI

AUTOR: YELCO RODRÍGUEZ MÉNDEZ

TUTOR: MANUEL PINEDA SÁNCHEZ

COTUTOR: ÁNGEL SAPENA BAÑÓ

Curso Académico: 2019-20

AGRADECIMIENTOS

A mi familia por animarme día a día a mejorar como persona.

A mis amigos por porcionar esa vía de escape en los momentos difíciles.

A mis tutores por estar disponibles siempre que su ayuda era necesaria.

A la Fundación Elecnor por darme la oportunidad de realizar este proyecto.

Muchas gracias a todos.

RESUMEN

En este trabajo se realiza el diseño e implementación de un sistema de control de iluminación, aplicado a un banco de ensayos con cuatro tecnlogías diferentes de iluminación: fluorescente con balasto ferromagnético y electrónico, tubo led y luminaria DALI. Las funcionalidades básicas del diseño son el control del encendido y apagado de cada luminaria, regulación del flujo luminoso a través del protocolo analógico 0-1/10 V y del bus de comunicación DALI, y la toma de medidas eléctricas suficientes para caracterizar el funcionamiento de cada tecnología presente en el banco. El sistema de control está integrado por un microcontrolador, que ejecutará las operaciones directas sobre el conjunto de luminarias, y una aplicación para dispositivos Android que actúa como interfaz de usuario, recibiendo las órdenes solicitadas y mostrando los resultados de las medidas capturadas. El intercambio de información entre estos equipos se realiza de forma inalámbrica mediante comunicación bluetooth.

Palabras Clave: iluminación, fluorescente, led, DALI, ESP32, banco de ensayos, bluetooth, IoT, regulación, control, flujo luminoso, Android Studio, Arduino

ABSTRACT

The design and implementation of a lighting control system is made in this project, applied to a test bench with four different lighting technologies: fluorescent with ferromagnetic and electronic ballast, LED tube and DALI luminaire. The basic functionalities of the design are: on/off control of each luminaire, luminous flux regulation through the analog protocol 0/10 V and the DALI communication bus, and data collection in order to analyze the operation of every technology present in the bench. The control system consists of a microcontroller, which will conduct direct operations on the set of luminaires, and an application for Android devices that acts as a user interface, receiving the requested orders and showing the measurements taken. Both devices communicate via Bluetooth link.

Palabras Clave: lighting, fluorescent, LED, DALI, ESP32, test bench, bluetooth, IoT, regulation, control, luminous flux, Android Studio, Arduino

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Esta memoria expone el conjunto de actividades realizadas y resultados obtenidos durante la realización del Trabajo Final de Máster (TFM) del alumno Yelco Rodríguez Méndez, conducente a la obtención del título universitario oficial de Ingeniero Industrial.

El presente trabajo tiene como objetivo general la utilización y desarrollo de conocimientos adquiridos durante los estudios de grado y máster, de cara a alcanzar los objetivos específicos del encargo requerido. De forma transversal, se ponen en práctica competencias necesarias para la vida profesional, como son la capacidad de análisis, resolución de problemas y síntesis de resultados.

En este trabajo se profundiza en conocimientos relacionados con las medidas e instrumentación eléctricas, la electrónica de señal y la programación de microcontroladores y dispositivos móviles. Este hecho supone un gran reto pues, si bien antes de la realización del proyecto ya se disponen de conocimientos de base en estos campos, nunca se había llegado a completar un diseño pormenorizado que permita satisfacer los requerimientos de un encargo real.

1.1. Objeto del trabajo

El resultado esperado de este trabajo es el diseño e implementación de un sistema de control de iluminación que represente una alternativa de bajo coste en comparación con soluciones comerciales. El campo de aplicación de este sistema, que incluye diferentes tecnologías de iluminación, no está orientado tanto a su desarrollo comercial como tal sino a su utilización como puesto de prácticas que permita experimentar con esas diferentes tecnologías de iluminación a los estudiantes de Grados de Ingeniería de la rama industrial.

El proyecto requiere el desarrollo de un equipo de control remoto, basado en el microcontrolador ESP32, que genere las señales necesarias para el encendido y apagado de los sistemas gestionados, así como la regulación de flujo luminoso a través de protocolos analógicos, como el 1-10 V, y la comunicación con dispositivos DALI.

El sistema de control debe integrar una etapa de instrumentación que adquiera las medidas necesarias para caracterizar el funcionamiento de luminarias industriales y reportar los datos capturados a un dispositivo móvil basado en el sistema operativo Android mediante comunicación bluetooth.

El usuario controlará el conjunto de luminarias a partir de una aplicación para Android, que recibirá los comandos de operación enviando esta información al microcontrolador y servirá de interfaz para la visualización de las medidas registradas.

Como se ha indicado al principio, este sistema debe estar aplicado a un banco de ensayos para su utilización en prácticas docentes de asignaturas de la rama de ingeniería eléctrica, en las que se presentan diferentes tecnologías de iluminación y se estudian sus variables eléctricas básicas. En concreto, deberá contar con dos tubos fluorescentes (uno con balasto convencional y el otro con balasto electrónico), un tubo led con capacidad de regulación de flujo luminoso (por simplicidad en lo que sigue se utilizará simplemente la palabra flujo, ya que

no da lugar a confusión en este contexto) y un bus de comunicación DALI para el control de luminarias que acepten este protocolo.

La comparación de la eficiencia energética de los equipos presentes en el banco de ensayos queda excluida de las funcionalidades que debe permitir el montaje final. Este ámbito de estudio se materializa en las prácticas docentes a partir de otras experiencias no relacionadas directamente con el presente encargo.

Es importante destacar que el banco de ensayos, que debe estar gestionado por el sistema a diseñar, no se construye totalmente desde cero, sino que se parte de un montaje previo sobre el que se añaden los equipos no presentes inicialmente.

1.2. Objetivos

El objetivo último de este trabajo es la realización del encargo solicitado, tal como se ha descrito en el punto 1.1.

En cuanto a los objetivos parciales, estos pueden ser expresados en función de las diferentes tareas a realizar durante el desarrollo del trabajo:

- Diseño e implementación de los circuitos de control de operación y regulación de flujo luminoso para las diferentes tecnologías del banco de ensayos.
- Selección de los sensores a integrar, atendiendo fundamentalmente a sus características de consumo de energía, precisión, respuesta en frecuencia y simplicidad.
- Diseño e implementación de las interfaces A/D que adapten las medidas de corriente y de tensión a las entradas del ESP32.
- Diseño e implementación de dos aplicaciones (servidor-ESP32/cliente-app Android) con las siguientes funcionalidades básicas:
    - ESP32
        - Recepción inalámbrica y ejecución de comandos de operación.
        - Intérprete de comandos de captura (duración, señales, tasa de muestreo).
        - Captura, almacenamiento local de datos de operación y envío inalámbrico a la app Android.
    - Aplicación Android
        - Recepción de información procedente del ESP32 y procesado de datos.
        - Reporte gráfico y numérico de los datos recibidos por bluetooth.
        - Recepción de comandos de operación desde la interfaz de usuario y envío inalámbrico hacia el ESP32.

Se podrían listar una serie de objetivos implícitos que influyen en la concepción y diseño del sistema a implementar, tales como la fiabilidad del montaje, la seguridad en su operación tanto para las personas como para los propios equipos utilizados, la competitividad y menor coste en comparación con soluciones comerciales, la facilidad de uso, etc.

1.3. Estructura de la memoria

El primer punto desarrollado en este documento consiste en el diseño del hardware (capítulo 2). Se analizan los antecedentes y el equipo disponible para la construcción del banco de

ensayos. Dentro de cada tecnología de iluminación se realiza una breve descripción de sus principios de funcionamiento. A continuación, se procede al diseño de los circuitos necesarios para el control de encendido y apagado, regulación del flujo luminoso y captura de medidas de tensión y corriente. Por último, se integran los circuitos parciales y se presenta el montaje final (Figura 38. Esquema de integración de circuitos).

El siguiente capítulo (3) supone el estudio del bus de comunicación DALI. Al inicio, se enuncian las posibilidades de control de esta tecnología y se describen las bases del protocolo de comunicación. Seguidamente, se diseña el bus de comunicación y la interfaz de envío y recepción de tramas. Por último, se presentan los resultados de varios ensayos de comunicación entre el microcontrolador (controlado por puerto USB) y la luminaria DALI.

El tercer punto principal de esta memoria se centra en la construcción del software de control del banco de ensayos (capítulo 4). En primer lugar, se describe el código de programación del microcontrolador. Este dispositivo recibe las consignas de operación de forma remota y ejecuta las acciones de control sobre el banco de ensayos. Por otra parte, debe realizar la captura periódica de medidas y envío de datos a través de comunicación bluetooth. El segundo bloque de este capítulo se centra en el desarrollo de la aplicación Android, que sirve de interfaz de usuario para la recepción de consignas de operación y visualización de las medidas realizadas.

El último capítulo principal de este documento (5) se corresponde con el análisis de los resultados obtenidos en este trabajo y la presentación de conclusiones y futuras líneas de mejora.

Adicionalmente al cuerpo principal de esta memoria se incluyen el presupuesto de este proyecto y los anexos con el código de programación.

2. DISEÑO DEL HARDWARE

En este capítulo se presenta el diseño del hardware para la implementación del sistema de control de iluminación propuesto.

2.1. Antecedentes

En este apartado se describe el estado del banco de ensayos existente al inicio de la realización de este trabajo.

El montaje del que parte este trabajo está conformado por dos tubos fluorescentes de 36 W, uno de ellos accionado por un balasto convencional y el otro por un balasto electrónico, que permite la regulación del flujo luminoso.

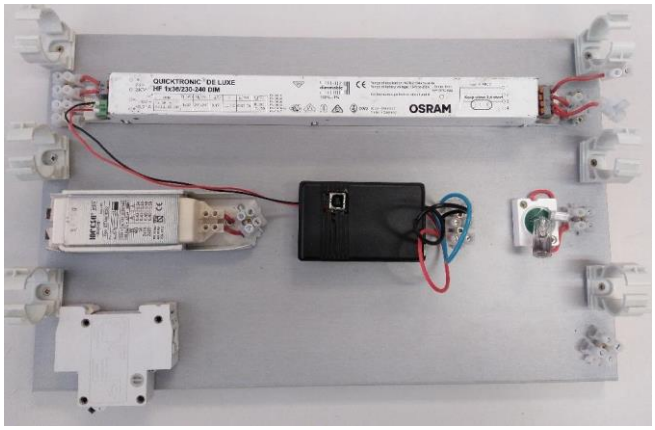


Figura 1. Banco de ensayos previo a este trabajo

Este banco de ensayos es utilizado en prácticas de asignaturas del Departamento de Ingeniería Eléctrica para caracterizar el funcionamiento de la tecnología de iluminación basada en tubos fluorescentes. Se estudia el comportamiento durante los transitorios de encendido y apagado, así como en régimen permanente, de este tipo de luminarias, obteniendo valores de tensión, corriente, potencia y factor de potencia, a fin de conocer un cierto orden de magnitud de las características eléctricas de estos equipos.

Por otra parte, se dispone de una placa para ensayos con tecnología led y de bus DALI recientemente adquirida y que aún no se ha utilizado en actividades docentes.

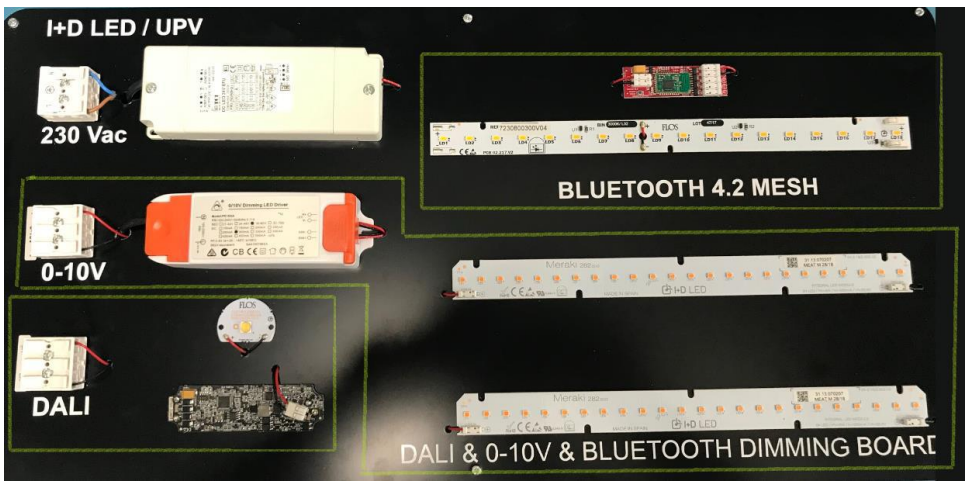


Figura 2. Dimming Board I+D LED

2.2. Equipo disponible

En este subapartado se detalla con más detalle el material disponible para la realización del presente trabajo.

2.2.1. Tubo fluorescente con balasto convencional

Los tubos fluorescentes son lámparas de descarga de vapor de mercurio a baja presión. La iluminación se produce por la circulación de corriente que provoca la ionización del gas en el interior del tubo. El movimiento de electrones resultante supone una emisión de radiación ultravioleta, que es convertida en radiación visible por el polvo fluorescente que recubre el tubo.

Esta tecnología requiere de tres elementos auxiliares para su funcionamiento:

- **Cebador**: Se trata de una lámina bimetálica en forma de U dentro de una ampolla con gases a baja presión. Provocan un breve pico de tensión entre los electrodos del tubo para iniciar la descarga y producir la ionización del gas de su interior.
- **Condensador**: Se coloca en paralelo al cebador. Su utilización permite atenuar el pico de tensión y extenderlo en el tiempo, incrementando la vida útil del conjunto, y absorber el ruido eléctrico producido por las descargas de los electrodos.
- **Balasto ferromagnético**: Está constituido por una reactancia inductiva que limita la corriente que circula por la lámpara. El tubo fluorescente se comporta como una impedancia de pendiente negativa, es decir, cuanto mayor sea la corriente que lo atraviesa menor es la resistencia. Si no se dispone de balasto, la corriente por el tubo se incrementaría hasta destruir el mismo.

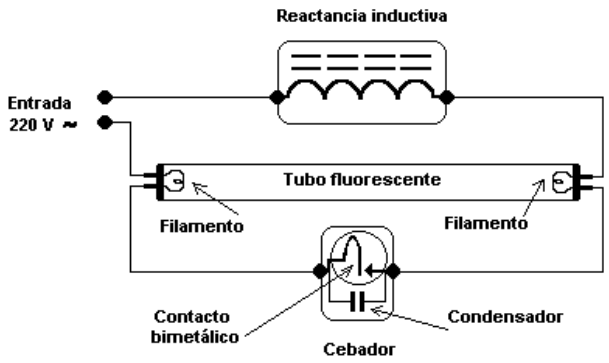


Figura 3. Esquema eléctrico del tubo fluorescente con balasto convencional. Fuente (Blanco Espinosa, 2016)

En el laboratorio se dispone de un tubo de 36 W “Philips TLD 36W/840”. El balasto utilizado es el “Inecsa mini ECNM 4022”, que se puede conectar a un tubo de 36 W con circulación de corriente de 0.43 A. El cebador se encuentra sin el capuchón, lo que permite la visualización de su funcionamiento.

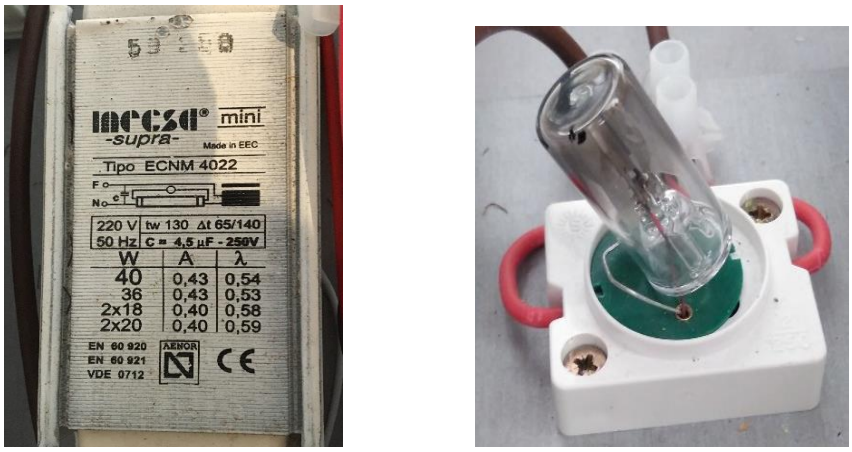


Figura 4. Balasto ferromagnético y cebador

2.2.2. Tubo fluorescente con balasto electrónico

El segundo tubo fluorescente está conectado a un balasto electrónico, que reemplaza el conjunto balasto convencional, cebador y condensador. En comparación con el balasto inductivo, proporciona un encendido más rápido, reduce el efecto flicker y mejora el rendimiento y el factor de potencia del conjunto. Como característica adicional, permite la regulación del flujo luminoso a través de los terminales 1-10 V.

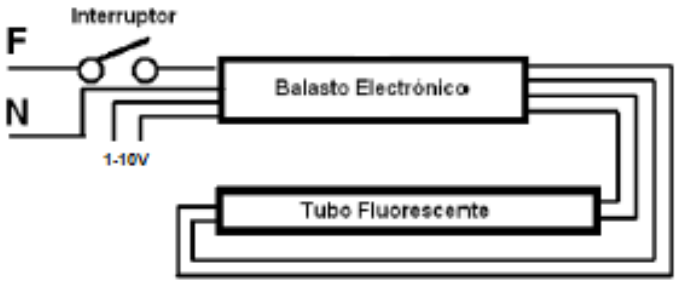


Figura 5. Esquema de conexión fluorescente con balasto electrónico. Fuente: (Práctica 6. Alumbrado. Medidas de iluminación con balasto electrónico)

En concreto, los equipos disponibles son un tubo fluorescente de 36 W “OSRAM L36W/21-840” y un balasto electrónico “Quicktronic Deluxe HF 1x36/230-240 DIM”, adecuado para su utilización con tubos de 36 W.

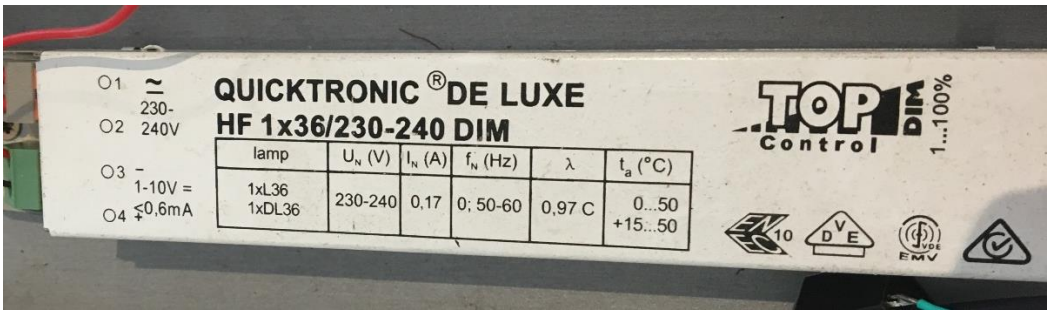


Figura 6. Balasto electrónico

2.2.3. Tubo Led y dimmer regulable

La tecnología de iluminación led se basa en la utilización de módulos constituidos por matrices de leds individuales. Un diodo led constituye una unión semiconductora p-n que libera energía en forma de fotones cuando se polariza directamente. Para su conexión a la red eléctrica se incorpora un controlador (driver), que constituye una fuente de alimentación de intensidad constante. Algunos drivers (dimmers) permiten la regulación del flujo luminoso, variando la tensión aplicada.

Para el caso concreto de este trabajo se dispone de un tubo led de 18 W “Prilux EC Tube T8”. El dimmer utilizado es el “0/10V Dimming Led Driver PE18AA”, procedente de la DIMMING BOARD. Este controlador proporciona un rango de tensión de salida de 30-60 V con una corriente de 300 mA. En el diseño de la DIMMING BOARD, este dimmer alimentaba un conjunto de tiras led de 12 W, pero en este trabajo se utilizará para la alimentación del tubo led de 18 W/90V. Para ello, se realiza una reconexión de los leds en la tira del tubo, reduciendo la longitud efectiva del mismo en 1/3 lo que permite la regulación de flujo completa con un driver de tensión máxima de 60V como el disponible. Con esto se consigue disponer de un tubo led de tensión nominal de 50V, que ya puede ser conectado al dimmer disponible.

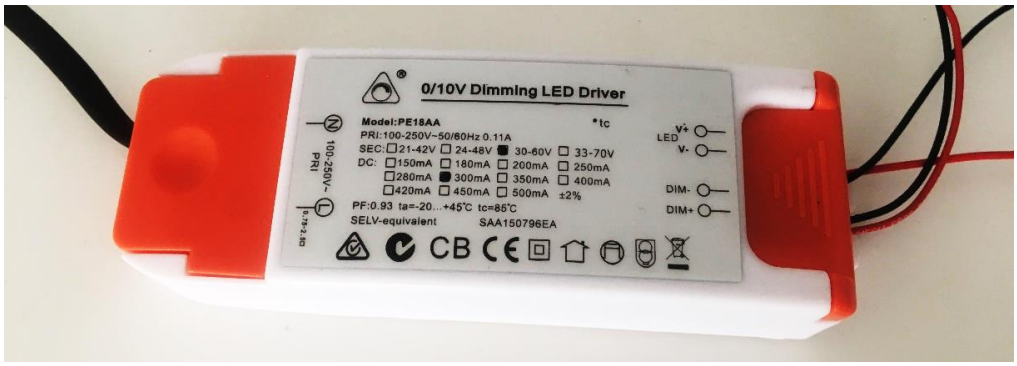


Figura 7. Dimmer led

2.2.4. Esclavo DALI

Para la realización del trabajo relacionada con la implementación de la tecnología DALI en el banco de ensayos se utiliza el esclavo de la placa DIMMING BOARD de I+D Led conectado a una carga de un led. Sus especificaciones se comentarán en detalle en el apartado correspondiente al diseño del bus DALI (capítulo 3). La alimentación del esclavo DALI se realiza a partir de una fuente de 24 V, ya incluida en la DIMMING BOARD.

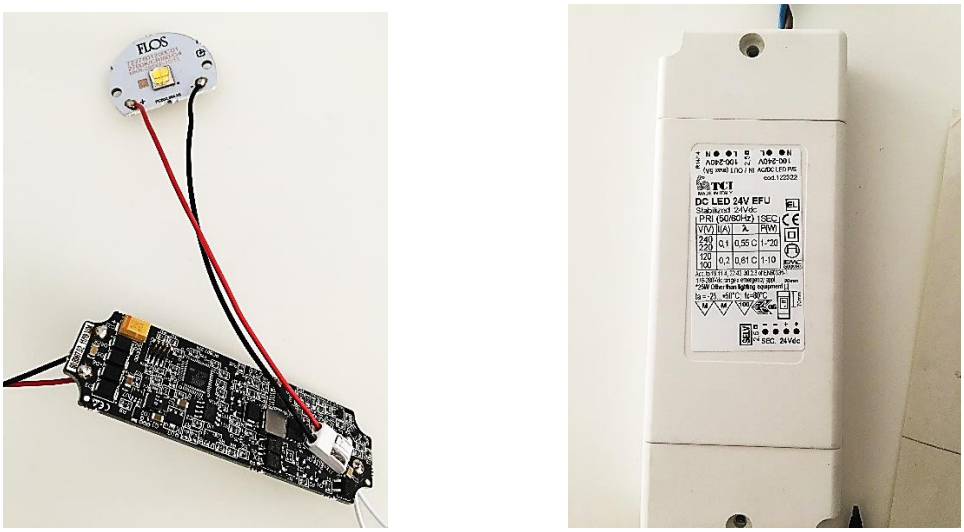


Figura 8. Esclavo DALI con led de prueba y fuente de alimentación de 24 V DC

2.2.5. Tablet Android

En el laboratorio se dispone de diversas tablets con sistema operativo Android y conexión Wifi y bluetooth. Estos dispositivos serán los encargados de recibir los datos procedentes del banco de ensayos y enviar las órdenes de encendido/apagado, regulación del flujo luminoso y captura de señales.

Durante los ensayos de este trabajo se ha utilizado el modelo “Asus Memo Pad 8” de 8 pulgadas.

Figura 9. Tablet Android utilizada en este trabajo

2.2.6. Equipos de apoyo

Durante la fase de diseño, así como en la implementación de la versión final, se han utilizado diversas herramientas que permiten disponer de la información necesaria para validar los circuitos planteados y servir de apoyo en la conexión de los circuitos.

-**Multímetro digital "ISOTECH IDM 62T"**: Se ha utilizado para la toma de medidas de tensión y corriente tanto en continua como en alterna, la comprobación de los valores de resistencias utilizadas en el circuito y la verificación de la capacidad de los condensadores utilizados.

-**Osciloscopio de conexión a PC "PicoScope2203":** Este osciloscopio compacto permite la visualización de dos señales simultáneas en un ordenador. Se ha utilizado para el estudio del funcionamiento del bus DALI durante la fase de diseño y control a través del puerto serie del PC.

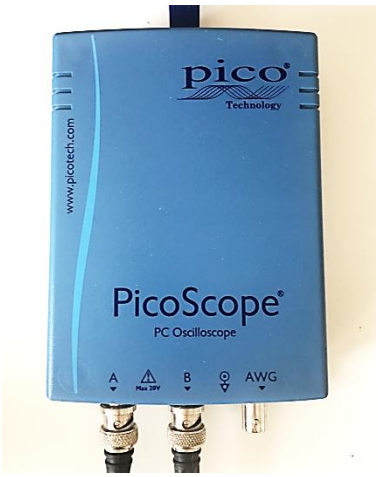


Figura 10, Osciloscopio PicoScope2203

-**Soldador de estaño "JBC 14ST" de 11 W:** Este equipo se ha utilizado para la soldadura de cables a los terminales de los relés y para realizar una terminación en los conductores que permita su incorporación a las placas de prototipos utilizadas.

-Placas de prototipos y cables Dupont: Para la realización de los montajes provisionales se utilizan placas de prototipos (protoboards) y cables Dupont macho a hembra que permiten la conexión entre el microcontrolador y el circuito. Debido a la situación de la pandemia de COVID-19, el circuito final se ha montado íntegramente en placas de prototipos.

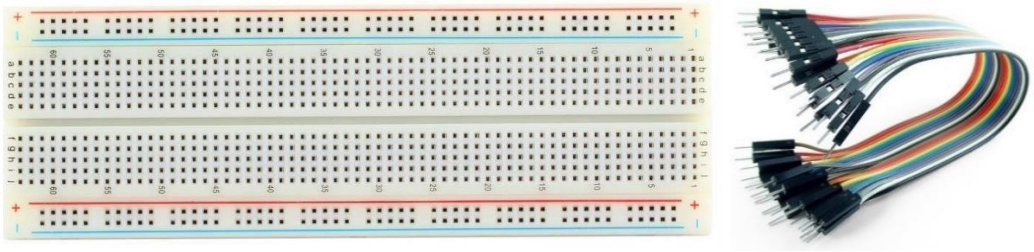


Figura 11. Protoboard y cables Dupont

-**Resto de elementos:** Cables de conexión, destornilladores, alicates pelacables, tijeras de electricista, etc.

2.3. Elección del microcontrolador

Para la implementación del sistema de control y toma de medidas a diseñar en este trabajo es necesario determinar en primer lugar el microcontrolador a utilizar. El microcontrolador escogido condicionará varios aspectos de la fase de diseño, como la elección de la fuente de

alimentación, la interfaz de control, regulación y toma de medidas del banco de ensayos, y la tipología de comunicación entre usuario y equipos utilizados.

Antes de presentar diferentes alternativas, es interesante distinguir entre lo que se conoce como **microcontrolador (µC)** y **microprocesador (µP).** En este trabajo, ya desde un principio se ha hablado de utilizar un microcontrolador, hecho que tiene su justificación.

De forma muy general, la diferencia entre µP y µC reside en las unidades presentes en cada uno y su rendimiento. Una analogía muy simple es que un µP se puede relacionar con un ordenador, mientras que un µC se asemeja a un autómata programable.

Un **microprocesador** es un circuito programable que se encarga del control y procesado de datos de un sistema digital (Tocci & Widmer, 2003). Consta de una unidad aritmeticológica, una unidad de control (CPU) y un conjunto de registros para el almacenamiento temporal de información. A través de una serie de buses, el microprocesador se comunica con las unidades de memoria y de entradas/salidas.

Los µP poseen CPUs más potentes, por lo que son utilizados para el procesamiento de gran cantidad de datos. Para su uso en aplicaciones de IoT ("Internet of Things") es necesario incorporar unidades auxiliares, como memoria e interfaces de entradas/salidas, incrementando su precio en comparación con los microcontroladores. Un ejemplo de la integración del microprocesador con el resto de unidades es la Raspberry Pi, cuyo precio medio es de 40 € frente al coste de un ESP32, en torno a 5 €.

Figura 12. Raspberry Pi 3 Modelo B+. Fuente: RS Components Spain

En contraposición, un **microcontrolador** es un circuito integrado que contiene una CPU, memoria y periféricos de entrada/salida. Los µC ejecutan el programa almacenado en su memoria, mientras que en un µP el sistema operativo es el que gestiona el uso del hardware disponible, actuando de intermediario con la aplicación ejecutada.

Los µC son ideales para la ejecución de programas que no requieren una alta velocidad de procesamiento ni una elevada cantidad de memoria y que interactúen directamente con el entorno, como son el control de equipos industriales y toma de medidas de forma remota.

Por el hecho de disponer en un solo circuito integrado de todas las unidades necesarias, así como por su menor precio, para este trabajo se plantea el uso de un microcontrolador para el control remoto del banco de ensayos.

En el mercado existe una amplia gama de microcontroladores diferentes. A continuación, se presentan y comparan diversas alternativas.

La marca Arduino es ampliamente conocida en el entorno de diseño de proyectos IoT. Proporciona placas de desarrollo fácilmente programables con el entorno de desarrollo integrado (IDE) de Arduino. Para su análisis se escoge el modelo **Arduino Uno R3**, ya que se trata de la placa de desarrollo más utilizada de la familia Arduino (ProyectoArduino, 2020).

Figura 13. Arduino Uno R3. Fuente: amazon.com

El Arduino Uno R3 está basado en el microcontrolador ATmega328P, con una frecuencia de reloj de 16 MHz. Dispone de conexión USB para su programación o alimentación, 14 pines digitales, de los cuales 6 permiten una modulación por ancho de pulso (PWM), 6 entradas con conversión analógico a digital (A/D) de 10 bits de resolución (1024) y posibilidad de alimentación externa en el rango de 6 a 20 V. A priori, tanto los pines digitales como los analógicos de esta placa serían apropiados para el control y toma de medidas del banco de ensayos.

El principal inconveniente de esta placa es la ausencia de comunicaciones inalámbricas por defecto. Para dotar de comunicación por Wifi o Bluetooth, es necesario adquirir módulos de expansión (shields), con precios medios entre 5 y 10 €. Esto sumado al precio de la placa (22 €), hace que el coste del conjunto placa+shield esté en torno a 30 €.

La segunda opción a analizar es el SoC (System on Chip) **ESP8266**, de la marca Espressif Systems. Este módulo consiste en un chip con capacidad de comunicación por Wifi con 16 pines de entradas/salidas de propósito general, con una entrada con conversión A/D de 10 bits (Martínez La Osa, Sapena Bañó, & Martínez Román, 2018).

Figura 14. Placa de desarrollo ESP8266 NodeMCU Iolin. Fuente: electronilab.co

La placa de desarrollo más utilizada es la **NodeMCU Iolin**, que incluye un chip ESP8266 modelo ESP-12, y que permite su programación a partir del IDE de Arduino y de la conexión USB. La

alimentación de la placa se puede realizar a través de la conexión USB o mediante una fuente externa de 3.3 V.

La principal ventaja de esta placa es su reducido precio, en torno a 5€. Esto provoca que haya sido utilizada ampliamente en los últimos años para proyectos de IoT.

En cuanto a los inconvenientes, el principal aspecto negativo de este módulo es que sólo dispone de una entrada con conversión A/D. En este trabajo es necesario medir dos variables eléctricas de forma simultánea, por lo que sería necesario adquirir un convertidor multicanal A/D externo, con el incremento de costes que esto supondría. Por otra parte, este módulo no dispone de comunicación bluetooth, reduciendo el abanico de posibilidades a la hora de plantear alternativas de comunicación de forma inalámbrica.

La última opción que se presenta es el SoC **ESP32**, de la marca Espressif Systems. Se trata de la evolución del ESP8266, y cuenta con características mejoradas.

Las principales diferencias del ESP32 con respecto a su predecesor son:

- Procesador más potente, de doble núcleo con mayor velocidad de reloj.
- Incorpora comunicación por bluetooth, además de Wifi (que también está disponible en el ESP8266).
- Dispone de mayor número de pines accesibles, así como 18 entradas con conversión A/D de 12 bits y 2 salidas con conversión digital a analógica (D/A).
- Dispone de mayor memoria, así como una serie de características adicionales tales como arranque seguro, encriptación por software, generador de números aleatorios y reloj de tiempo real.

Existen numerosos kits de desarrollo con ESP32. Se escoge el módulo **DevKit v1**, fabricado por Espressif Systems, y que destaca por su diseño compacto, similar al ESP8266 Node MCU. Sus principales características diferenciadoras son que incorpora el convertidor serie CP2102, que permite controlar el puerto USB con velocidades de hasta 3 Mbit/s; mantiene la mayoría de los pines del ESP32 accesibles para su uso externo; y dispone de un regulador de tensión que permite su alimentación en un rango amplio de tensiones (Martínez Pelayo & Mena Rodríguez, 2019).

El conjunto de ventajas presentadas del microcontrolador ESP32 en comparación con el ESP8266, junto a su similar precio de venta (en torno a 7€), suponen la elección de la placa de desarrollo ESP32 DevKit v1 para el control y captura de medidas requerido en este trabajo.

Figura 15. Placa de desarrollo ESP32 DevKit v1. Fuente: tiendatec.es

2.4. Fuente de alimentación

La elección de la fuente de alimentación es un apartado clave de este trabajo. Se necesita alimentar el microcontrolador escogido, los circuitos de encendido y apagado, y el bus DALI. A priori y salvo que no fuese posible, se busca utilizar una única fuente de alimentación para toda la parte de tensión continua del diseño a realizar.

El ESP32 dispone de un regulador lineal disipativo que proporciona una tensión de salida de 3.3 V, concretamente el NCP1117 (Espressif Systems, 2016). Acudiendo a la hoja de características de este regulador se puede comprobar que el rango de tensión de entrada para una salida de 3.3 V es de 4.75 a 15 V. En todo caso, se buscará una opción de alimentación que no esté en el límite del rango, para evitar calentamientos excesivos.

En cuanto al bus DALI, debe presentar un rango de tensión para el nivel alto de entre 11.5 y 20.5 V, y entre -4.5 y 4.5 V para el nivel bajo. Por tanto, se deberá escoger una fuente que proporcione una salida de tensión dentro del rango admisible para el nivel alto.

Por último, para el encendido y apagado de los circuitos de potencia de forma remota se plantea el uso de relés. En el laboratorio existe una cierta gama de relés con tensión de activación de 5 V. No obstante, la alimentación de este circuito podría realizarse con otro nivel mediante soluciones sencillas de implementar, como por ejemplo a través de un divisor resistivo. Por esta razón, la elección del relé no supone un factor crítico a la hora de determinar la fuente de alimentación.

Teniendo en cuenta estos aspectos comentados, se plantea el uso de una fuente de alimentación de 12 V y corriente máxima de 450 mA, con un precio estimado de 2.95 €.

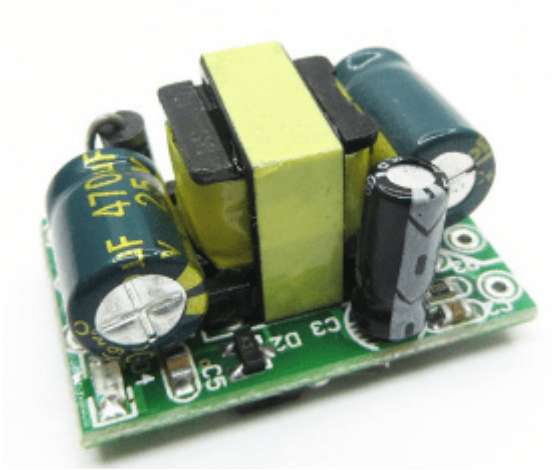


Figura 16. Fuente de alimentación de 12 V DC

La fuente escogida está dentro de los límites admisibles para la alimentación del ESP32. Sin embargo, se encuentra cerca del límite de funcionamiento del bus DALI. Por esta razón, será de especial importancia verificar la correcta comunicación entre el microcontrolador y el esclavo DALI para aceptar el uso de la fuente escogida como definitivo en el diseño del banco de ensayos. Esto se comprueba en el capítulo 3.

Una vez adquirida la fuente y estando ya en el laboratorio, se han soldado las conexiones necesarias para su uso y se ha encapsulado posteriormente para su utilización de forma segura. El resultado final se muestra en la siguiente figura:

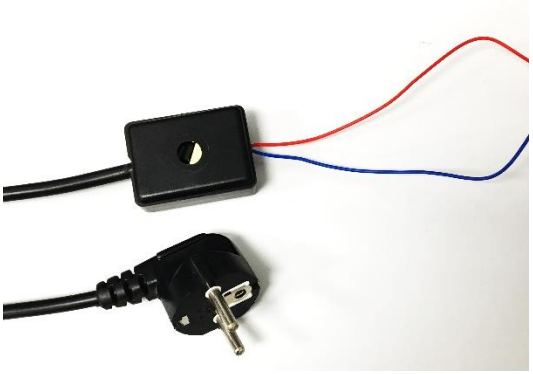


Figura 17. Fuente de alimentación encapsulada

2.5. Control de Encendido y Apagado

Una de las especificaciones que debe cumplir el diseño propuesto para este trabajo es el encendido y apagado de los equipos instalados en el banco de pruebas (tubos fluorescentes con balasto convencional y electrónico y tubo led).

Para la conmutación de los circuitos, la opción más simple que existe es la utilización de interruptores mecánicos de accionamiento manual. Esta posibilidad es directamente descartada, ya que lo que se busca es el control remoto del encendido y apagado mediante el microcontrolador.

A través de un pin de salida digital del microcontrolador se busca crear un circuito de mando que controle la apertura y cierre de los circuitos de potencia. Con un circuito en continua y de baja potencia se pretende gobernar los diferentes circuitos de potencia en corriente alterna del banco de ensayos. La solución más extendida que permite este control es la utilización de relés.

El modelo disponible en el laboratorio es el relé sin enclavamiento “KEMET EC2/EE2” de tensión nominal de accionamiento de la bobina de 5 V y resistencia de bobina de 178 Ω. Es capaz de maniobrar un circuito de alterna de hasta 250 Vrms con tiempos de conmutación entre 1 y 2 ms, por lo que a priori es válido para el diseño requerido.

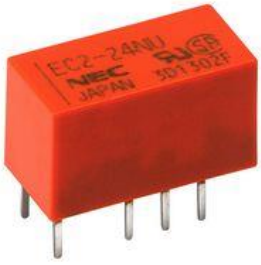


Figura 18. Relé KEMET EC2/EE2. Fuente: Hoja de características de la serie de relés EC2/EE2

Las salidas digitales del ESP32 presentan un nivel de tensión de 3.3 V, mientras que la tensión de funcionamiento del relé es de 5 V. Es necesario diseñar un circuito que permita adaptar estos niveles de tensión.

Se plantea un divisor de tensión a partir de la alimentación a 12 V y un control de conmutación del relé a través de un transistor con su base conectada a una salida digital del microcontrolador. El circuito resultante se muestra en la siguiente figura:

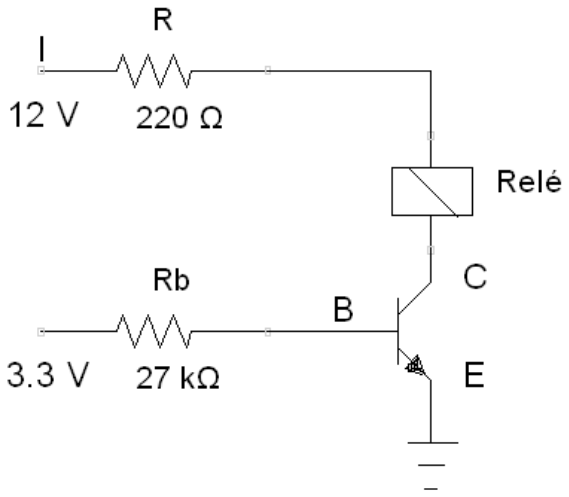


Figura 19. Circuito de control On/Off

Se utiliza el transistor NPN BC337-40, disponible en el laboratorio, que presenta las siguientes características:

Maximum ratings (T_A = 25°C) — **Grenzwerte (T_A = 25°C)**

			BC337	BC338
Collector-Emitter-volt. – Kollektor-Emitter-Spannung	E-B short	V_{CES}	50 V	30 V
Collector-Emitter-volt. – Kollektor-Emitter-Spannung	B open	V_{CEO}	45 V	25 V
Emitter-Base-voltage – Emitter-Basis-Spannung	C open	V_{EBO}	5 V	
Power dissipation – Verlustleistung		P_{tot}	625 mW [1])	
Collector current – Kollektorstrom (dc)		I_C	800 mA	
Peak Collector current – Kollektor-Spitzenstrom		I_{CM}	1 A	
Base current – Basisstrom		I_B	100 mA	
Junction temperature – Sperrschichttemperatur Storage temperature – Lagerungstemperatur		T_j T_S	-55...+150°C -55...+150°C	

Characteristics (T_j = 25°C) — **Kennwerte (T_j = 25°C)**

			Min.	Typ.	Max.
DC current gain – Kollektor-Basis-Stromverhältnis [2])					
V_{CE} = 1 V, I_C = 100 mA	Group -16	h_{FE}	100	160	250
	Group -25	h_{FE}	160	250	400
	Group -40	h_{FE}	250	400	630
V_{CE} = 1 V, I_C = 300 mA	Group -16	h_{FE}	60	130	–
	Group -25	h_{FE}	100	200	–
	Group -40	h_{FE}	170	320	–
Collector-Emitter saturation voltage – Kollektor-Emitter-Sättigungsspg. [2])					
I_C = 500 mA, I_B = 50 mA		V_{CEsat}	–	–	0.7 V
Base-Emitter-voltage – Basis-Emitter-Spannung [2])					
V_{CE} = 1 V, I_C = 300 mA,		V_{BE}	–	–	1.2 V

Tabla 1. Características del transistor NPN BC337-40. Fuente: hoja de características Diotec

Los datos de partida para el diseño son:

$$R_{relé} = 178\,\Omega;\ \beta = 400; V_{ce} = 0.7\,V; V_{be} = 1.2\,V$$

El primer paso consiste en obtener la caída de tensión entre la alimentación a 12 V y el colector del transistor.

$$V_{IC} = 12 - V_{ce} = 12 - 0.7 = 11.3\,V$$

A continuación, se calcula la resistencia del divisor resistivo para que la tensión en bornes del relé sea de 5 V.

$$V_{relé} = V_{IC} \cdot \frac{R_{relé}}{R + R_{relé}} = 11.3 \cdot \frac{178}{178 + R} = 5 \leftrightarrow R = 224.28\,\Omega \rightarrow 220\,\Omega$$

Se obtiene un valor de resistencia de 224.28 Ω, que se normaliza al valor más cercano disponible en el laboratorio, que es 220 Ω.

Por último, se obtiene la corriente que circula por la base al activar la salida digital del microcontrolador y la resistencia de base necesaria para lograr la saturación del transistor en la conmutación.

$$I_c = \frac{V_{IC}}{R + R_{relé}} = \frac{11.3}{220 + 178} = 28\,mA \rightarrow I_b = \frac{I_c}{\beta} = 70\,\mu A$$

$$R_b = \frac{3.3 - V_{be}}{I_b} = \frac{3.3 - 1.2}{70}\frac{V}{\mu A} = 30\,k\Omega \rightarrow 27\,k\Omega$$

La resistencia de base finalmente utilizada, una vez normalizada al valor más cercano disponible, es de 27 kΩ.

Una vez obtenido el circuito teórico se realizan los primeros ensayos para comprobar su correcta funcionalidad. Se comprueba que la conmutación se realiza de forma correcta. Sin embargo, tras varios ciclos de conmutación, el relé que controla el encendido y apagado del tubo fluorescente con balasto convencional deja de funcionar, quedando en la posición de circuito cerrado.

Este fallo se debe a los picos de corriente que se producen durante el encendido del tubo fluorescente, que pueden superar los 700 mA para un tubo de estas características (giangrandi, 2017).

Por esta razón, se decide utilizar otro modelo de relés que sea capaz de soportar los picos de corriente observados, con un precio ligeramente superior al anterior modelo[1]. Se escoge el relé "PCH-105L2M" de la marca "TE Connectivity", que es capaz de soportar cargas en alterna a 250 V y corrientes de hasta 5 A.

Figura 20. Relé PCH-105L2M. Fuente: Hoja de características de TE Connectivity

[1] El relé KEMET EC2 tiene un precio medio de 1.55€ y el relé PCH-105L2M 1.70€. Fuente: Farnell.com

Se realiza un redimensionado del circuito, teniendo en cuenta que la resistencia del bobinado del nuevo relé a utilizar es de 125 Ω.

$$V_{relé} = V_{IC} \cdot \frac{R_{relé}}{R + R_{relé}} = 11.3 \cdot \frac{125}{125 + R} = 5 \leftrightarrow R = 157.5\ \Omega \rightarrow 220\ \Omega$$

La resistencia del divisor de tensión no varía con respecto al diseño original, ya que la resistencia normalizada más cercana al valor teórico y disponible en el laboratorio es de 220 Ω.

$$I_c = \frac{V_{IC}}{R + R_{relé}} = \frac{11.3}{220 + 125} = 32.8\ mA \rightarrow I_b = \frac{I_c}{\beta} = 82\ \mu A$$

$$R_b = \frac{3.3 - V_{be}}{I_b} = \frac{3.3 - 1.2}{82}\frac{V}{\mu A} = 25.6\ k\Omega \rightarrow 27\ k\Omega$$

De manera análoga, la resistencia de base más cercana al valor calculado sigue siendo la del diseño original (27 kΩ), por lo que la sustitución de un relé por otro no varía el diseño inicialmente planteado.

Con el nuevo relé ("PCH-105L2M") se realizan ensayos de encendido y apagado de los tubos fluorescentes, de balasto convencional y electrónico, y del tubo led comprobándose el correcto funcionamiento del diseño planteado.

2.6. Regulación del flujo luminoso

En este apartado se aborda el diseño de la interfaz que permite regular el flujo luminoso del tubo fluorescente con balasto electrónico y del tubo led. Ambos tubos están controlados por dispositivos que admiten una regulación a través de un protocolo de control similar. En el balasto electrónico, la regulación es 1-10 V y en el driver led es 0-10 V. Por esta razón, se plantea de forma preliminar un circuito de regulación compartido entre ambas tecnologías que deberá se verificado mediante ensayos posteriores.

2.6.1. Control del balasto electrónico

El balasto electrónico es un dispositivo que consta de una serie de etapas de rectificación, filtrado y generación de señales de alta frecuencia (30 kHz) y permite el encendido del tubo fluorescente sin necesidad de disponer de cebador y reactancia inductiva (Blanco Espinosa, 2016).

Proporciona una serie de ventajas respecto al balasto convencional, que se resumen en el aumento del rendimiento y del factor de potencia, ausencia del efecto flicker y capacidad de regulación.

El balasto realiza la regulación del flujo variando la frecuencia y la amplitud de la tensión aplicada al tubo. El control externo del balasto se realiza a partir de una entrada 1-10 V.

El esquema de conexión se muestra en la siguiente figura:

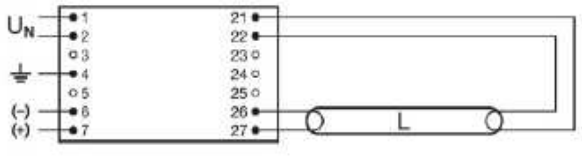


Figura 21. Esquema de conexión del balasto electrónico. Fuente: (OSRAM, 2014)

El sistema de control 1-10 V permite regular el flujo luminoso, a partir de dos hilos cuya polaridad debe respetarse, desde el 1 % para una tensión de 1 V y el 100 % para una tensión de 10 V (GEALed, 2016).

Internamente, el balasto contiene una fuente de tensión conectada a una resistencia limitadora, por lo que el sistema más sencillo para variar la tensión entre los dos terminales de control es la utilización de una resistencia variable de accionamiento manual. Esta solución se descarta, ya que se busca un control remoto sin intervención directa sobre el banco de ensayos.

Una alternativa es el uso de la regulación por PWM (Pulse Width Modulation), a través de un pin de salida digital del microcontrolador. La modulación por ancho de pulso permite variar el ciclo de trabajo de la señal de salida de un pin digital. El ESP32 dispone de 16 canales independientes en los que se puede configurar la frecuencia, resolución y ciclo de trabajo de la señal de salida. El inconveniente de esta opción es que el nivel máximo de tensión de salida del microcontrolador es 3.3 V, por lo que la conexión directa de un pin digital operado con PWM a la entrada del balasto no permite la regulación del flujo en todo el rango posible (1-10 V).

La solución escogida es conectar el pin controlado por PWM a un transistor funcionando en saturación que se encarga de la conmutación según la frecuencia programada. Variando el ciclo de trabajo se modifica el valor medio de tensión en los bornes de las entradas del balasto electrónico. El esquema del circuito descrito se presenta en la siguiente figura:

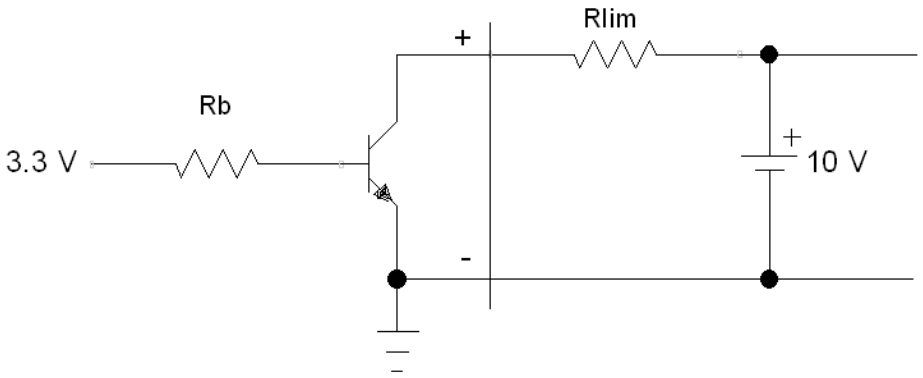


Figura 22. Esquema de regulación del balasto electrónico

El balasto electrónico presenta una limitación de corriente de 0.6 mA por los terminales de control de atenuación (OSRAM, 2014). Esto se utilizará como punto de partida para el dimensionado de la resistencia conectada a la base del transistor, que será el mismo que para el control de encendido y apagado (BC337-40).

El diseño se basa en conseguir que el transistor actúe en la zona saturada. Para ello, cuando el transistor esté en conducción, la corriente por el colector será la máxima permitida por el balasto (0.6 mA).

$$V_{be} = 1.2\,V;\ \beta_{min} = 250;\ I_{c,max} = 0.6\,mA$$

$$I_b = \frac{I_{c,max}}{\beta_{min}} = \frac{0.6\,mA}{250} = 2.4\,\mu A \ \rightarrow R_b = \frac{3.3 - 1.2}{2.4}\frac{V}{\mu A} = 875\,k\Omega$$

El valor de resistencia escogido debe ser inferior al calculado para asegurar el funcionamiento en saturación del transistor. Se escoge para los primeros ensayos una resistencia de 100 kΩ.

Un aspecto a tener en cuenta es que, de forma habitual, las curvas de regulación de flujo luminoso en balastos electrónicos son no lineales, tal como se puede apreciar en la siguiente figura:

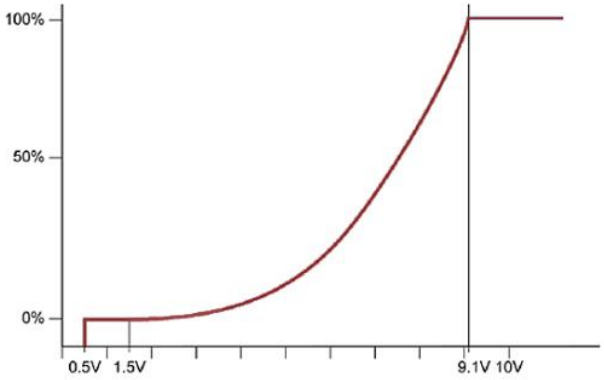


Figura 23. Curva de regulación 0-10 V. Fuente: electroschematics.com/analog-dimming/

Por esta razón es interesante caracterizar el comportamiento del balasto electrónico en función de la modulación por ancho de pulso y su ciclo de trabajo.

Se realiza un ensayo en el que variando el ciclo de trabajo se observa la variación de flujo luminoso y se toman medidas de tensión en los bornes de la conexión 1-10 V. Los datos obtenidos se muestran en la siguiente tabla:

i (8bits)	Duty Cycle	Tensión (V)
0	0,00%	11,38
5	1,96%	10,93
8	3,14%	9,07
9	3,53%	8,57
10	3,92%	7,82
11	4,31%	6,85
12	4,71%	6,02
13	5,10%	4,24
14	5,49%	3,44
15	5,88%	2,82
16	6,27%	2,59
17	6,67%	2,43
18	7,06%	2,3
19	7,45%	2,18
20	7,84%	2,08
25	9,80%	1,85
30	11,76%	1,51
40	15,69%	1,28
50	19,61%	1,16

100	39,22%	1,12
200	78,43%	1,04

Tabla 2. Regulación 1-10 V por PWM

Se observa que para ciclos de trabajo entre 0 y 10 % se produce la mayor variación en la tensión (10-2 V). La regulación no es efectiva en todo el rango posible, por lo que se decide aumentar la resolución del canal utilizado para la modulación por ancho de pulso a 12 bits[2]. Con esto se pretende aumentar el rango de entrada para el que el fondo de escala en la regulación es efectivo.

Un aspecto a destacar es el hecho de que para caracterizar adecuadamente la curva de atenuación del tubo fluorescente a través del balasto electrónico sería necesario contar con algún tipo de sensor o equipo de medida de iluminancia, como por ejemplo un luxómetro. De esa forma, para cada ciclo de trabajo se podría determinar el porcentaje de flujo luminoso emitido por el tubo frente al máximo posible. Para la realización de este trabajo no se cuenta con ningún equipo de estas características, por lo que el ajuste del ciclo de trabajo al comportamiento real de la lámpara se hará mediante la percepción visual de los cambios de luminosidad.

Se realiza un nuevo ensayo, esta vez configurando el canal del PWM con 12 bits. Se busca modificar los ciclos de trabajo observados hasta detectar tramos en los que el cambio de flujo luminoso sea lo más constante posible, es decir, linealizar el comportamiento del tubo frente a las consignas de flujo luminoso.

Se obtienen 20 casos de variación de flujo luminoso equiespaciados, es decir, 20 ciclos de trabajo para los que el cambio de iluminancia entre dos consecutivos es aproximadamente igual. Con esto es posible construir una función que, para cada consigna de flujo luminoso demandado por el usuario, permita obtener el ciclo de trabajo que se debe programar en el microcontrolador. Este razonamiento se aprecia más claramente a partir de la siguiente figura:

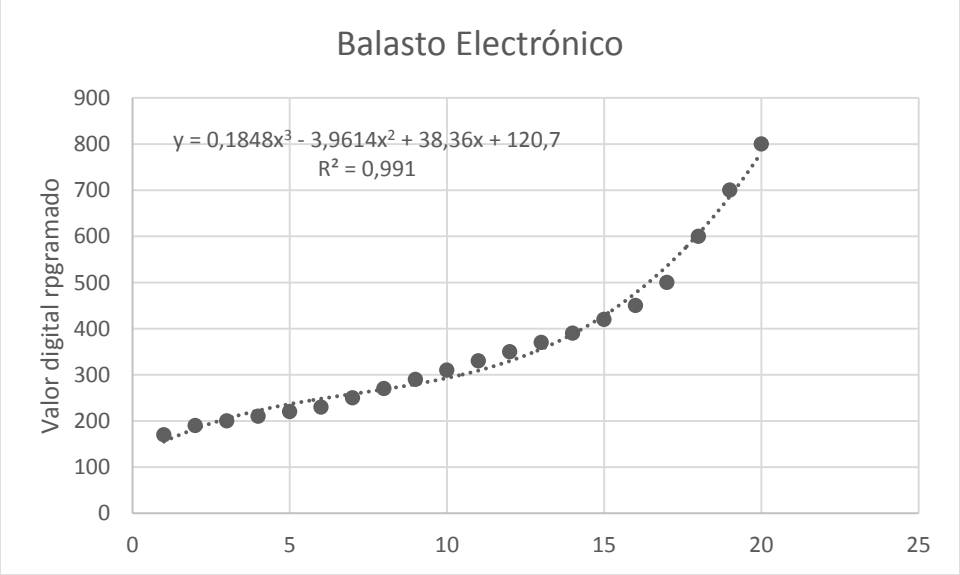


Figura 24. Curva de regulación en el balasto electrónico

[2] El ESP32 permite configurar la resolución del canal de regulación PWM desde 1 a 16 bits.

Lo que muestra la gráfica son los diferentes valores programados en el microcontrolador para obtener 20 niveles equiespaciados de flujo luminoso[3]. El conjunto de puntos obtenidos se aproxima a una función polinómica de grado 3 a través de la herramienta de cálculo Excel, obteniéndose la siguiente ecuación:

$$y = 0.1848 \cdot x^3 - 3.9614 \cdot x^2 + 38.36 \cdot x + 120.7$$

El procedimiento para la determinación del entero a programar para el PWM es el siguiente:

- Primero se divide el porcentaje de atenuación demandada por el usuario entre 5, para obtener la correspondencia sobre 20 muestras disponibles.
- A continuación, se calcula el entero a programar en el ESP32 a partir de la función obtenida.
- Si se desea conocer el ciclo de trabajo programado basta con dividir el entero obtenido en el paso anterior por 4095 (12 bits de resolución).

Para facilitar la comprensión del procedimiento se incluye un sencillo ejemplo numérico:

$$consigna = 80\ \%\ de\ atenuación$$

$$x = \frac{80}{5} = 16$$

$$y = 0.1848 \cdot 16^3 - 3.9614 \cdot 16^2 + 38.36 \cdot 16 + 120.7 = 477$$

$$Duty\ Cycle = \frac{477}{4095} = 11.65\ \%$$

El método descrito se lleva a la práctica y se comprueba que la regulación del flujo luminoso sigue ahora un comportamiento aproximadamente lineal.

Es importante volver a destacar que, al no contar con equipos de medida del nivel de iluminación, la curva de atenuación se basa en la percepción visual del firmante del trabajo. Para un desempeño exacto, se debe calibrar la curva de atenuación con el equipo adecuado antes de poner en funcionamiento el banco de ensayos.

2.6.2. Control del driver led

De forma general, los drivers led pueden ser fuentes de alimentación a tensión constante (para módulos de baja potencia hasta 24 V) o fuentes de alimentación de intensidad constante (para la mayoría de equipos comerciales) (Blanco Espinosa, 2016).

La razón por la que las luminarias led de cierta potencia son alimentadas a partir de una fuente de corriente constante radica en su comportamiento frente a la variación de intensidad. Los módulos led son muy sensibles a las variaciones de corriente. Ante pequeños cambios de la tensión de entrada, la intensidad varía en gran medida, hasta el punto de que podría producirse el fallo por sobreintensidad en caso de no existir límite de corriente.

En el caso concreto del dimmer utilizado en este trabajo, su comportamiento es el de fuente de intensidad constante, con posibilidad de regulación 0-10 V. Este control significa que, para una tensión de 0 V, el flujo luminoso es del 0 % sobre el máximo, y para una tensión de entrada de 10 V, el flujo es del 100 %. El control llevado a cabo por el atenuador se basa en la

[3] Se recuerda que la programación del PWM se ha hecho en base a 12 bits (máximo de 4095).

regulación por corte de fase, modificando el nivel de tensión aplicado al tubo led (GEALed, 2016).

El driver permite tres posibilidades de regulación a través de la entrada 0-10 V (Hoja de características PE18AA 18W Dimming Led Driver, 2019).

La primera es el uso de una resistencia variable de 100 kΩ, obteniendo una regulación directamente proporcional al porcentaje de resistencia aplicado sobre el nominal. Esta opción es descartada ya que no permite el control a distancia del dimmer led.

La segunda opción es realizar una modulación por ancho de pulso directamente sobre los terminales de entrada al driver led. Esta posibilidad tampoco es válida, ya que el nivel de tensión de salida máximo del ESP32 es 3.3 V, con lo que la atenuación no se podría realizar en todo el fondo de escala disponible.

La alternativa escogida es el control por PWM a partir de un pin de salida digital del microcontrolador conectado a la base de un transistor, de forma análoga al circuito planteado para la regulación en el tubo fluorescente.

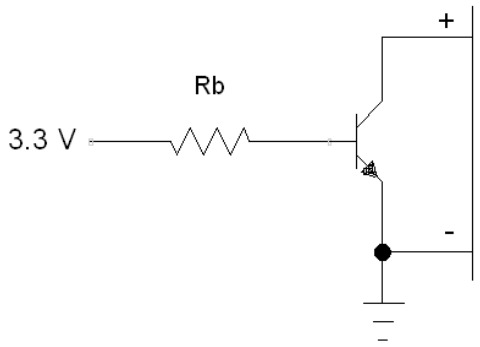


Figura 25. Esquema de regulación del dimmer led

Se plantea el mismo diseño que para la regulación 1-10 V en el balasto electrónico. Como comprobación previa al ensayo de funcionamiento, se calcula la corriente máxima por el colector cuando el transistor está en conducción y se verifica que no sea excesiva.

$$V_{be} = 1.2\,V\,;R_b = 100\,k\Omega\,;\,\beta_{max} = 630$$

$$I_b = \frac{3.3 - 1.2}{100}\frac{V}{k\Omega} = 21\,\mu A\;\rightarrow I_c = 21\,\mu A \cdot 630 = 13.23\,mA$$

La corriente prevista es de 13.23 mA. El fabricante no aporta ningún dato de corriente máxima admisible. No obstante, teniendo en cuenta que la salida de la fuente es de 300 mA, la corriente por los terminales de atenuación supone menos del 5 % de la nominal de salida. A priori, el circuito dimensionado debería ser válido.

Una vez diseñado el circuito de regulación, se realizan diversos ensayos en los que se modifica el ciclo de trabajo de la regulación PWM y se observa la variación en el flujo luminoso. Se recogen los datos obtenidos y se determina la relación entre la regulación efectiva del flujo y el ciclo de trabajo programado, de forma análoga a lo realizado para la regulación del tubo

fluorescente. Para ello, se han registrado los enteros programados[4] en el canal de salida PWM para 12 tramos equiespaciados de variación en el flujo luminoso.

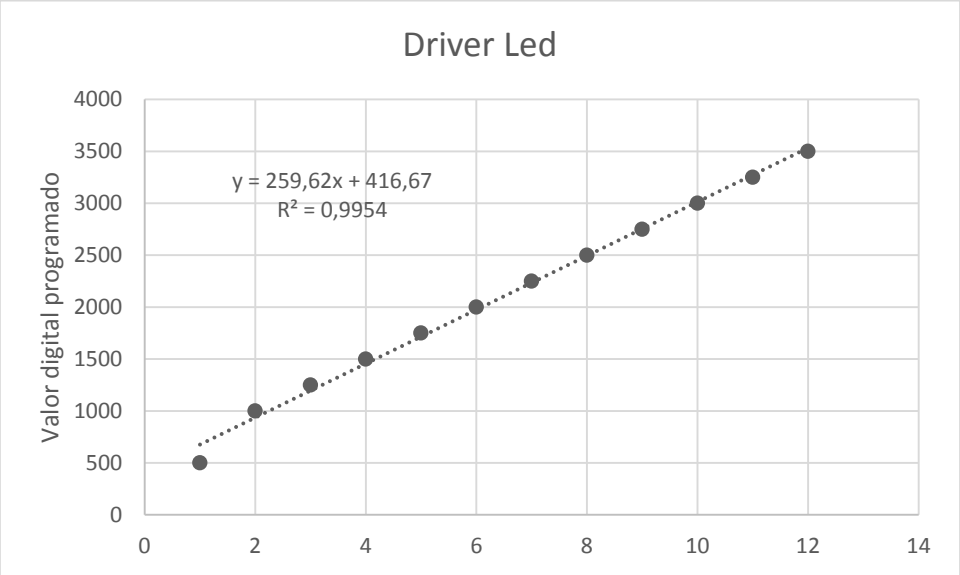


Figura 26. Curva de regulación para el dimmer led

Se aprecia que los puntos obtenidos del ensayo describen un comportamiento prácticamente lineal, apreciándose desviaciones para ciclos de trabajo por debajo del 25 % (entero programado de 1000) y por encima del 85 % (entero programado de 3500). Esto provoca que la recta resultante no pase por el origen de coordenadas. La función que relaciona la consigna establecida por el usuario y el valor a programar en la modulación PWM se muestra en la siguiente ecuación:

$$y = 259.62 \cdot x + 416.67$$

El procedimiento de cálculo del entero de programación se resume en los siguientes puntos:

- Se divide el porcentaje de atenuación demandada por el usuario entre 8.333 (12 muestras).
- Se calcula el valor a programar en el PWM a través de la función obtenida.
- Si se desea obtener el ciclo de trabajo basta con dividir el entero a programar entre 4095 (12 bits de resolución).

A continuación, se muestra un breve ejemplo numérico que ilustra el procedimiento descrito:

$$consigna = 40\ \%\ de\ atenuación$$

$$x = \frac{40}{8.333} = 4.8$$

$$y = 259.62 \cdot 4.8 + 416.67 = 1663$$

$$Duty\ Cycle = \frac{1663}{4095} = 40.61\ \%$$

Se prueba el método descrito y se comprueba su correcto funcionamiento. Por el mismo motivo comentado en el apartado de regulación del tubo fluorescente, para un desempeño

[4] Se establece una resolución del canal PWM de 12 bits.

más ajustado se recomienda la calibración de la curva de regulación a partir de mediciones reales de flujo luminoso.

2.6.3. Integración de la regulación de tubo fluorescente y led

Inicialmente se había planteado el uso de un único circuito de regulación tanto para el tubo led como para el fluorescente controlado por balasto electrónico. Al realizar las pruebas de funcionamiento del control remoto definitivo a partir de la app y del ESP32 se ha comprobado que la regulación del flujo no se produce de forma correcta.

La conexión en paralelo de los terminales de atenuación del balasto electrónico y del dimmer led provoca un comportamiento errático en la regulación del flujo luminoso. Por esta razón, se ha decidido utilizar el mismo circuito de regulación para balasto electrónico y driver led pero de forma duplicada. Esto implica que se utilizan dos circuitos iguales, pero totalmente independientes, para la regulación del tubo fluorescente y del tubo led.

Con la nueva configuración de dos circuitos de regulación independientes, el funcionamiento de la regulación de flujo luminoso es el esperado y el obtenido en los ensayos previos antes de proceder a la integración de conjunto del banco de ensayos.

2.7. Circuito de referencia de tensión

La captura de medidas de tensión y corriente en el banco de ensayos se realiza a partir de la lectura de la tensión de entrada en los pines analógicos del microcontrolador conectados a los convertidores analógico digital. El rango de entrada en el que se pueden realizar medidas es de 0 a 3.3 V, sin poder realizar capturas de valores de tensión negativos. Esto supone un inconveniente en el uso directo del ESP32 que se debe abordar, ya que tanto la tensión como la corriente a medir sigue una señal alterna periódica a 50 Hz.

La solución propuesta consiste en establecer una referencia de tensión flotante de 1.65 V (mitad del rango de entrada). Esto permite que, cuando la tensión o la corriente pasan por cero, la tensión medida en el pin analógico del microcontrolador es 1.65 V y no 0 V. Ajustando los sensores a diseñar de forma adecuada, un valor positivo de tensión o corriente en el banco de ensayos se transforma en un valor de tensión en el pin de entrada entre 1.65 V y 3.3 V, y de forma análoga, un valor negativo de tensión o corriente se convierte en una tensión leída en el canal de entrada de entre 0 y 1.65 V.

En este apartado se aborda el diseño del circuito que proporciona el nivel de tensión de referencia estable de 1.65 V. La solución más directa es diseñar un divisor resistivo que mantenga 1.65 V entre ambas resistencias. Se podría haber planteado utilizar reguladores lineales disipativos o incluso conmutados, pero incrementarían la complejidad del circuito y no supondrían una mejora real, ya que la salida de este circuito no se conecta a una carga, sólo se utiliza como referencia de tensión.

Para reducir las variaciones en la señal y mejorar la estabilidad se plantea el uso de condensadores en paralelo a las resistencias. Los condensadores acumulan energía en forma de campo eléctrico, oponiéndose a la variación de tensión entre sus terminales y reduciendo así el posible rizado de la señal.

La implementación del divisor resistivo se podría realizar a partir de la tensión de salida de la fuente de alimentación de 12 V o a través del terminal de salida a 3.3 V del ESP32. Se ha

optado por la segunda opción por requerir de dos resistencias y dos condensadores iguales, reduciéndose el error derivado de las tolerancias de estos dispositivos, en comparación con la presencia de valores de resistencia y capacidad diferentes a ambos lados del divisor.

El circuito propuesto se muestra en la siguiente figura:

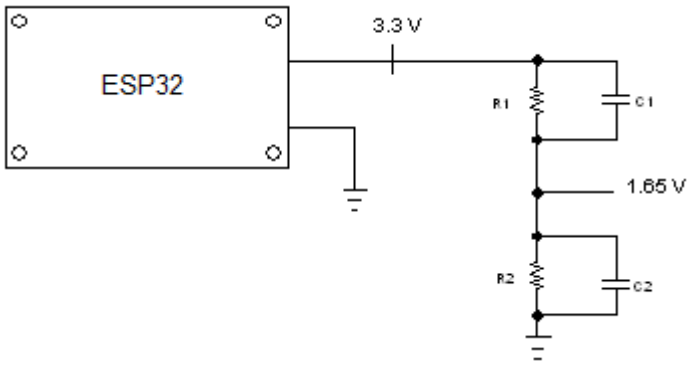


Figura 27. Circuito de referencia de tensión

Como criterio de diseño para el dimensionado de las resistencias se establece que la corriente que circule por el divisor de tensión debe ser la décima parte de la corriente nominal de salida del pin 3.3 V del ESP32, que es 40 mA (Espressif Systems, 2020).

$$\frac{1.65\ V}{3.3\ V} = \frac{R_2}{R_1 + R_2} \leftrightarrow R_1 = R_2$$

$$i = \frac{i_n}{10} = \frac{40\ mA}{10} = \frac{3.3\ V}{2 \cdot R_1} \leftrightarrow R_1 = 412.5\ \Omega \rightarrow \mathrm{R_1 = R_2 = 470\ \Omega}$$

Las resistencias a utilizar, una vez normalizadas al valor disponible en el laboratorio, son de 470 Ω.

El dimensionado de los condensadores se realiza a partir del siguiente esquema:

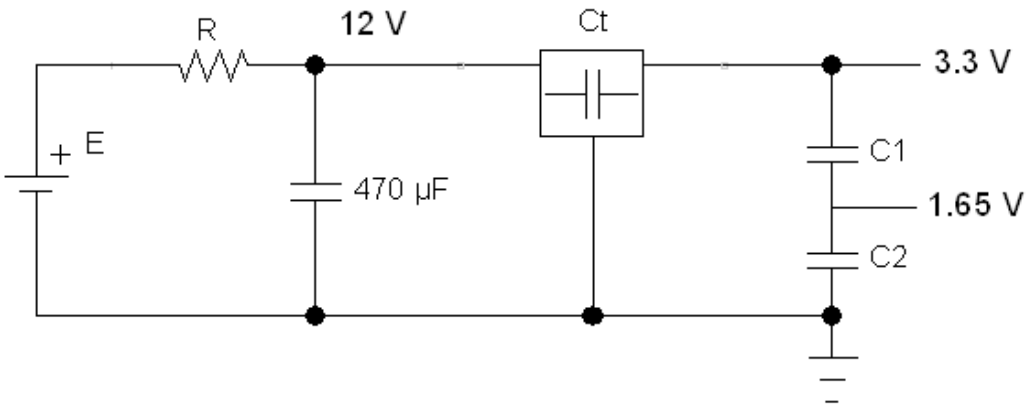


Figura 28. Circuito equivalente dimensionado condensadores

Se utiliza un circuito equivalente simplificado para la fuente de alimentación, consistiendo en una fuente ideal, una resistencia y un condensador de 470 µF (que se corresponde con el condensador real a la salida de la fuente), que proporciona 12 V de salida.

El ESP32 se conecta a la alimentación a 12 V y proporciona una salida de 3.3 V. Esto se consigue gracias a un regulador lineal disipativo (Espressif Systems, 2016). Para simplificar el diseño, se modeliza el regulador como un condensador, de tal forma que la salida de tensión a 3.3 V se consigue con un divisor capacitivo.

El objetivo es obtener las capacidades de los condensadores C_t, C_1 y C_2 que configuran un divisor capacitivo desde 12 V a 3.3 V y 1.65 V respectivamente. Como criterio de diseño se establece que el consumo de corriente desde la fuente de alimentación sea la décima parte de la corriente nominal, o alternativamente, que la capacidad equivalente del divisor capacitivo sea la décima parte de la capacidad del condensador de salida de la fuente, es decir, 47 µF.

Nombrando C_b a la unión en serie de C_1 y C_2, se realizan los siguientes cálculos:

$$\frac{1}{C_{eq}} = \frac{1}{47\ \mu F} = \frac{1}{C_t} + \frac{1}{C_b} \leftrightarrow C_t = \frac{1}{\frac{1}{47} - \frac{1}{C_b}}$$

$$3.3\ V = \frac{C_t}{C_b + C_t} \cdot 12\ V \rightarrow \frac{3.3}{12} = \frac{\frac{1}{\frac{1}{47} - \frac{1}{C_b}}}{\frac{1}{\frac{1}{47} - \frac{1}{C_b}} + C_b} \leftrightarrow C_b = 170.91\ \mu F$$

$$C_t = \frac{1}{\frac{1}{47} - \frac{1}{170.91}} = 64.83\ \mu F$$

Una vez obtenido el valor de capacidad de la asociación en serie C_b, se calculan las capacidades C_1 y C_2.

$$1.65\ V = 3.3\ V \cdot \frac{C_1}{C_1 + C_2} \leftrightarrow C_1 = C_2$$

$$\frac{1}{C_b} = \frac{1}{C_1} + \frac{1}{C_2} = \frac{2}{C_1} \leftrightarrow C_1 = C_2 = 341.82\ \mu F$$

La capacidad más cercana al valor calculado, de los condensadores disponibles en el laboratorio, es 470 µF.

2.8. Sensor de tensión

En este apartado se incluye el diseño del circuito de medida de tensión del banco de ensayos. Previamente, se realiza un análisis comparativo de diferentes tipologías que posibilitan la medida de tensión. La lectura de la medida de tensión se hará a través de una entrada analógica del ESP32, que presenta un rango de tensión de entrada de 0 a 3.3 V.

La medida directa de la tensión de red a través de una entrada del microcontrolador no es posible. Por ello, se plantea la primera alternativa: el uso de un **transformador de tensión**.

Esta solución permite adaptar el nivel de tensión de la red al rango de entrada del ESP32, estableciendo además aislamiento galvánico entre la red y el microcontrolador. No obstante, se deben considerar una serie de peculiaridades.

Los transformadores reales presentan un error de tensión, debido a que la relación de transformación no es exactamente igual a la nominal y un error de fase entre la tensión del

primario y la del secundario. Ambos errores se componen de un error en vacío y de un error en carga (Arteche, 2018).

Para la implementación buscada, la existencia de un error de fase es crítica, ya que afecta tanto a la medida de potencia como al factor de potencia. Esto se debe a que, mientras en la medida de tensión existiría un error de fase (que puede llegar al 5-10 %), en la medida de corriente no hay error de fase, por lo que el cálculo a partir de muestras discretas da resultados erróneos[5].

En aplicaciones reales es posible reducir en gran medida los errores de tensión y de fase con la utilización de transformadores de precisión e instrumentación. Sin embargo, el precio de estos equipos sube considerablemente, con lo que su uso incumpliría una de las premisas básicas en este trabajo, que es la búsqueda de soluciones de bajo coste.

Figura 29. Transformador de precisión. Fuente: mouser.es

La segunda opción se basa en el uso de **convertidores de tensión a frecuencia (VFC)** y aislamiento óptico a través de optoacopladores. El convertidor proporciona una señal de salida a una frecuencia directamente proporcional a la tensión de entrada, tal como se puede apreciar en la siguiente figura:

[5] La obtención de estas variables a partir de valores discretos muestreados se aborda en el capítulo de diseño de software.

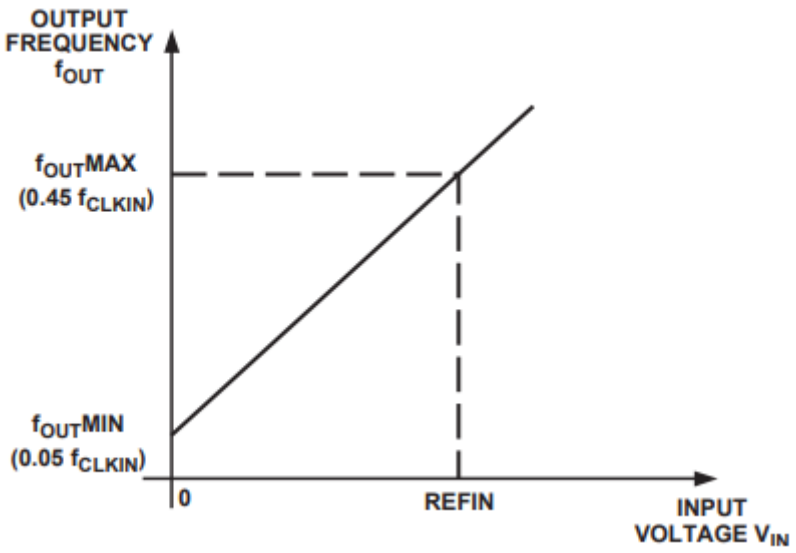


Figura 30. Función de transferencia frecuencia-tensión. Fuente: Hoja de características Single and Multichannel, Synchronous Voltage-to-Frequency Converters AD7741

Una vez obtenida la señal de frecuencia variable, esta se transmite a un pin de entrada digital del ESP32 a través de un optoacoplador, tras atravesar una serie de etapas de acondicionamiento de señal. Finalmente, la detección de la frecuencia y su correspondencia en nivel de tensión se realiza a través del microcontrolador (Egido Nieto, Frías Marín, & Egido Cortés, 2015).

El coste conjunto del convertidor VFC y del optoacoplador se sitúa en torno a 10 €.

La última alternativa analizada es el uso de un **divisor resistivo de tensión**. Esta solución es la más sencilla y la de menor precio de las tres analizadas. Se basa en la incorporación de dos resistencias de alta impedancia a la fase y al neutro, y en serie, entre ambas, una resistencia de medida de menor valor.

La medida de tensión se obtiene como la caída de tensión que se produce en esta resistencia. Para ello, uno de los extremos de esta resistencia se conecta a una entrada analógica del ESP32 y el otro extremo a la referencia flotante de tensión de 1.65 V (valor intermedio del rango de tensión de entrada al microcontrolador admisible).

Esta alternativa presenta el inconveniente frente al resto de propuestas de que no hay aislamiento entre el circuito de medida y la red eléctrica. Sin embargo, al incluir una resistencia de alta impedancia entre fase y circuito de medida (y entre neutro y circuito de medida), si se produjese un contacto de una persona con algún componente del circuito de medida, la intensidad de descarga estaría limitada por esta resistencia. En todo caso, se debe comprobar en el diseño que la corriente de derivación sea inferior a 10 mA, umbral a partir del que la circulación de corriente es peligrosa para el ser humano (Villarubia, 2000).

Teniendo en cuenta las características comentadas de cada alternativa, **se escoge** la implementación del sensor de tensión a partir de un **divisor resistivo**. Si bien la propuesta de

utilizar el VFC junto al optoacoplador es viable, presenta un mayor coste y dificultad de implementación que la opción escogida, por lo que finalmente es descartada[6].

Una vez escogida la opción de sensor de tensión a implementar es necesario analizar las características de lectura de las entradas analógicas del microcontrolador escogido.

El ESP32 dispone de dos convertidores de 12 bits analógico digital, capaces de gestionar 18 canales de entrada simultáneamente (Espressif Systems, 2020). El rango de entrada es de 0 V a 3.3 V, siendo el fondo de escala de 0 a 4095. Es posible configurar el convertidor modificando la resolución o la atenuación en la medida, aunque en este caso se busca aprovechar al máximo el rango de medida del microcontrolador, por lo que estos parámetros se dejarán por defecto.

Un aspecto a tener en cuenta es que el ESP32 muestra un comportamiento no lineal en la medida de la tensión. Esto se muestra en la siguiente figura:

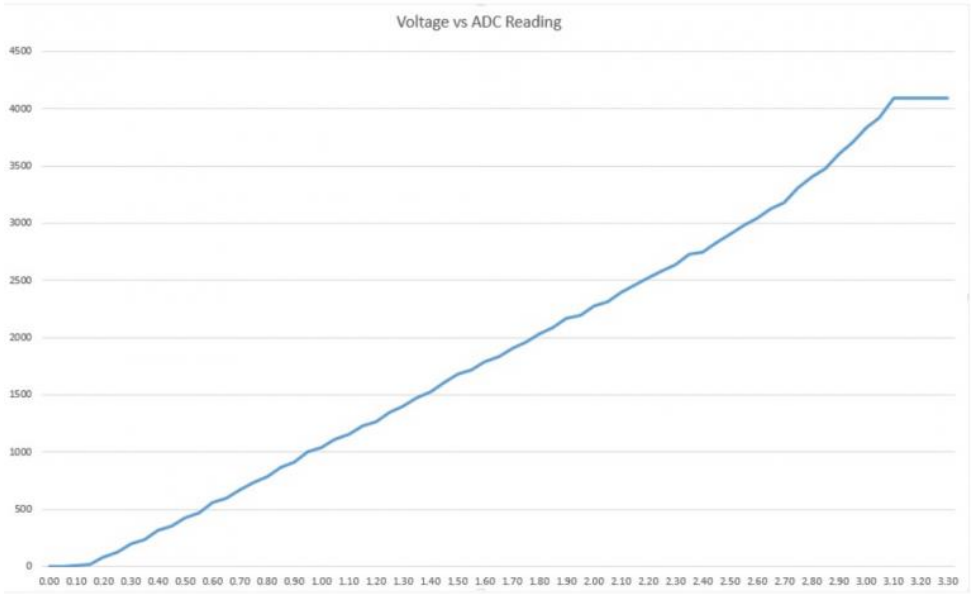


Figura 31. Conversión de tensión del ESP32. Fuente: (randomnerdtutorials, 2020)

Para evitar la lectura errónea de medidas se propone limitar el rango de entrada efectivo, de tal forma que la tensión no esté por debajo de 0.2 V ni por encima de 3.1 V.

Las lecturas de tensión se obtienen a partir del divisor de tensión resistivo que adecúa el nivel de tensión al rango aceptado por el microcontrolador. Ya que la tensión de entrada es alterna y el microcontrolador sólo puede medir valores positivos, es necesario utilizar una referencia de tensión en un punto intermedio del rango de entrada del ESP32 para tomar lecturas tanto de los valores positivos como de los negativos. El cálculo de la referencia de tensión ya se ha abordado en el capítulo 2.7.

La estructura del circuito de medida se muestra en la siguiente figura:

[6] Se recuerda en este punto que el bajo coste del diseño es uno de los aspectos clave de este trabajo.

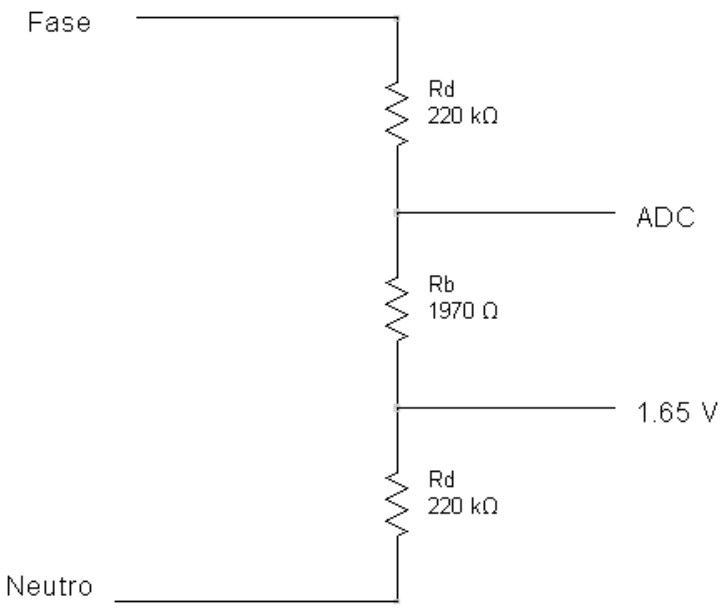


Figura 32. Sensor de tensión

Para el dimensionado de las resistencias del divisor se tienen en cuenta una serie de suposiciones. Como criterio de diseño se establece que las pérdidas de potencia sean mucho menores que la potencia nominal de la resistencia[7] que, para reducir coste y tamaño se elije de ¼ de W: es decir, la potencia consumida debe ser menor de 62.5 mW. Con esto se consigue una muy baja variación de temperatura en servicio y una mayor precisión en la medida de tensión. Por otra parte, se asume que el nivel de tensión de la red es de 230 Vrms en un régimen estacionario sinusoidal. Por último, se diseña el divisor de tal forma que la tensión en el pin de entrada analógico no supere 1.45 V de pico por encima o por debajo de la referencia, a fin de evitar medidas en la zona no lineal del convertidor analógico digital.

A partir de la potencia nominal de pérdidas y considerando que la caída de tensión de 230 Vrms se produce únicamente entre las dos resistencias Rd (ya que la caída de tensión en la resistencia Rb será de apenas 1 Vrms), se obtiene el valor de las resistencias a colocar desde la fase y el neutro de la red.

$$P_{pérdidas} = 62.5\ mW = \frac{115^2}{R_d} \leftrightarrow R_d = 211.6\ k\Omega\ \rightarrow 220\ k\Omega$$

Tras haber normalizado al valor más cercano disponible, se escoge el valor de 220 kΩ.

En el siguiente paso sí se considera la resistencia base sobre la que se mide la tensión, y en la que la caída de tensión máxima debe ser de 1.45 Vp.

$$\frac{\frac{1.45}{\sqrt{2}}}{230} = \frac{R_b}{2 \cdot R_d + R_b} = \frac{R_b}{2 \cdot 220\ k\Omega + R_b} \leftrightarrow R_b = 1970.2\ \Omega\ \rightarrow 1.5\ k\Omega + 470\ \Omega$$

Como resistencia de referencia se escoge una asociación en serie de dos resistencias de 1.5 kΩ y 470 Ω.

[7] La potencia nominal de las resistencias utilizadas es 0.25 W.

El último cálculo de este apartado consiste en comprobar la corriente que podría circular en caso de un contacto directo con el circuito de medida.

$$I_{cd} = \frac{V_{fase}}{R_d} = \frac{230\ V}{220\ k\Omega} = 1.05\ mA < 10\ mA$$

Si se produjese un contacto accidental con el circuito de medida, la corriente que circularía sería inferior a la peligrosa para el cuerpo humano. En todo caso, a fin de evitar contactos directos, se deberá aislar el circuito de medida y el microcontrolador en la medida de lo posible en el montaje final.

2.9. Sensor de corriente

Antes de abordar el diseño concreto de la interfaz es imprescindible realizar una comparativa entre diferentes tecnologías que permitan la captura de medidas de corriente.

La opción más sencilla posible es el uso de una **resistencia de bajo valor**, en serie con la fase a la entrada al banco de ensayos. La intensidad que circula por la resistencia se obtiene midiendo la caída de tensión que se produce en la misma, tal como establece la Ley de Ohm.

Con este sistema, se debe llegar a un compromiso entre la pérdida de potencia en la resistencia y la precisión en la medida. Cuanto más reducido sea el valor de la resistencia, menores son las pérdidas de potencia, pero también la caída de tensión, lo que provoca que la medida sea más sensible al ruido.

En el mercado existen resistencias específicas para el uso en la medida de corriente, cuya tolerancia es menor a la habitual de resistencias de uso general, para reducir el error cometido en la medida. Se pueden adquirir a precios inferiores a 1 €.

Figura 33. Resistencia de medida de corriente "Shunt". Fuente: mouser.es

El principal inconveniente de este método es que no ofrece aislamiento galvánico. Las entradas analógicas del ESP32 no están aisladas, por lo que no se podrían conectar directamente a los extremos de la resistencia de medida. Este aspecto hace descartar totalmente esta opción.

La segunda alternativa que se valora es el uso de un **transformador de corriente**. Este dispositivo consiste en un núcleo magnético atravesado por el conductor del que se quiere medir la intensidad (circuito primario) y por el secundario varias veces (en función de la relación de reducción de corriente).

El circuito secundario se cierra a partir de una resistencia de valor reducido para mantener la proporcionalidad entre circuitos primario y secundario. Esto se debe a que, para resistencias reducidas, el secundario presenta un comportamiento cercano al cortocircuito, reduciendo la corriente de magnetización.

La medida de la corriente se realiza a través de la resistencia de cierre del secundario, midiendo la caída de tensión entre sus terminales. Para obtener la intensidad que circula por el circuito primario basta con deshacer la relación de reducción de corriente del transformador.

Esta tipología sí proporciona aislamiento galvánico entre el circuito de potencia y el de medida. De esta forma es posible conectar un extremo de la resistencia de cierre a una entrada analógica del ESP32 y el otro a la referencia de tensión.

El uso de transformadores de corriente está limitado a circuitos de corriente alterna, debido a que su principio de funcionamiento se basa en la inducción electromagnética. Esto no supone ningún inconveniente, ya que el banco de ensayos sobre el que se requiere la toma de medidas está conectado a corriente alterna.

En función del rango de frecuencias de la corriente a medir existen diferentes configuraciones de transformadores. Para el caso de este trabajo, 50 Hz, se puede encontrar una amplia gama de transformadores por entre 1 y 5€.

La tercera alternativa que se valora es el uso de un **sensor de efecto Hall**. El sensor está formado por un núcleo magnético que concentra las líneas de flujo y rodea a un conductor. En una sección intermedia del núcleo se ubica una placa metálica, por la que se hace circular una corriente de control, de forma que las líneas de campo magnético sean perpendiculares (García del Buey, Martínez Román, & Sapena Bañó, 2019).

El principio de funcionamiento se basa en la fuerza de Lorentz, que establece que una carga eléctrica en movimiento (o un conductor por el que circula corriente) es sometida a una fuerza al atravesar un campo magnético.

$$\vec{F} = q \cdot (\vec{v} \times \vec{B} + \vec{E})$$

La corriente a medir se hace pasar por el bobinado, produciendo un campo magnético. Este campo altera la circulación de corriente sobre la placa conductora del interior del sensor y provoca un gradiente de tensión dentro de la placa, debido a la distribución desigual de la carga. Esta diferencia de potencial es proporcional a la intensidad de campo magnético, y por tanto a la corriente a medir.

En la siguiente imagen se puede apreciar el efecto de la aparición del campo magnético sobre la placa conductora.

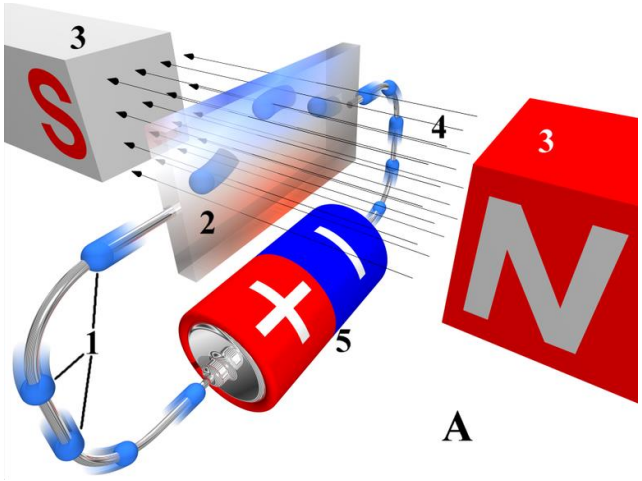


Figura 34- Efecto Hall. Fuente: Wikipedia.org

Existen sensores de efecto Hall para medición de campo magnético o corriente. La diferencia está en si el campo magnético, que afecta a la corriente de control por la placa, está producido por un campo magnético externo o la corriente a medir.

Figura 35. Sensor de corriente de efecto Hall. Fuente: farnell.com

La mayor ventaja de este sensor, en relación con el transformador de corriente, es que permite la medida de corriente continua, además de alterna. Para este trabajo se debe medir corriente alterna, por lo que ese hecho no supone una ventaja real.

El precio de un sensor de corriente de efecto Hall varía entre 10 y 20 €. Este rango de precios supera ampliamente al de un transformador de corriente para las características demandadas en este trabajo. Esto, unido al hecho de que, ajustando la resistencia de cierre en el transformador de corriente a un valor reducido la precisión en la medida es elevada, hace que la **opción finalmente escogida** sea el uso de un **transformador de corriente**.

El transformador de corriente escogido es el "Talema AX-0500", apropiado para la medida de corriente alterna a 50/60 Hz y con una relación de reducción de 500:1.

Figura 36. Transformador de corriente Talema AX-0500

El circuito de medida propuesto es el siguiente:

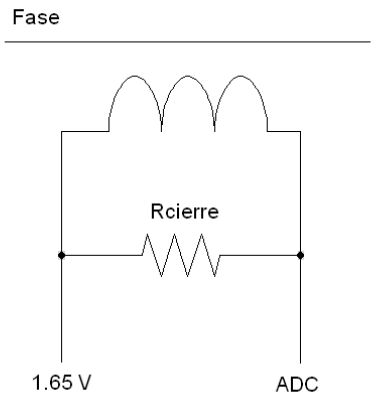


Figura 37. Circuito de medida de corriente

Para poder solventar la imposibilidad de medir tensiones negativas por parte del ESP32, se establece 1.65 V de referencia. De esta forma, las medidas corriente positivas o negativas se traducen en tensiones siempre positivas.

De forma análoga a lo establecido para la medida de tensión, la caída de tensión máxima en la resistencia de cierre del transformador de corriente no debe superar 1.45 Vp, a fin de evitar entrar en la zona no lineal de medida del microcontrolador.

La situación más desfavorable en términos de mayor circulación de corriente se produce durante el funcionamiento del tubo fluorescente con balasto convencional. Se prevé una corriente eficaz cercana a 0.5 A, tal como se ha comprobado en ensayos anteriores con la misma tipología (Martínez La Osa, Sapena Bañó, & Martínez Román, 2018). En este caso, y teniendo en cuenta la relación de transformación de 500:1, la corriente que circularía por el secundario sería de 1 mA, lo que obligaría a disponer de una resistencia de cierre muy elevada para aprovechar el rango de entrada disponible en el convertidor analógico digital. Una resistencia de cierre elevada provoca un aumento de la corriente de magnetización y en consecuencia un incremento del error en la medida, tal como se puede apreciar en la siguiente fórmula (Arteche, 2018):

$$\epsilon_i(\%) = 450000 \cdot \frac{L \cdot Z_t}{{N_s}^2 \cdot S \cdot \mu}$$

siendo:

$$L = Longitud\ media\ del\ circuito\ magnético\ (cm)$$

$$Z_t = Impedancia\ total\ del\ secundario\ (\Omega)$$

$$N_s = Número\ de\ espiras\ de\ la\ bobina\ en\ el\ secundario$$

$$S = Sección\ del\ núcleo\ magnético\ (cm^2)$$

$$\mu = Permeabilidad\ de\ la\ chapa\ magnética\ (\frac{G\ cm}{A\ V})$$

Se plantea reducir la relación de transformación. Dando 10 vueltas al primario, se consigue una relación de 50:1.

$$\frac{N_1}{N_2} = \frac{I_2}{I_1} = \frac{10}{500} = \frac{1}{50}$$

La resistencia de cierre se dimensiona para el caso más desfavorable de circulación de corriente (fluorescente con balasto convencional), a fin de evitar sobrepasar el rango de entrada del ESP32.

$$\frac{I_2}{I_1} = \frac{I_2}{0.5\ A} = \frac{1}{50} \leftrightarrow I_2 = 10\ mA$$

$$V_{p,max} = 1.45\ V$$

$$V_{rms} = R_{cierre} \cdot I_{rms} = \frac{1.45}{\sqrt{2}} = R_{cierre} \cdot 10\ mA \leftrightarrow R_{cierre} = 102.53\ \Omega \rightarrow R_{cierre} = 100\ \Omega$$

El valor de resistencia normalizada más cercano al valor teórico es 100 Ω.

2.10. Integración de los circuitos y montaje final

En este apartado se presenta el montaje completo, instalado en placa de prototipos. Debido a la situación extraordinaria derivada de la pandemia de COVID-19, y las restricciones de acceso al laboratorio, el montaje final se realiza sobre protoboards y no en placa de circuito impreso (PCB).

El esquema de conexionado del conjunto de sistemas se muestra en la siguiente figura[8]:

[8] La posición de cada elemento en el esquema no se corresponde con la ubicación real en el montaje.

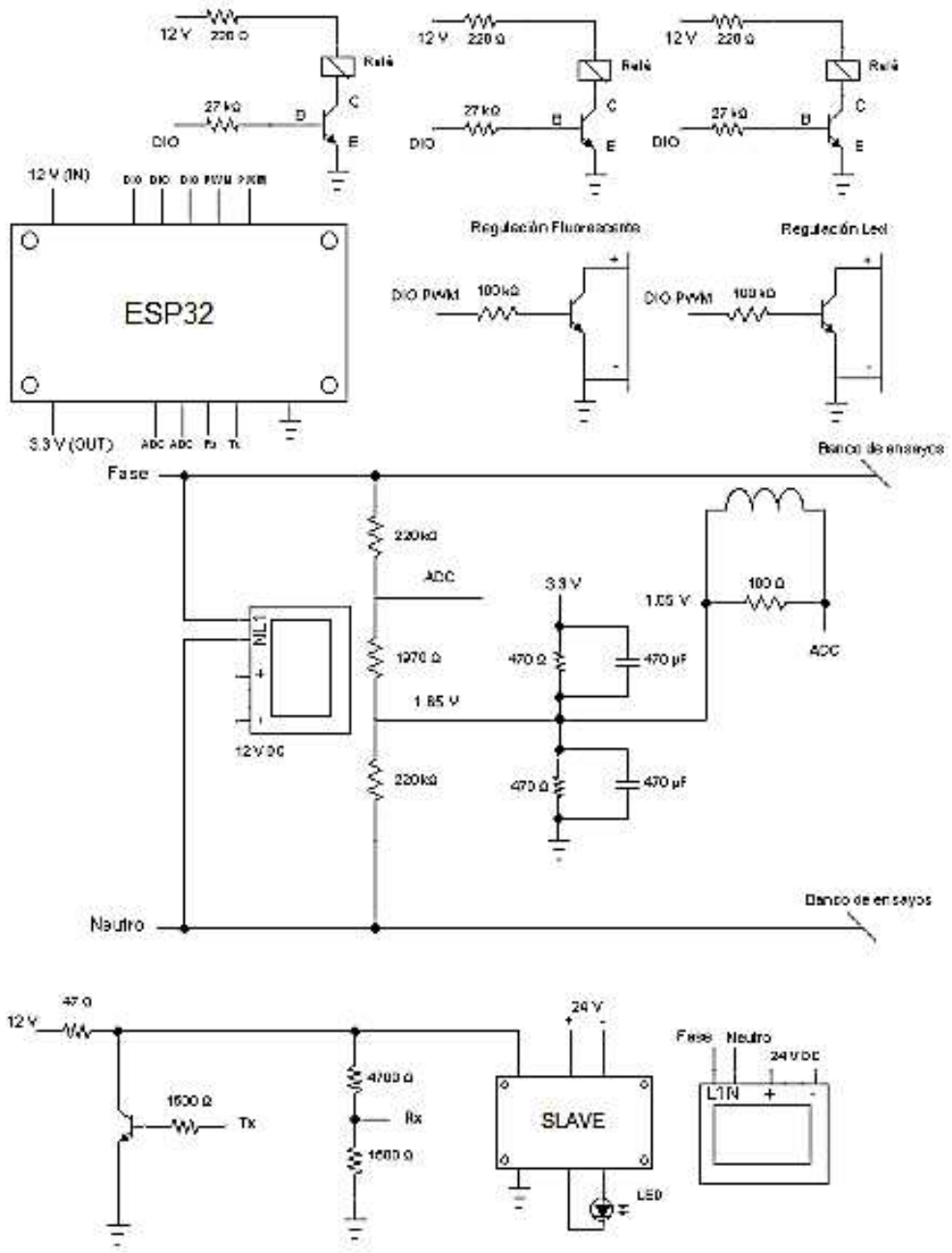


Figura 38. Esquema de integración de circuitos

En la anterior figura se incluyen:

- 3 circuitos de control de encendido/apagado de cada tubo
- 2 circuitos de regulación del flujo luminoso del tubo fluorescente con balasto electrónico y del tubo led
- Sensor de tensión
- Sensor de corriente
- Circuito de referencia de tensión
- Fuente de alimentación de 12 V
- Bus DALI con circuitos de transmisión y recepción de señales, esclavo DALI y fuente de alimentación de 24 V del esclavo

El resultado del montaje final se muestra en la siguiente imagen:

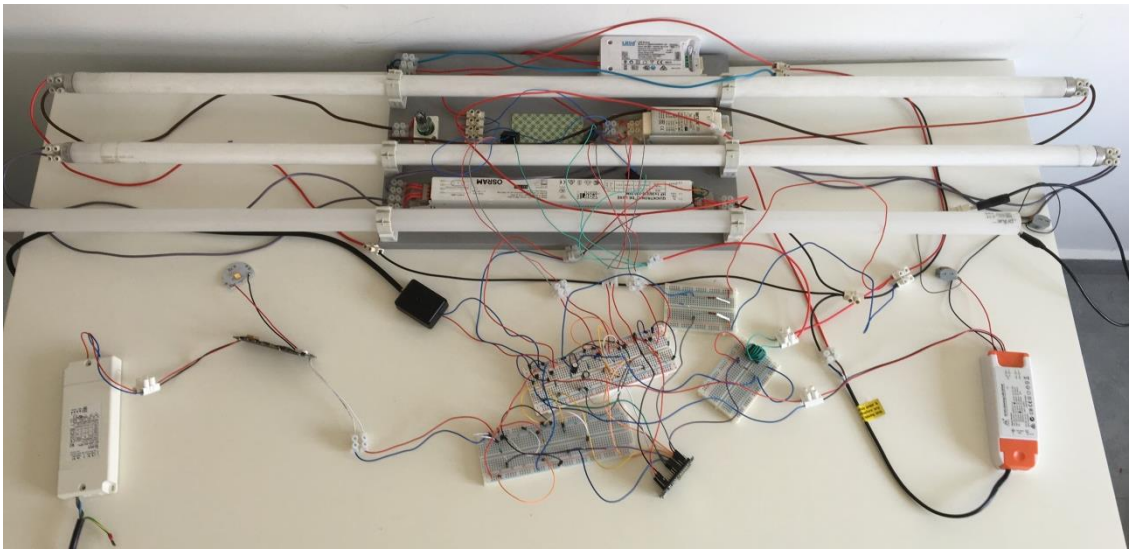


Figura 39. Montaje final

Para poder apreciar con más detalle la zona más crítica de conexionado, se presenta una nueva vista del montaje centrándose en la zona de las protoboards.

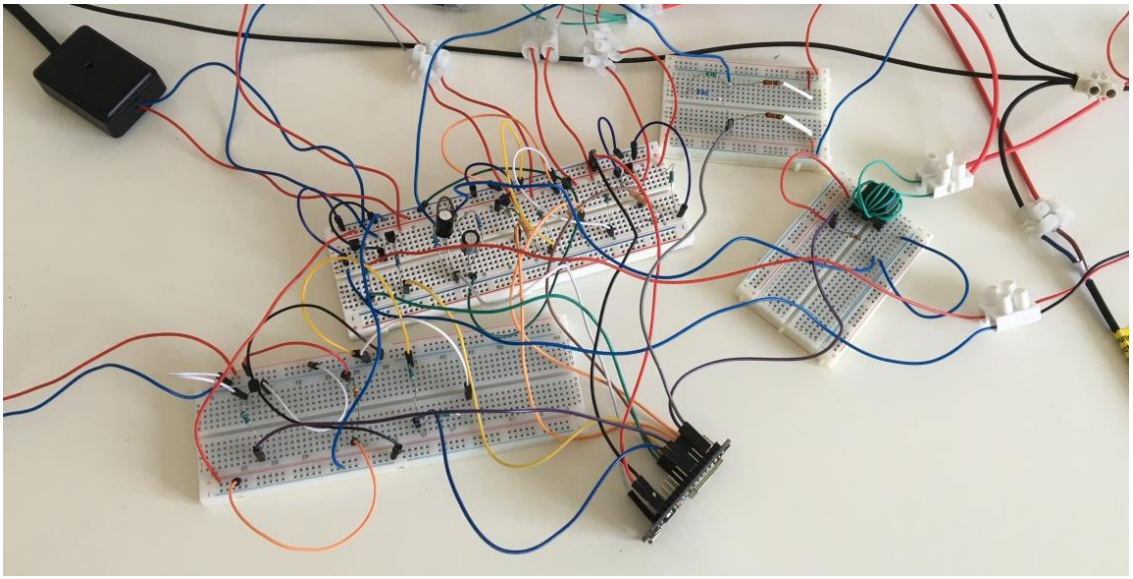


Figura 40. Montaje final (zona de placas de prototipos ampliada)

Se puede apreciar el sensor de tensión en la protoboard de pequeño tamaño en la parte superior de la imagen. Las dos resistencias de la derecha están conectadas a la fase y neutro de la red, en paralelo al banco de ensayos. Las dos resistencias de la izquierda configuran la resistencia de medida de la caída de tensión.

La otra protoboard de tamaño reducido contiene al sensor de corriente. Se puede apreciar el transformador de corriente con las 10 vueltas en el primario. La caída de tensión se mide en la resistencia de cierre.

Tanto al circuito de medida de tensión como de corriente llegan dos cables (además de la conexión con el circuito de potencia): uno establece la referencia de tensión flotante y el otro se conecta a la entrada con conversión A/D del ESP32.

En la protoboard de mayor tamaño situada en el centro de la imagen se ubican los 3 circuitos de encendido/apagado (zona de la derecha), el circuito que proporciona la tensión de referencia para las medidas (zona central), y los dos circuitos de regulación del flujo luminoso (zona de la izquierda).

Por último, en la protoboard de la zona inferior de la imagen se encuentra el bus DALI. La alimentación se obtiene de la protoboard superior. En la zona central se ubica la resistencia limitadora de corriente de cortocircuito, en la parte izquierda la rama de transmisión de señal y en la parte derecha la rama de recepción de señal.

3. DALI

DALI (Digital Addressable Lighting Interface) es un protocolo de comunicación entre dispositivos de iluminación estandarizado a nivel internacional. Está basado en la norma IEC[9] 62386 (el estándar europeo equivalente es la Norma europea EN 62386).

3.1. Introducción al protocolo DALI

El protocolo DALI configura un bus de comunicaciones entre dispositivos maestros y esclavos. Permite conectar diversos dispositivos de iluminación y control electrónico como balastos regulables, módulos led, sistemas de alumbrado de emergencia, transformadores, módulos de relés, etc, y gestionar cualquier tipo de luminaria cuyo control y regulación se realice a través de un balasto electrónico o un dimmer compatible con las especificaciones DALI.

Algunos ejemplos de dispositivos DALI se muestran en la siguiente figura:

Figura 41. Relé, balasto electrónico para fluorescentes y dimmer led DALI. Fuente: (Tridonic)

El estándar DALI está bajo el amparo de la Digital Illumination Interface Alliance, consorcio de compañías que promocionan y engloban los esfuerzos de investigación de esta tecnología.

El protocolo DALI posibilita una serie de funcionalidades:

- Encendido/apagado
- Regulación de la intensidad luminosa
- Agrupación de distintas luminarias bajo unas consignas comunes
- Control individualizado de luminarias

Este protocolo surge como una evolución del sistema tradicional de regulación 1-10V, permitiendo una mayor flexibilidad en la conexión. Por ejemplo, es posible controlar el nivel de iluminación de las luminarias en función de la hora del día para mejorar la eficiencia o establecer diferentes niveles según el área en la que se ubique el equipo. Se trata de una herramienta muy versátil que posibilita un control minucioso de cada luminaria, con el objetivo de ajustar adecuadamente el nivel de iluminación al requerido aumentando la eficiencia de la instalación.

Las especificaciones generales de la interfaz DALI son las siguientes:

- Protocolo de transmisión asíncrona.
- Tasa de información de 1200 bit/s.
- Caída de tensión máxima de 2 V entre dispositivos conectados al bus.
- Tolerancias en las especificaciones temporales de ±10%.

[9] International Electrotechnical Commission

- Dispositivo esclavo (control gear) no puede actuar como maestro.
- Máximo de 64 dispositivos esclavos asignables con dirección individual.
- Máximo de 16 grupos de dispositivos direccionables por comandos de grupo.
- Parámetros almacenados en la memoria del dispositivo intercambiables.

Además de la flexibilidad, presenta gran facilidad de conexión. Sólo es necesario conectar los dispositivos mediante los dos cables DALI sin polaridad, que de forma general se incluyen dentro de un mismo aislante junto a la fase, neutro y tierra.

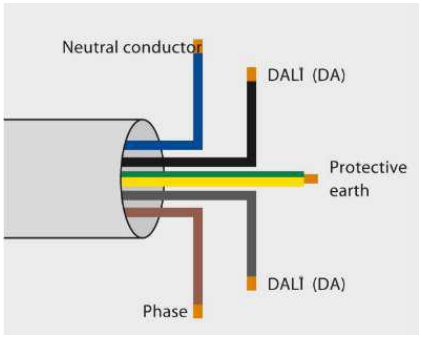


Figura 42. Cable multihilo con bus DALI (Martínez Pérez & López Antón, 2016)

La máxima caída de tensión en el bus DALI no debe ser superior a 2 V cuando circula la máxima corriente (250 mA). Esto limita la sección a utilizar. Para un cable de cobre de 1.5 mm^2, la longitud máxima no debe ser superior a 300 m. Para distancias inferiores, es posible utilizar conductores de menor sección, siempre que la caída de tensión no supere 2 V. Con un cable de cobre de una sección de 0.14 mm^2, la longitud máxima del bus sería 31 m a 25 ^{0}C. En el entorno de este trabajo, la longitud del cableado no es crítica y no se tendrá en cuenta.

3.2. Especificaciones eléctricas del bus DALI

Los niveles de tensión que se asocian a un nivel alto o bajo son los siguientes:

- Nivel alto:
  - Como receptor: 9.5 a 22.5 V
  - Como transmisor: 11.5 a 20.5 V
- Nivel bajo:
  - Como receptor: -6.5 a 6.5 V
  - Como transmisor: -4.5 a 4.5 V

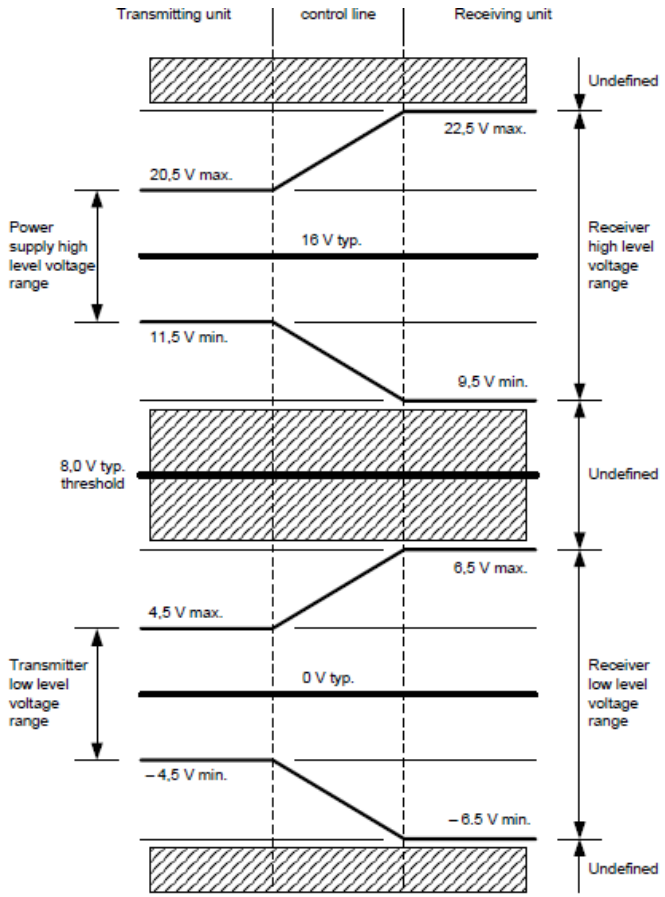


Figura 43. Niveles de tensión del bus DALI. Fuente: (AENOR)

En cuanto a la corriente, se deben cumplir las siguientes especificaciones:

- En estado no activo (sin transmitir), los dispositivos pueden consumir un máximo de 2 mA.
- En estado activo, tanto maestro como esclavo deben ser capaces de drenar 250 mA.

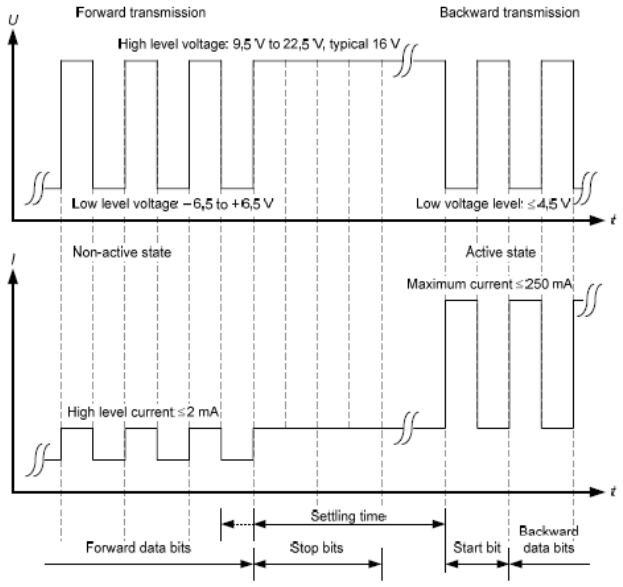


Figura 44.Niveles de tensión y corriente para tramas forward y backward en el esclavo. Fuente: (AENOR)

3.3. Conexionado e inicialización

En un único bus es posible conectar hasta 64 dispositivos esclavos con direcciones individuales simultáneamente.

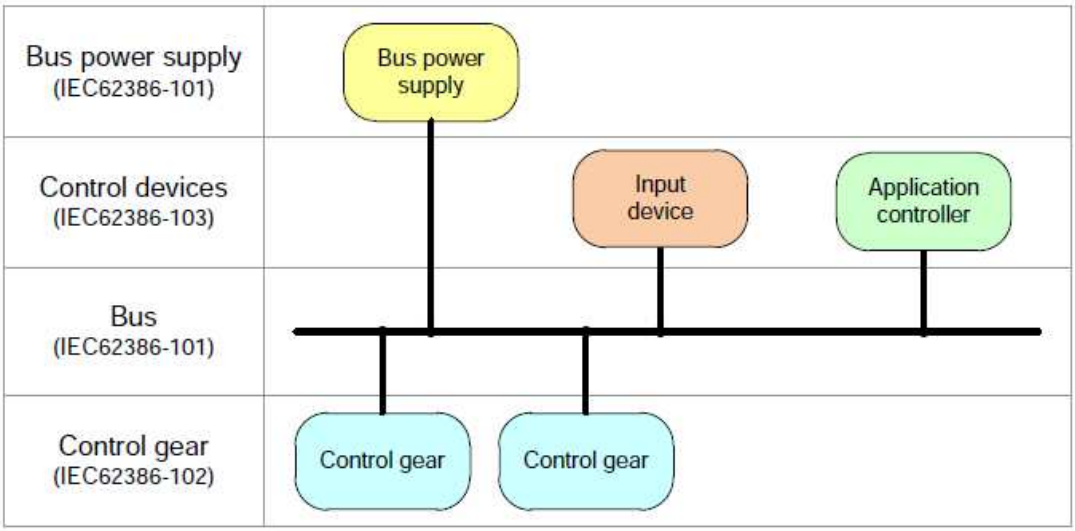


Figura 45. Esquema de conexionado bus DALI con dos dispositivos esclavos. Fuente: (AENOR)

La puesta en marcha de los equipos conectados al bus está realizada por el maestro. Inicialmente no hay ninguna comunicación por el bus y es el maestro el que inicia el envío de comandos, mediante la asignación de direcciones individuales, grupos y escenas, así como el encendido o apagado de todos los dispositivos a través de comandos de difusión.

3.4. Operación del bus DALI

Los dispositivos DALI realizan el envío de señales a través del cortocircuito del bus (nivel bajo de tensión) y la apertura del mismo (nivel alto de tensión).

Los tiempos de conmutación entre estados en el bus deben estar dentro de unos límites. Tanto para el tiempo de subida como el de bajada el rango debe ser de 3us a 25 us en el caso de un único dispositivo maestro y de 3us a 15 ms si hay más de un maestro.

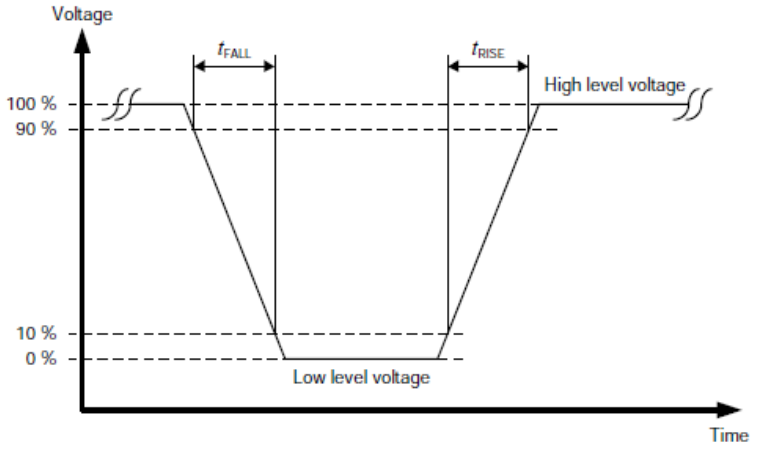


Figura 46. Tiempos de subida y de bajada en cambios de estado. Fuente: (AENOR)

Se utiliza la codificación Manchester, un método de codificación autosincronizada. Las señales de reloj y de datos se combinan en una sola y la lectura de la información se realiza a través de un cambio de estado en la mitad del intervalo de duración de cada bit. El flanco de subida se traduce en un 1 lógico y el flanco de bajado en un 0 lógico.

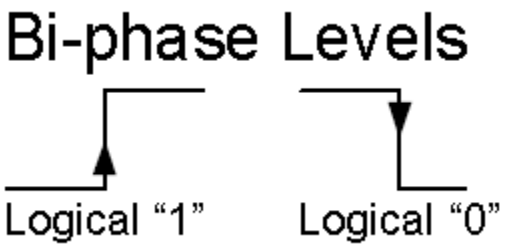


Figura 47. Codificación Manchester (Martínez Pérez & López Antón, 2016)

La información se transmite a partir de tramas de bits. Existen dos tipos de tramas, la trama forward (enviada de un dispositivo maestro a un esclavo) y tramas backward (respuesta de esclavo a una petición de información por parte del maestro). El bus en reposo se encuentra en el nivel alto de tensión.

La trama forward está formada por un bit de inicio, 16 bits (1 byte de direccionamiento y 1 byte de datos) y 4 bits de parada. La velocidad del bus es 1200 Bps.

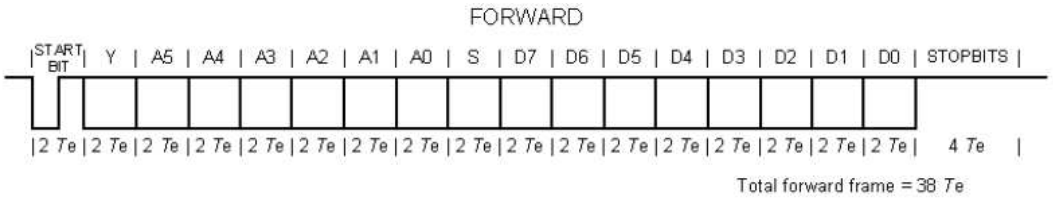


Figura 48. Trama Forward (Martínez Pérez & López Antón, 2016)

La duración de cada semi intervalo de bit (semiperiodo) es:

$$T_e = \frac{1}{2 \cdot 1200} = 416.7\ \mu s$$

El bit de inicio es un 1 lógico, es decir, un flanco de subida a mitad del periodo. Indica el inicio de la trama, y por tanto el origen de tiempos para decodificar la información contenida en esa trama.

Los 4 bits de parada mantienen el bus a nivel alto durante cuatro periodos. Marcan el fin de la trama.

El tiempo total de una trama forward es:

$$T_{forward} = 38 \cdot T = 15.83\ ms$$

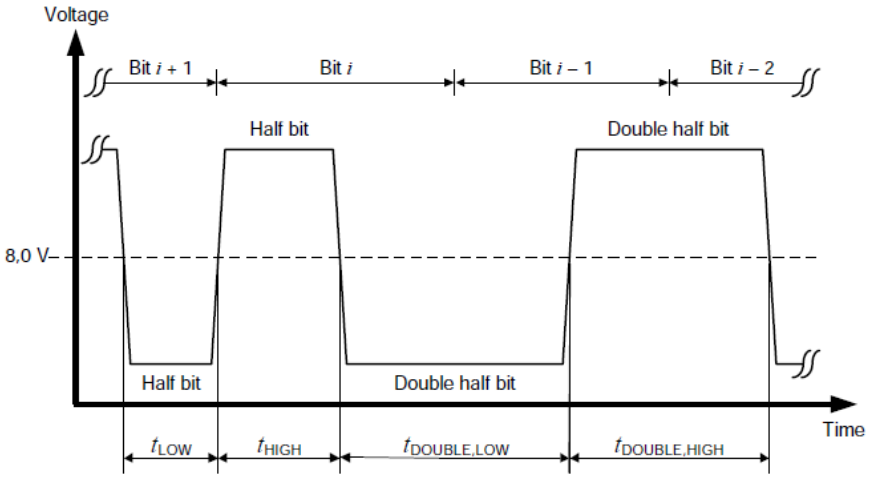


Figura 49. Ejemplo de distribución de tiempos en el envío. Fuente: (AENOR)

La temporización debe estar dentro de un rango admisible. De forma general, si un dispositivo detecta un envío que no se ajusta al número de bits o a los rangos de tiempos admisibles, descartará esa trama de información.

	Minimum	Typical	Maximum
Half bit time t_{HIGH}, t_{LOW}	366,7 µs	416,7 µs	466,7 µs
Double half bit time $t_{DOUBLE,LOW}$, $t_{DOUBLE,HIGH}$	733,3 µs	833,3 µs	933,3 µs
Stop condition time T_{STOP}	2 450 µs		

Figura 50. Rangos de tiempo admisibles entre estados opuestos. Fuente: (AENOR)

La trama backward está constituida por el bit de inicio, 8 bits de datos y los 4 bits de parada.

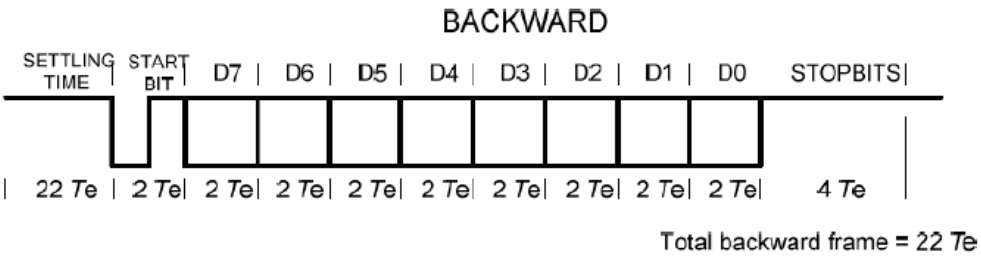


Figura 51. Trama backward (Martínez Pérez & López Antón, 2016)

El tiempo total de una trama backward es:

$$T_{backward} = 22 \cdot T = 9.17\ ms$$

Para realizar una comunicación efectiva es necesario establecer tiempos de espera entre la emisión de nuevas tramas de datos:

- Entre dos tramas forward consecutivas debe existir una espera mínima de 22 Te.
- Entre una trama forward y una backward la espera debe estar comprendida entre 7 Te y 22 Te.
- Entre una trama backward y una forward el tiempo de espera debe ser de al menos 22 Te.

4 tramas forward con los tiempos de espera mínimos entre envíos suponen un tiempo total de 100 ms.

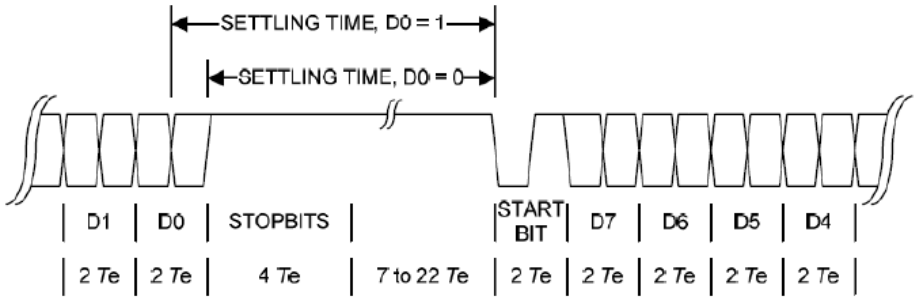


Figura 52. Settling time entre tramas forward-backward (Martínez Pérez & López Antón, 2016)

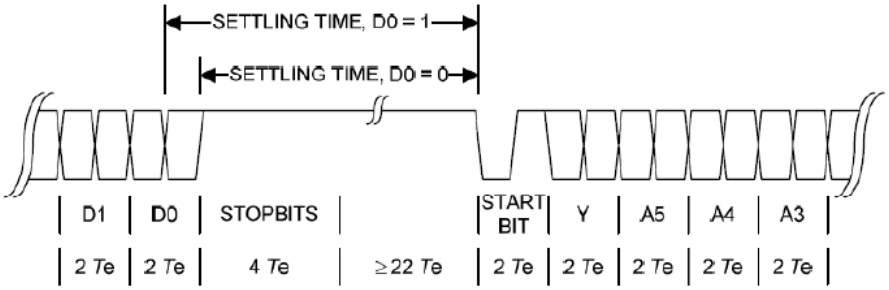


Figura 53. Settling time entre tramas backward-forward (Martínez Pérez & López Antón, 2016)

Para reducir la probabilidad de que se produzcan colisiones entre envíos de información de varios maestros, se establecen tiempos de espera en función de la prioridad asignada a cada dispositivo, superiores a los mínimos correspondientes al caso de un solo maestro. A mayor prioridad, el tiempo de espera es menor.

En la siguiente figura se muestran estos tiempos de espera:

Settling time between	Minimum	Typical	Maximum
Forward frame and backward frame [a]	5,5 ms		10,5 ms
Any frame and forward frame (priority 1) [b]	13,5 ms	[c]	14,7 ms [d]
Any frame and forward frame (priority 2) [b]	14,9 ms	[c]	16,1 ms [d]
Any frame and forward frame (priority 3) [b]	16,3 ms	[c]	17,7 ms [d]
Any frame and forward frame (priority 4) [b]	17,9 ms	[c]	19,3 ms [d]
Any frame and forward frame (priority 5) [b]	19,5 ms	[c]	21,1 ms [d]

[a] This is a delay time where the backward frame can start regardless of transitions in between.

[b] Also applicable after overlapping backward frames causing a receiver bit timing violation or a receiver frame size violation.

[c] It is strongly recommended that a multi-master transmitter starts its transmission at a random point of time within the minimum and maximum settling time corresponding to the intended priority, as this helps in avoiding collisions. Clock tolerances need to be considered.

[d] When a multi-master transmitter intends to send a frame with a certain priority but the maximum settling time for this priority has already passed, the transmitter can start its transmission immediately considering collision avoidance.

Figura 54. Tiempos de establecimiento para la tranmisión multi maestro

Las colisiones no pueden ser evitadas en todas las situaciones. La norma EN 62386 establece unos protocolos de actuación en los maestros basados en la interrupción inmediata del mensaje enviado y la medición de los anchos de pulso de las transiciones que permanecen en el bus, si el envío por parte de otro dispositivo permanece. Como en el entorno de este trabajo sólo se usará un único maestro en el bus, no se entrará en más detalle sobre la gestión de colisiones entre envíos de maestros.

3.5. Tipos de comandos en la comunicación DALI

El envío de información de maestro a esclavo se puede realizar de tres formas principales:

Comandos de difusión (broadcast):

El maestro envía una orden cuyos destinatarios son todos los dispositivos esclavos que están conectados al bus, que deben actuar a la vez. Este tipo de comandos se utilizan en la inicialización de los sistemas y cuando se desea que todos los dispositivos reaccionen de la misma forma.

Comandos de grupo:

Dentro de un único bus DALI es posible designar 16 grupos diferentes. La asignación de un balasto a un grupo se realiza guardando en la memoria del balasto a qué grupo pertenece (0-15). Cuando un maestro envíe un comando de grupo, todos los esclavos asignados a ese grupo deben responder a la orden enviada. Este tipo de comandos se utiliza para el control de iluminación de diferentes áreas de forma totalmente independiente a través de un único bus.

Comandos individuales:

Una de las características que otorga gran flexibilidad al protocolo DALI es la posibilidad de controlar de manera totalmente independiente cada uno de los balastos conectados al bus. Esto se realiza través de comandos con direcciones individuales. En un bus puede haber hasta 64 direcciones diferentes (0-63). De forma similar a lo que ocurre con los comandos de grupo, es necesario inicializar el balasto y asignarle una dirección única, que queda guardada en la memoria del dispositivo. La diferencia con respecto a los comandos de grupo es que cada balasto debe tener una dirección diferente. Cuando el maestro envía un comando individual, sólo responde el balasto cuya dirección se especifique en la trama forward.

Comandos de escena:

Este comando difiere con respecto a los anteriores en que no sirve para determinar qué balastos deben responder a una petición, sino que establece un nivel de iluminación que ha sido pregrabado en la memoria del esclavo. Existen 16 escenas diferentes (0-15) que se corresponden con niveles determinados de iluminación, que deben haber sido guardados en la memoria del balasto previamente. Es posible establecer la escena en la que debe funcionar un esclavo a través de los tres tipos de comandos de direccionamiento definidos anteriormente (difusión, grupo e individual).

En cuanto a la entidad del mensaje, se distinguen diferentes tipos de información enviada por el bus de comunicación:

- **Comandos de alimentación:** Indican el nivel de alimentación de la luminaria, y por tanto el flujo luminoso.

- **Comandos de configuración:** Permiten configurar el funcionamiento del balasto. Deben ser enviados dos veces en menos de 100 ms para que no sean ignorados por los balastos.
- **Comandos de consulta:** El maestro solicita algún tipo de información al esclavo, como la dirección o el nivel de iluminación.
- **Comandos especiales:** Se utilizan para inicializar los balastos. Deben ser enviados dos veces en menos de 100 ms para no ser ignorados y sólo son escuchados durante los primeros 15 minutos desde que el maestro haya enviado la instrucción de inicialización.

3.6. Regulación de la intensidad luminosa

Por defecto, los balastos DALI pueden controlar la intensidad luminosa mediante una curva de ajuste logarítmico. También es posible configurar una curva de atenuación lineal, aunque para este trabajo se ha utilizado la curva logarítmica. La justificación es que el ajuste logarítmico proporciona un mejor control de la intensidad luminosa, en relación a la percepción del ojo humano.

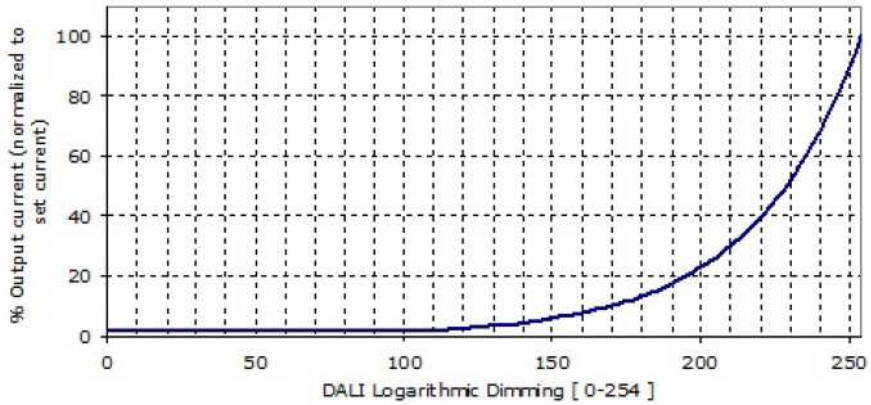


Figura 55. Curva de ajuste logarítmico de la intensidad luminosa. Fuente: (Roal Electronics)

Además de poder regular el nivel de iluminación, en dispositivos compatibles el protocolo DALI permite ajustar la temperatura de color de la iluminación para, por ejemplo, aumentar la productividad o corregir problemas de salud como el estrés o el insomnio.

3.7. Diseño del bus DALI

La fuente de alimentación del bus debe proporcionar una tensión de salida mínima de 12 V y máxima de 20.5 V (el valor típico es 16 V), con una limitación de corriente de 250 mA en cortocircuito.

La norma EN 62386 especifica características dinámicas que debe cumplir la fuente de alimentación. Se incluyen valores de tensión y de corriente por exceso y por defecto con respecto a los límites de funcionamiento estable, así como los anchos de pulso máximo de estas desviaciones. A efectos prácticos para el presente trabajo, es de interés resaltar que la fuente de tensión no deberá producir interrupciones de suministro durante más de 450 ms para evitar fallos en el sistema.

Se utilizan niveles de tensión superiores a los usados en familias lógicas como la transistor-transistor logic (TTL) de 5 V. Esto se debe a que un mayor nivel de tensión provoca una mejor inmunidad frente al ruido causado por las interferencias del cableado eléctrico cercano.

La fuente de tensión escogida sirve de alimentación tanto al bus DALI como al microcontrolador ESP32. Presenta una tensión de salida de 12 V y una corriente máxima de 450 mA[10].

Será necesario incorporar una resistencia de salida de la fuente que limite la corriente en cortocircuito a 250 mA. Por otra parte, se requiere diseñar la interfaz de conexión del ESP32 con el bus.

El diseño del bus DALI propuesto se muestra en la siguiente figura:

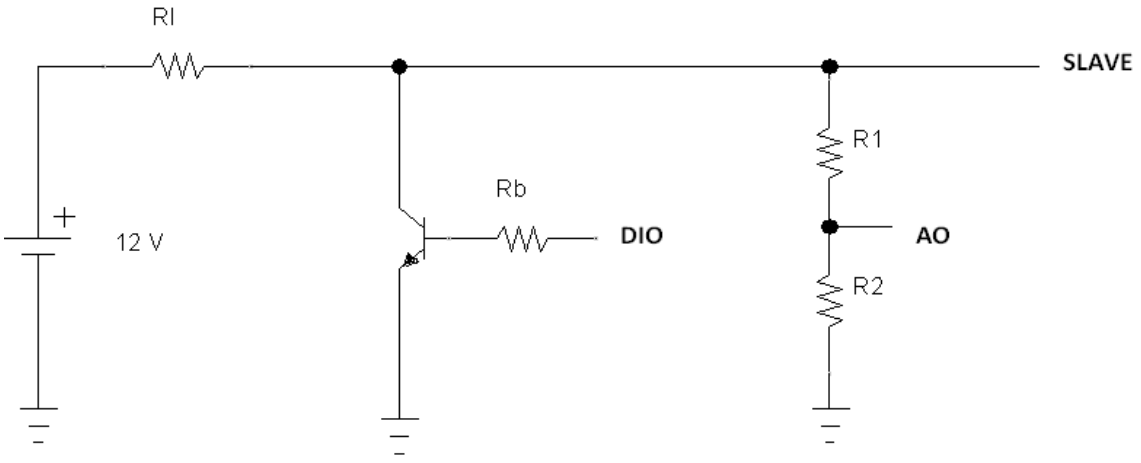


Figura 56. Esquema del diseño del bus DALI

El primer punto a considerar es la limitación de corriente del bus cuando se produce un cortocircuito. Se utiliza una resistencia dimensionada a partir de la siguiente expresión:

$$R_l = \frac{U}{I} = \frac{12\ V}{0.25\ A} = 48\ \Omega \rightarrow 47\ \Omega$$

Se escoge una resistencia de 47 Ω por ser la de valor más cercano existente en el laboratorio.

El siguiente aspecto a tratar es la resistencia de base del transistor NPN. Se utiliza un transistor BC337-40 que satura provocando un cortocircuito en el bus cuando la salida del pin digital del ESP32 está a nivel alto. Las especificaciones del transistor son las siguientes:

$$\beta_{min} = 170; U_{be} = 1.2\ V\ (I_c = 300\ mA)$$

La corriente por la base en el límite de la zona lineal es:

$$I_b = \frac{I_c}{\beta} = \frac{250\ mA}{170} = 1.471\ mA$$

La resistencia de base que garantice esa corriente de base mínima es:

$$R_b = \frac{U_{DIO} - U_{be}}{I_b} = \frac{3.3 - 1.2}{1.471 \cdot 10^{-3}} = 1427.6\ \Omega \rightarrow 1k5\ \Omega$$

[10] La elección de la fuente se ha descrito en el apartado 2.4

El valor normalizado escogido es 1500 Ω.

Por último, se dimensionan las resistencias que permiten adaptar el nivel de tensión del bus DALI al nivel aceptado por las entradas analógicas del ESP32. Se utiliza un divisor resistivo para pasar de 12 V en nivel alto a un máximo de 3.3 V en la entrada del microcontrolador. Como criterio de diseño se establece que la circulación de corriente debida únicamente a las resistencias sea de 2 mA [11].

$$R_1 + R_2 = R_t = \frac{U_{DALI}}{i} = \frac{12\ V}{2\ mA} = 6000\ \Omega$$

$$U_{R2} = U_{AO} = 3.3\ V = 12\ V \cdot \frac{R_2}{R_t} \rightarrow R_2 = 1650\ \Omega \rightarrow 1\text{k}5\ \Omega$$

Se normaliza el valor de la resistencia inferior a 1500 Ω. De esta forma, la resistencia superior se calcula como:

$$R_1 = R_t - R_2 = 6000 - 1500 = 4500\ \Omega \rightarrow 4k7\ \Omega$$

Los valores de corriente y tensión una vez normalizadas las resistencias son:

$$i = \frac{12}{4700 + 1500}\frac{V}{\Omega} = 1.94\ mA < 2mA$$

$$U_{AO} = 12 \cdot \frac{1500}{4700 + 1500} = 2.9\ V < 3.3\ V$$

La fuente de tensión escogida es de 12 V, el valor mínimo recomendable para la alimentación del bus DALI. Como cálculo previo al montaje se determina la caída de tensión que se produciría en la resistencia limitadora de corriente a fin de comprobar que la tensión en el bus está dentro de los márgenes de niveles de tensión con los que trabajan los dispositivos DALI.

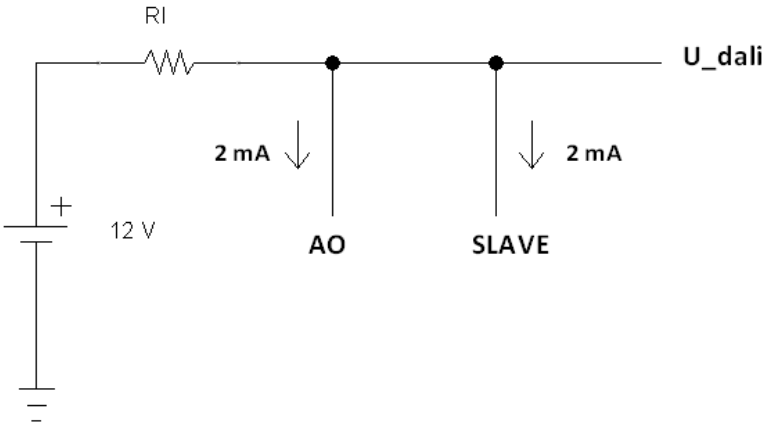


Figura 57. Consumo de corriente en el bus con maestro y esclavo en modo escucha

El límite más desfavorable a considerar es la tensión para el nivel alto en la transmisión de mensajes, que es 11.5 V. Por tanto, considerando el consumo de corriente de un esclavo DALI en escucha y del pin de recepción de señales del ESP32, la tensión efectiva en el bus sería:

$$U_{DALI} = 12\ V - 2 \cdot 2mA \cdot 47\ \Omega = 11.81\ V > 11.5\ V$$

[11] Se recuerda que este es el valor de corriente que circula por un dispositivo DALI certificado en reposo.

El valor teórico calculado está por encima del valor mínimo. No obstante, se deberá comprobar experimentalmente que el funcionamiento del circuito DALI es el adecuado.

El bus DALI diseñado queda de la siguiente forma:

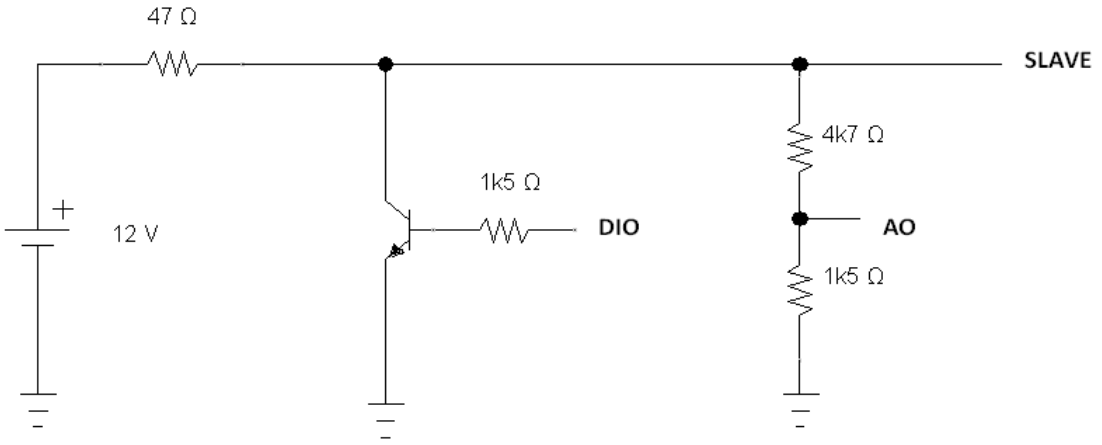


Figura 58. Interfaz DALI con valores de diseño

3.8. Montaje del circuito en Protoboard

Inicialmente se ha probado el funcionamiento del circuito diseñado en una placa de prototipos. En la figura siguiente se muestra el conexionado realizado.

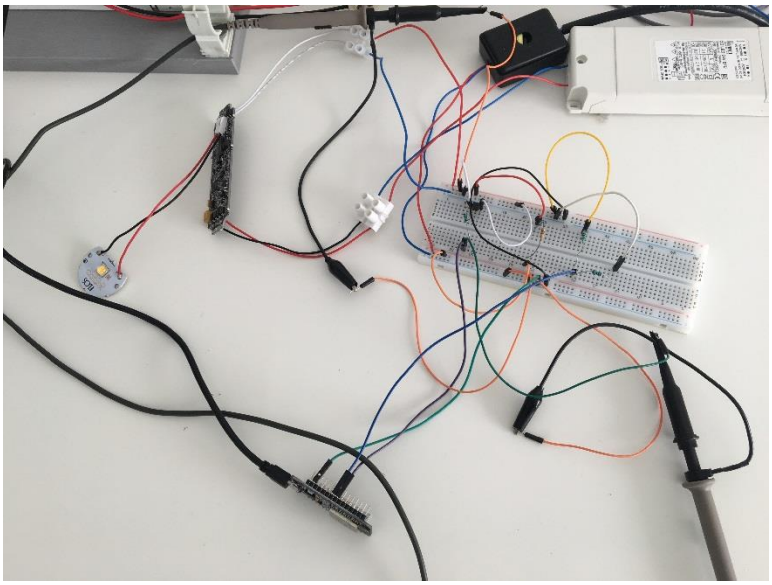


Figura 59. Montaje en protoboard del circuito DALI

En la anterior figura se muestran los distintos componentes que forman parte de la experimentación sobre la tecnología DALI.

El dispositivo esclavo DALI está alimentado con una fuente de tensión de 24 V y su dispositivo de salida es un led de 24 V, con el que se realizan todas las pruebas de control y regulación de iluminación a través del protocolo DALI.

Para las primeras pruebas, la comunicación con el ESP32 se realiza a través del puerto serie con un PC. En el montaje final, la comunicación con el usuario se realizará a través de la app Android y no a partir de un ordenador.

Por último, en la anterior figura se pueden apreciar las sondas del osciloscopio, que permitirá obtener capturas del comportamiento del bus durante la comunicación entre el ESP32 funcionando como maestro y el esclavo.

3.9. Ensayos de comunicación DALI

En primer lugar, se comprueba el ancho de pulso en las tramas enviadas por el ESP32 y se contrasta con las tolerancias admisibles según la norma EN 62386.

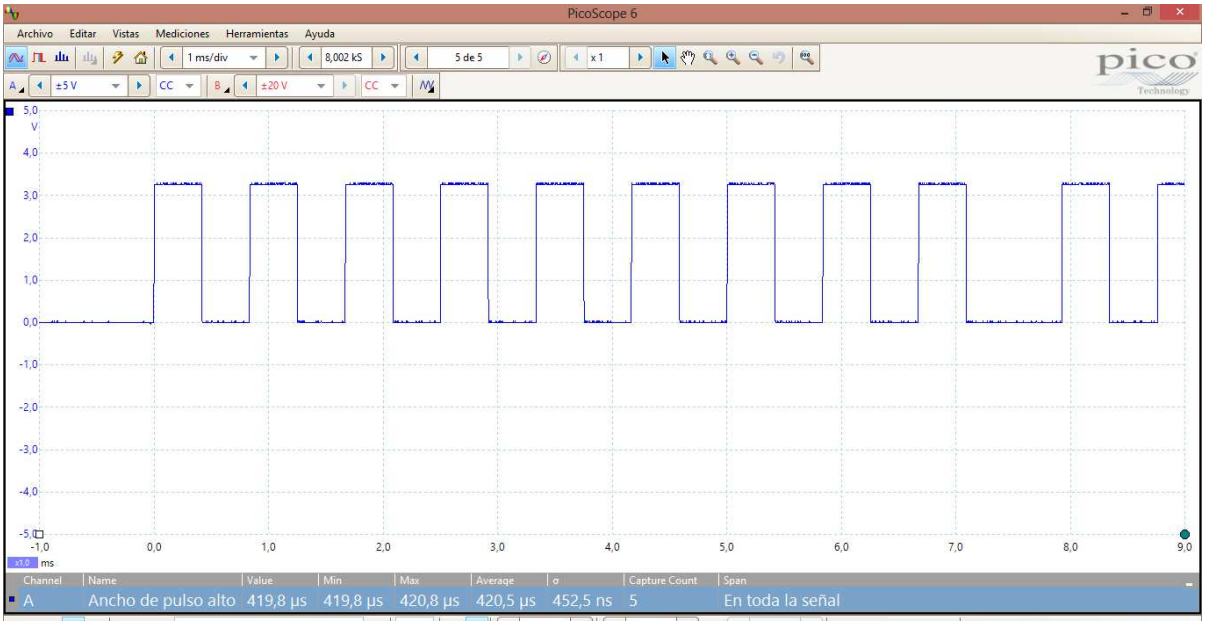


Figura 60. Trama forward visualizada con el osciloscopio

Después de enviar 5 tramas forward con diferentes órdenes, el ancho de pulso medio es 420.5 µs, comprobándose que está dentro del rango permitido[12].

Utilizando dos sondas, una en la salida del pin digital que controla la transmisión desde el ESP32 y otro en bornes del bus DALI, se puede observar la evolución del bus ante distintos comandos enviados.

Las siguientes figuras muestran las tramas transmitidas por el bus para diferentes ensayos realizados:

En el primer caso se corresponde con el **encendido de la luminaria mediante un control de difusión** con el envío del comando de encendido con máximo nivel de flujo luminoso. La trama enviada es: 1111 1111 0000 0101.

[12] Para un único maestro: 366.7 µs – 466.7 µs; para configuración con varios maestros: 400 µs – 433.3 µs.

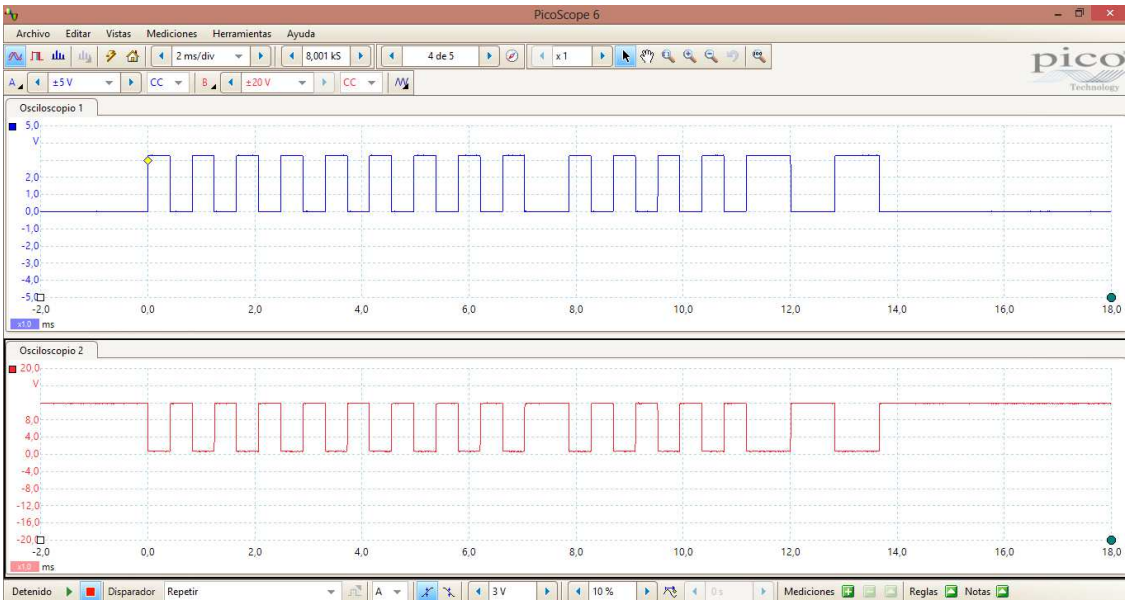


Figura 61.Trama de encendido del esclavo por comando de alimentación

El segundo ensayo realizado consiste en la **consulta del estado** mediante un comando de difusión. El maestro envía la trama 1111 1111 1001 0000, y el esclavo contesta con la siguiente trama backward: 0000 0100.

Cada bit de la trama backward tiene el siguiente significado:

Note STATUS INFORMATION: 8-bit data indicating the status of a slave. The meanings of the bits are as follows:

bit 0	Status of control gear :<0>=OK	bit 4	Fade running:<0>=fade is ready, <1>=fade is running
bit 1	Lamp failure :<0>=OK	bit 5	Query RESET STATE :<0>=No
bit 2	Lamp arc power on :<0>=OFF	bit 6	Query Missing short address :<0>=No
bit 3	Query Limit Error :<0>=No	bit 7	Query POWER FAILURE :<0>=No

Tabla 3. Significado de los bits de la trama backward de respuesta a una consulta de estado

Considerando que el bit 7 se corresponde con el más significativo y el bit 0 con el menos, la respuesta del esclavo indica que:

- No hay fallo de alimentación (0).
- No falta la “short address” (0).
- El esclavo no está en modo reinicio (0).
- El modo de desvanecimiento no está activo (0).
- No hay error de límite de consulta (0).
- La luminaria sí está encendida (1).
- No hay fallo en la luminaria (0).
- El esclavo funciona correctamente (0).

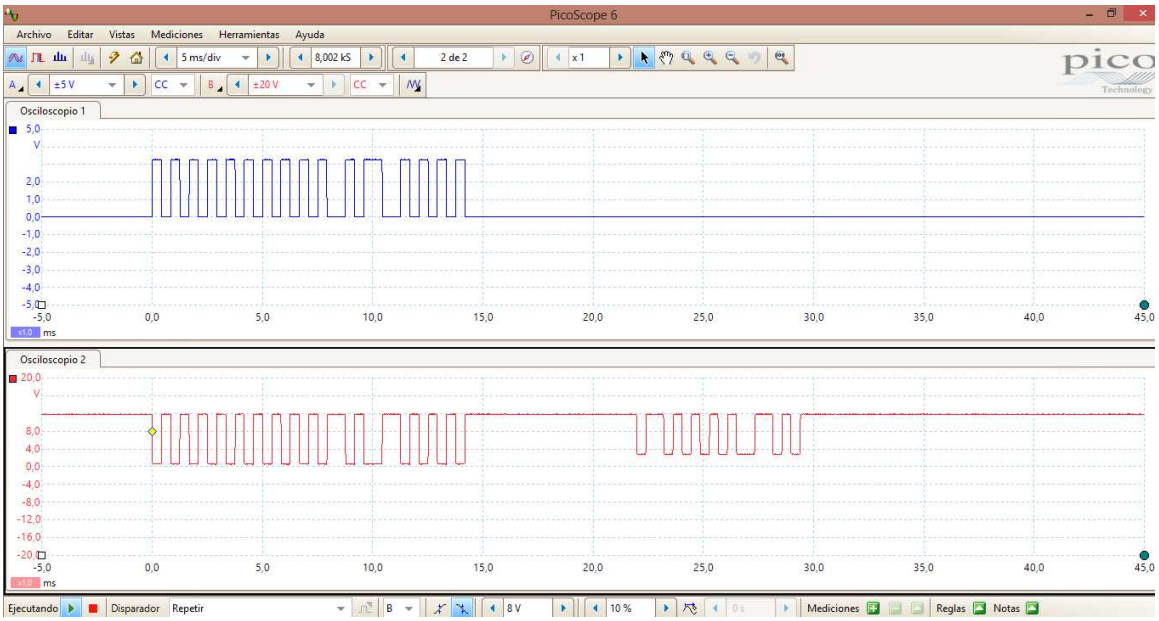


Figura 62. Trama forward y respuesta del esclavo a una petición de estado

El último ensayo realizado consiste en **la consulta del nivel actual de regulación del flujo** luminoso en el esclavo. La trama forward enviada es 1111 1111 1010 0000, y la respuesta del esclavo es 1010 1010 (170 sobre un máximo de 254).

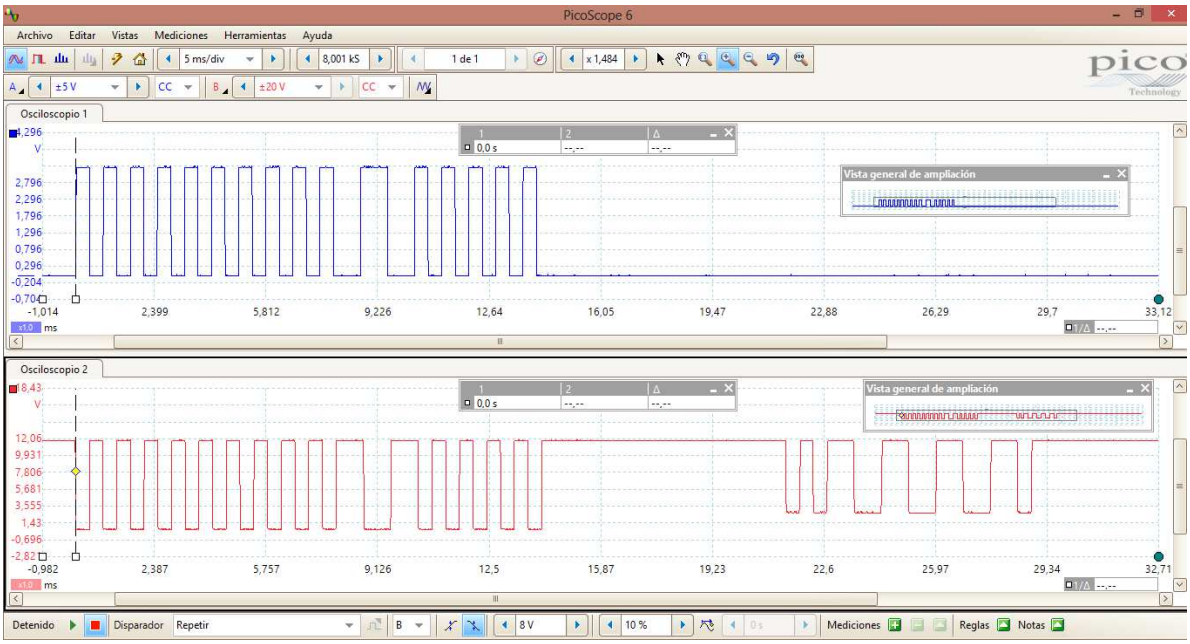


Figura 63. Trama forward y respuesta del esclavo a la petición de nivel actual de regulación

En los ensayos realizados se ha probado la comunicación a través de tramas de difusión por control directo de la alimentación y por comandos, así como a través de comandos de dirección individual. Se concluye que el comportamiento del bus DALI es el adecuado.

4. DISEÑO DEL SOFTWARE

En este capítulo se presenta el desarrollo de la programación tanto del microcontrolador como de la interfaz de comunicación con el usuario. En este punto del trabajo ya se ha determinado la placa de desarrollo a utilizar (ESP32). Sin embargo, falta plantear diferentes altarnativas que permitan el control del microcontrolador de forma remota.

Como especificación previa en este trabajo se estableció que el usuario debería controlar el banco de ensayos a través de una Tablet con sistema operativo Android, siendo el desarrollo de la aplicación uno de los puntos principales del trabajo. Por esta razón, no se analizarán opciones en las que se use un ordenador como dispositivo de control, como podría ser la configuración de un servidor web.

4.1. Análisis de alternativas de comunicación usuario-microcontrolador

El primer aspecto a considerar es ver de qué posibilidades de comunicación inalámbrica dispone el ESP32. Las dos alternativas posibles son la comunicación por Wifi y la comunicación por bluetooth.

El **Wifi** (“Wireless Fidelity”) es una tecnología de comunicación inalámbrica basada en el estándar IEEE 802.11. Se trata de una red de área local inalámbrica WLAN (“Wireless Local Area Network”) que permite la conexión de varios dispositivos simultáneamente sin cable con alcance moderado (hasta 200 m en interior). Existen dos bandas principales de funcionamiento 2.4 GHz y 5 GHz, con configuraciones de canales y anchos de banda diferentes (Muriel Stefanuto, Melquiades dos Santos, & Taveira Torres, 2016). Es apropiado para la transmisión de grandes cantidades de datos, teniendo el ESP32 una tasa de transferencia por Wifi de hasta 150 Mbps (Mega bits por segundo).

Figura 64. Logo Wifi. Fuente: Wikipedia.org

Por otra parte, el **bluetooth** es una plataforma de comunicación que opera a la frecuencia de 2.4 GHz y se basa en el estándar IEEE 802.15 (Ela Nsogo Nsa & Parra Rodríguez, 2018). Configura una red ad hoc inalámbrica, esto es una red descentralizada en la que dos dispositivos se pueden comunicar directamente sin depender de infraestructura adicional como pueden ser los routers en la comunicación por Wifi. En comparación con la tecnología Wifi presenta un menor alcance (hasta 20 m en la versión 4.2) y una menor tasa de transmisión de datos (la velocidad máxima en el ESP32 es 4 Mbps).

Figura 65. Logo bluetooth. Fuente: Wikipedia.org

A priori, cualquiera de las dos tecnologías de comunicación inalámbrica sería perfectamente válida para los objetivos de este trabajo.

Por una parte, no se requiere una velocidad de transmisión excesivamente elevada. Los datos enviados por el usuario son códigos discretos que el microcontrolador interpreta para realizar las acciones requeridas, mientras que los datos recibidos de forma periódica son un conjunto de muestras correspondientes a dos canales de medida. En ningún caso se alcanzarán límites cercanos a 1 Mbps.

En cuanto al alcance, la operación del banco de ensayos se realiza desde la misma habitación en la que se encuentra el microcontrolador, por lo que no se alcanzará el límite de 10-20 m a partir del que el bluetooth deja de funcionar de forma adecuada.

Se opta por escoger la tecnología de bluetooth para establecer un enlace entre el ESP32 y las Tablets disponibles en el laboratorio. La razón principal es que posibilita una comunicación entre dispositivos más directa, sin necesidad de tener que configurar un servidor MQTT, además de no depender del funcionamiento del router de enlace y de la disponibilidad de la red.

Dentro de las posibilidades del bluetooth, el ESP 32 dispone de **bluetooth clásico** en la versión 4.2 y de **bluetooth de baja energía** (BLE). El BLE es una variación del bluetooth clásico que permite reducir el consumo de potencia hasta en 100 veces. Esto se consigue manteniendo el dispositivo en suspensión excepto cuando se inicia una nueva conexión. Está pensado para dispositivos que intercambien pequeñas cantidades de datos de forma ocasional y en los que la disponibilidad de energía es muy limitada.

En el caso concreto de este trabajo, el microcontrolador está conectado a una fuente de alimentación, por lo que el consumo de energía del módulo bluetooth no es un aspecto crítico, teniendo en cuenta además que el consumo máximo del módulo bluetooth es de 100 mW y que permanecerá apagado cuando el banco de ensayos no se esté utilizando. Por tanto, no existen razones aparentes que justifiquen el uso del BLE frente al bluetooth clásico.

4.2. Programación del ESP32

La programación del microcontrolador se realiza en el entorno de desarrollo integrado Arduino IDE. Se trata del entorno oficial de programación de microcontroladores de la marca Arduino, aunque se puede utilizar con placas de desarrollo de otros proveedores, como es el caso del ESP32. Este software está basado en el entorno de Processing (lenguaje de programación basado en Java) y la plataforma Wiring, que dispone de un compilador C/C++ y permite la

programación del código en estos lenguajes (Falcón Oubiña & Armesto Quiroga, 2017). Como resultado, el usuario dispone de partida de dos funciones ya programadas:

- Setup: Ejecutado una sola vez al inicio del programa y que se utiliza para definir la configuración inicial de la placa.
- Loop: Bucle de repetición continuada, en el que se puede programar el núcleo de ejecución del programa.

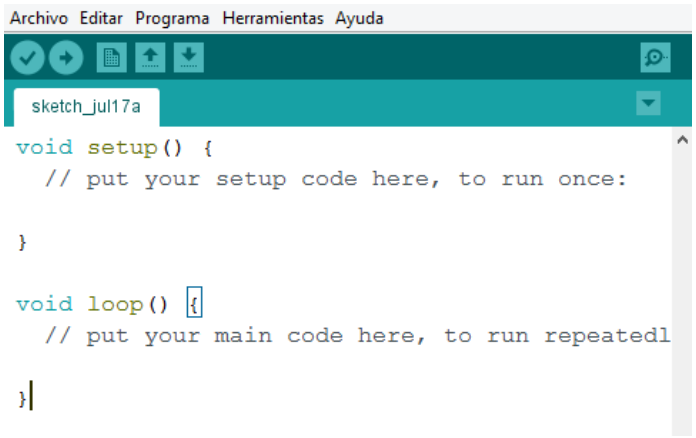


Figura 66. Arduino IDE

La placa de desarrollo debe realizar una serie de funciones básicas que se detallan en los siguientes puntos:

- **Gestión de la comunicación por bluetooth** con la Tablet. Se debe comprobar de forma continua si se ha recibido algún dato por el enlace bluetooth, y de ser así, interpretar la información recibida y realizar la acción requerida. De forma paralela, se realizará el envío de forma ininterrumpida de los datos de las medidas realizadas.
- **Encendido y apagado de las diferentes lámparas**. Esto se realiza a través de las salidas digitales que controlan los relés.
- **Regulación del flujo luminoso**. Cuando se desee modificar el nivel de iluminancia, la app enviará un conjunto de datos que el ESP32 interpretará y traducirá a un duty cycle para un control PWM específico.
- **Lectura de medidas de tensión y corriente**. El microcontrolador recibirá un conjunto de lecturas a través de dos puertos de entrada analógicos, y tras la conversión a digital enviará un código a la app con la información de la medida.
- **Control del bus DALI**. A través de los puertos de transmisión (pin de salida digital) y recepción (pin de entrada analógico), el ESP32 realiza el envío de trenes de pulsos por el bus, según la orden sea de encendido, apagado o control del flujo luminoso y recibe información del estado del propio bus y del esclavo conectado.

En el siguiente diagrama de flujo se puede apreciar con más claridad la relación entre las diferentes acciones del programa:

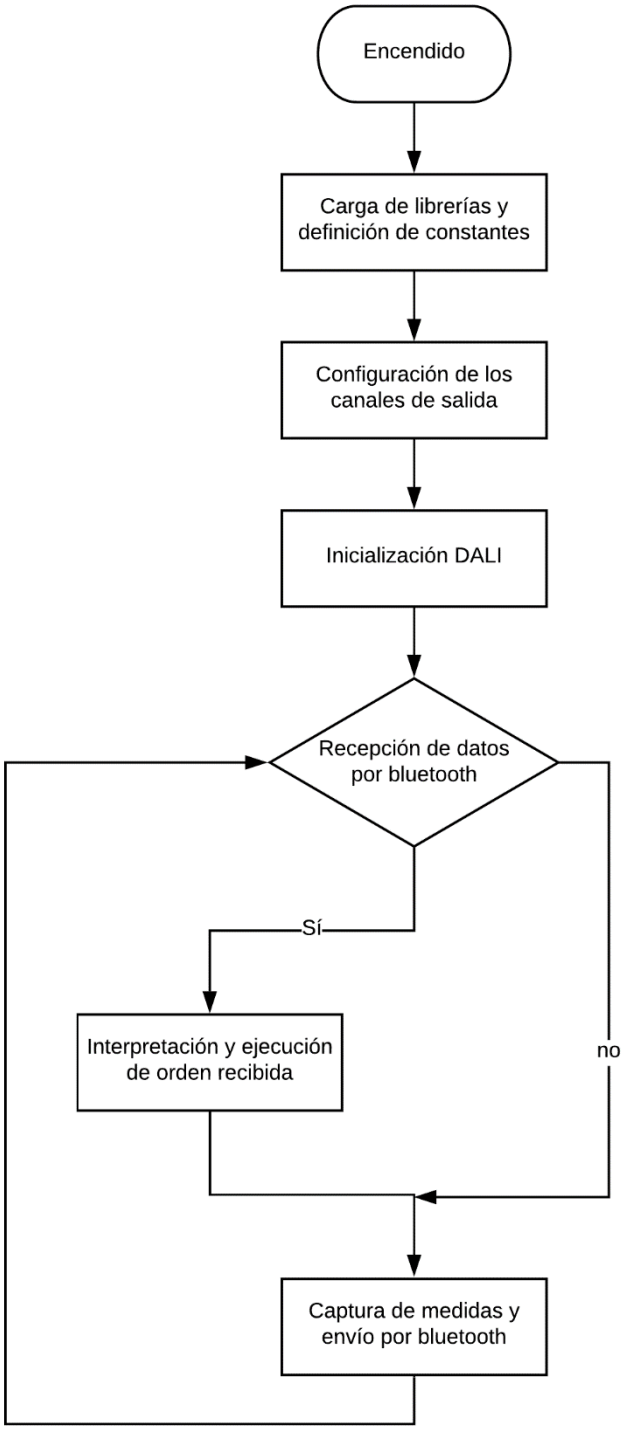


Figura 67. Diagrama de flujo en el ESP32

En los siguientes apartados se incluyen los fragmentos de código asociados a cada tarea, así como una breve explicación de su funcionamiento.

4.2.1. Inicialización del programa

La primera tarea a realizar es la incorporación de las librerías y definición de las constantes utilizadas en el programa.

Las librerías empleadas son la librería interna Dali.h (cuya funcionalidad se abordará en el apartado 4.2.6) y la librería externa <BluetoothSerial.h>, que permite controlar el hardware en la comunicación por Bluetooth.

En cuanto a la definición de constantes, es interesante destacar la configuración de PWM y de captura de muestras elegida.

- PWM: Se configura con una frecuencia de 5000 Hz, que se ha comprobado que funciona adecuadamente tanto con el balasto electrónico como con el dimmer led, y una resolución de 12 bits.
- Muestreo de medidas: La frecuencia de muestreo es 5000 Hz (100 muestras por cada periodo de señal eléctrica a 50 Hz). Se utilizan vectores de variable tipo short para el registro del tiempo y del entero medido.

El siguiente fragmento de código muestra la incorporación de las librerías y la definición de las variables principales.

```
#include "Dali.h"
#include <BluetoothSerial.h>

//Conmutación relés
#define RELAY1 22
#define RELAY2 23
#define LED 19

//Control PWM Dimmer Led
const int ledPin = 18;
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 12;   //Resoluciones de 1 a 16 bits; Se escoge
12 bits
//Control PWM Fluorescente Balasto Electrónico
const int bePin = 21;

//Variables usadas en la regulación del flujo luminoso
int dutyCycle = 0;
int a, b, c;
int reg;
float rbe;
float rled;

//Constantes Dali
const int DALI_TX = 32;
const int DALI_RX_A = 33;
#define BROADCAST_DP 0b11111110 //Broadcast Direct Power Arc Control
#define BROADCAST_C 0b11111111  //Broadcast Command
#define ON_DP 0b11111110        //Nivel máximo 254
#define OFF_DP 0b00000000       //Nivel mínimo 0
#define ON_C 0b00000101         //Nivel máximo por comandos
#define OFF_C 0b00000000        //Off
#define MIN_LEVEL 0b00000110    //Nivel mínimo por comandos
#define QUERY_STATUS 0b10010000 //Consulta de estado
#define QUERY_LEVEL 0b10100000  //Consulta de nivel actual
#define RESET 0b00100000        //Reset del esclavo

BluetoothSerial ESP_BT; //Objecto Bluetooth
int incoming; //Dato recibido por bluetooth
```

```
//Configuración muestreo de medidas
int sf =5000;   //Frecuencia de muestreo. Por defecto son 100 muestras
por periodo
int br = 12;    //Bit resolution
int deltat = 200;   //200 us, tiempo entre muestras
int samples = 100;  //100 muestras por periodo
int opmode = 0; //código de operación
unsigned short ch0[100], ch1[100];  //vector de valores medidos (100
muestras por periodo)
unsigned short anaread[100];  //vector de tiempos
```

Dentro ya de la función setup, se realizan una serie de tareas:

- Configuración de los canales PWM con las constantes definidas previamente.
- Inicialización de la comunicación serie[13] y bluetooth.
- Configuración de los pines de salida.
- Inicialización del bus DALI (esto se abordará con más detalle en el apartado 4.2.6).

La función setup se incluye en el siguiente fragmento de código.

```
void setup() {

  // configuración canal PWM
  ledcSetup(ledChannel, freq, resolution);

  // vinculación canal PWM a pines de salida
  ledcAttachPin(ledPin, ledChannel);
  ledcAttachPin(bePin, ledChannel);

  //Inicialización serial y bluetooth
  Serial.begin(115200);
  ESP_BT.begin("IMIE_006");

  pinMode(RELAY1, OUTPUT);
  pinMode(RELAY2, OUTPUT);
  pinMode(LED,OUTPUT);
  digitalWrite(RELAY1, LOW);
  digitalWrite(RELAY2, LOW);
  digitalWrite(LED, LOW);

  analogSetAttenuation(ADC_11db); //Permite ajustar la atenuación y el
fondo de escala de las medidas

  delay(500);   //Margen de tiempo para la estabilización de la
alimentación

  //Inicialización DALI
  dali.setupTransmit(DALI_TX);
  dali.setupAnalogReceive(DALI_RX_A);
  dali.busTest();
  dali.msgMode = true;
  dali.initialisation();
}
```

[13] Para el control remoto, la comunicación serie no es necesaria. Sin embargo, es de gran utilidad para la depuración de errores durante las fases previas a la conexión en el montaje final. No obstante, es importante destacar que al estar el circuito de medida, y por tanto el ESP32, sin aislar de la red eléctrica, no se debe establecer una comunicación serie con un ordenador cuando se realicen medidas sobre el banco de ensayos.

4.2.2. Recepción de órdenes por Bluetooth

Una vez que se entra al bucle del loop, el primer paso consiste en comprobar si hay alguna emisión pendiente por el enlace bluetooth. Si es así, se lee el dato recibido y se interpreta.

El envío de órdenes por parte de la aplicación se realiza a partir de caracteres codificados en ASCII. Existen 8 órdenes diferentes que se codifican con los caracteres del 1 al 8, que corresponden con los enteros del 49 al 56.

Figura 68. Tabla de caracteres ASCII. Fuente: https://informatica653.wordpress.com/

A través de comprobaciones condicionales, se interpreta el código enviado y se realizan las diferentes acciones requeridas.

El fragmento de código mostrado a continuación incluye la gestión de las comunicaciones por bluetooth.

```
 if (ESP_BT.available()) //Comprueba si se ha recibido algo por el
enlace bluetooth
  {
    incoming = ESP_BT.read(); //Lectura del dato recibido

    if (incoming == 49) //1 en ASCII    Apagado general
        {
        digitalWrite(LED, LOW);
        digitalWrite(RELAY1, LOW);
        digitalWrite(RELAY2, LOW);
        opmode = 1;
        }

        ......................

else {
    capture();
    ESP_BT.println("sampling finished");
    report();
  }
  delay(500);
```

Entre cada comprobación y envío de señales por Bluetooth, se incorpora un estado de espera de medio segundo que permite reducir la potencia consumida y el calentamiento del microcontrolador. Esto implica además que el refresco de la visualización de datos en la

aplicación Android se produce a intervalos regulares de tiempo y no de forma totalmente continua, reduciendo posibles problemas de ralentización de la app por el elevado número de datos a mostrar.

4.2.3. Encendido y apagado a través de los relés

Esto se realiza enviando por los pines digitales correspondientes una señal de nivel alto o bajo. En el caso de que el usuario seleccione el modo de operación DALI, en primera instancia se procede al apagado del resto de sistemas del banco de ensayos.

En el siguiente fragmento de código se presentan los condicionales para cada orden diferente y las acciones realizadas.

```
 if (incoming == 49) //1 en ASCII    Apagado general
        {
        digitalWrite(LED, LOW);
        digitalWrite(RELAY1, LOW);
        digitalWrite(RELAY2, LOW);
        opmode = 1;
        }

    if (incoming == 50) //2 en ASCII    Fluorecente convencional
        {
        digitalWrite(LED, LOW);
        digitalWrite(RELAY2, LOW);
        digitalWrite(RELAY1, HIGH);
        opmode = 2;
        }

    if (incoming == 51) //3 en ASCII    Fluorescente Balasto
Electrónico
        {
        digitalWrite(LED, LOW);
        digitalWrite(RELAY1, LOW);
        digitalWrite(RELAY2, HIGH);
        opmode = 3;
        }

    if (incoming == 52) //4 en ASCII    Led
        {
        digitalWrite(RELAY1, LOW);
        digitalWrite(RELAY2, LOW);
        digitalWrite(LED, HIGH);
        opmode = 4;
        }
```

4.2.4. Regulación del flujo luminoso

Si se recibe el código asociado a la regulación del flujo por el enlace bluetooth, se procede a decodificar la consigna de flujo enviada desde la aplicación Android.

Desde la app, el entero que establece el nivel de atenuación del flujo se transforma en un string y se envía por el enlace bluetooth. El ESP32 recibe ese string en formato ASCII y lo transforma a un entero con el que se puede operar. Por último, en función de si está operando el tubo fluorescente o el tubo led (que se comprueba a través de la variable “opmode”), se ajusta el valor de la atenuación en base 100 demandado por el usuario, según las curvas de

regulación obtenidas en el apartado 2.6, al entero que establece el ciclo de trabajo de la regulación PWM.

En el siguiente fragmento de código se muestra la gestión de la regulación del flujo luminoso.

```
if (incoming == 56) //8 en ASCII    Regulación Flujo
    {
      //Decodifica el valor de regulación transmitido con caracteres
ASCI
      // y lo transforma en un entero entre 0-100
      incoming = ESP_BT.read();
      a = incoming-48;
      if (ESP_BT.available()){
      incoming = ESP_BT.read();
      b = incoming-48;
      a = a*10;
      }
      if (ESP_BT.available()){
      incoming = ESP_BT.read();
      c = incoming -48;
      a = a*10;
      b = b*10;
      }
      reg = a + b + c;
      if (opmode==3)  //Balasto Electrónico
      {
        rbe = reg/5;  //20 muestras, se divide entre 5
        rbe = 0.1848*pow(rbe,3)-3.9614*pow(rbe,2)+38.36*rbe+120.7;
        dutyCycle = rbe;
        ledcWrite(ledChannel, dutyCycle);
        }

      if (opmode==4)  //LED
      {
        rled = reg/8.3333;  //12 muestras
        rled = 259.62*rled+416.67;
        dutyCycle = rled;
        ledcWrite(ledChannel, dutyCycle);
      }
      if (opmode==5 || opmode==6)  //DALI
      {
        dutyCycle = reg*255/100;
        dutyCycle = 255-dutyCycle;    //En DALI 254 es el nivel
máximo, al contrario que en PWM
      if (dutyCycle >254) dutyCycle = 254;  //Se ha comprobado que 255
no produce respuesta en slave DALI
        dali.transmit(BROADCAST_DP, dutyCycle);
      }
    }
```

El caso de la regulación del flujo DALI es diferente al de los tubos del banco de ensayos. No se programa el ciclo de trabajo como en la regulación por PWM, sino que el valor de atenuación marcado por el usuario se traduce en el nivel de iluminancia que se debe transmitir al esclavo a través del bus DALI. Este nivel está codificado con 8 bits, y se envía a través de un comando de difusión (broadcast) de control directo de potencia.

Sólo se ejecuta el conjunto de órdenes asociadas a la regulación del bus DALI si la variable “opmode” es igual a 5 ó 6. Esto se produce cuando el usuario ha seleccionado desde la aplicación Android el modo de operación DALI y la última operación previa no ha sido la de

apagado del dispositivo DALI. Esto implica que, si el usuario apaga el led del esclavo DALI, para variar el flujo luminoso primero hay que solicitar su encendido de nuevo.

Como resultado, hay dos formas de regular el flujo del esclavo DALI:

- nada más entrar al modo DALI (sin el encendido previo)
- o cuando se ha solicitado el encendido a través del botón correspondiente[14].

Con esta forma de control se busca poner énfasis en que en el accionamiento por bus DALI no se envía sólo la orden de encendido, sino que se acompaña siempre de información adicional acerca del nivel de iluminación requerido. No es necesario tener una lámpara ya encendida para regular su flujo, como si sucede en la regulación 0-1/10 V.

4.2.5. Envío de datos de medida por bluetooth

Una vez que se ha atendido a la petición pendiente, a través de la lectura e interpretación del dato recibido por el enlace bluetooth, se realiza el muestreo de las señales de medida y el envío de los datos. Esto se muestra en el siguiente fragmento de código.

```
void loop() {

 if (ESP_BT.available()) //Comprueba si se ha recibido algo por el
enlace bluetooth
  {
    incoming = ESP_BT.read(); //Lectura del dato recibido

……..

 else {
    capture();
    ESP_BT.println("sampling finished");
    report();
  }
```

Estas acciones se ejecutan a partir de dos funciones: “capture” y “report”.

La función capture se incluye a continuación:

```
void capture(){

        unsigned int nowus, nextus;
        unsigned short ana40, ana41;

        for(int i =0; i<samples;i++){
          //Solo entra en la primera muestra
          //Espera a que se inicie un nuevo ciclo
          if(!i) {
            ana40 =
analogRead(36)+analogRead(36)+analogRead(36)+analogRead(36);
            // Se discretiza a 8192 (Punto medio==0 de tensión) --> 4
medidas*4096/2 (máximo de 3.3 V)/ 2
            // si estamos en el semiciclo positivo, esperamos a pasar
al negativo
            while(ana40 > 8192) ana40 =
analogRead(36)+analogRead(36)+analogRead(36)+analogRead(36);
            // si ya estamos en el negativo, esperamos a pasar al
positivo
```

[14] Para una mejor comprensión se recomienda consultar el apartado 4.3.2, en donde se explica la interfaz de la app.

```
            while(ana40 < 8192) ana40 =
analogRead(36)+analogRead(36)+analogRead(36)+analogRead(36);

            anaread[0]=0;   //anaread contiene el vector de tiempos
iniciado en cero
            nowus = micros();
            nextus = nowus + deltat;
          }
          else anaread[i] = micros()-nowus;
          //Toma 4 muestras de cada vez y realiza la media

          ana40 = analogRead(36);
          ana41 = analogRead(39);

          ana41 += analogRead(39);
          ana40 += analogRead(36);

          ana40 += analogRead(36);
          ana41 += analogRead(39);

          ana41 += analogRead(39);
          ana40 += analogRead(36);

          ana40 = (ana40+2)>>2;            //Equivale a dividir por 4
          ana41 = (ana41+2)>>2;

          ch0[i] = ana40;
          ch1[i] = ana41;

          while(nextus > micros()){}

          nextus +=deltat;
        }
}
```

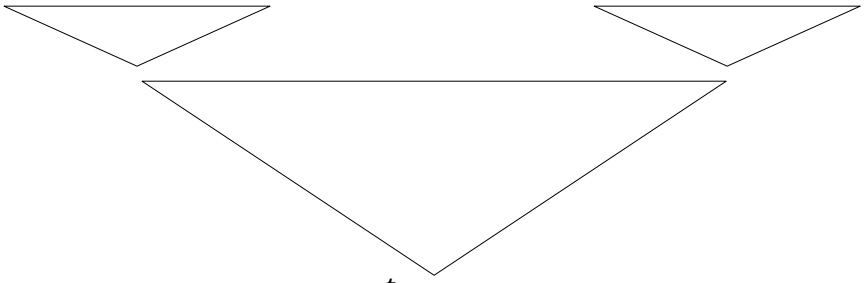
El primer paso consiste en detectar el instante en el que la tensión leída pasa por el punto medio del rango de entrada (1.65 V) en el canal de entrada correspondiente a la medida de tensión del banco de ensayos. Este punto medio se corresponde con el valor teórico del cero de tensión y de corriente.

Una vez detectado el cambio entre el semiciclo positivo o negativo, se establece el origen de tiempos y se procede a la captura del conjunto de muestras que definen un periodo. Esto se realiza de esta forma para mejorar la visualización de la gráfica, ya que los datos siempre parten desde el instante de tensión nula.

La gestión del instante de captura se realiza a partir de las variables “nowus” y “nextus”. En nowus se registra el tiempo en microsegundos en el que se inician las medidas y que marca el origen de tiempos. Con nextus se establece el tiempo de espera entre muestras consecutivas.

Para reducir la variación temporal del instante en que se muestrea un canal u otro, se capturan medidas cuadriplicadas y se realiza un promedio posterior. Se aprovecha la elevada tasa de muestreo que puede desarrollar el ESP32, al capturar los datos de los 2 canales de forma alterna, y no primero un canal y después el otro. Así, el tiempo promedio en el que se ha realizado el muestreo y retención de las medidas es el mismo entre ambos canales, eliminando el desfase temporal que se produciría con una captura de señal única y con un orden fijo entre los canales. El siguiente esquema resume el fundamento lógico de este procedimiento:

$$|canal\ 0\ \&\ canal\ 1| + |canal\ 1\ \&\ canal\ 0| + |canal\ 0\ \&\ canal\ 1| + |canal\ 1\ \&\ canal\ 0|$$



$$t_{promedio}$$

El resultado de la función capture es un vector con el registro de tiempos ("anaread") y otro vector con las medidas capturadas (enteros con resolución 12 bits). Es la aplicación Android la que se encarga de interpretar estos enteros y transformarlos en los valores reales medidos del banco de ensayos.

La función report se muestra a continuación:

```
void report(){
  unsigned int nch0, nch1;
  for(int i =0; i<samples;i++){
    nch0=ch0[i];
    nch1=ch1[i];
    ESP_BT.print(";");ESP_BT.print(anaread[i]);ESP_BT.print(";");
    ESP_BT.print(nch0);ESP_BT.print(";");
    ESP_BT.print(nch1);ESP_BT.print(";");
    ESP_BT.println(i);
  }
}
```

El envío de los datos capturados se realiza a través de un bucle en el que se recorren los vectores de tiempo, medida del canal 1 (asociado a la tensión) y medida del canal 2 (asociado a la corriente). En cada entrada al bucle, se trasladan los datos de una posición determinada en los vectores al enlace bluetooth.

La aplicación Android se encarga de interpretar esta información enviada, en base a la estructura de separadores ";" entre cada categoría de dato, y los saltos de línea entre cada posición en los vectores.

4.2.6. Control del bus DALI

El manejo del bus DALI es la parte más compleja del código programado en el microcontrolador. Se requiere la definición de una serie de funciones que gestionan el envío y recepción de señales atendiendo al sistema de codificación Manchester a la velocidad de 1200 Bps, tal como se ha descrito en el apartado 3.4.

Para el desarrollo de esta parte del trabajo, se ha partido de una librería que implementa las funciones básicas para el control de luminarias DALI a partir del microcontrolador Arduino Nano (create.arduino.cc, 2017). Se han realizado algunas modificaciones en estas funciones, para adaptar la librería al uso con ESP32 y a los requerimientos de este proyecto.

En este apartado, a diferencia de los anteriores no se incluye la totalidad del código de programación, sólo los aspectos más destacables para la comprensión de la lógica ejecutada.

En el anexo (apartado 8) se incluye todo el código de la librería Dali, tanto el archivo de cabecera (Dali.h) como el de código fuente (Dali.cpp).

El primer paso del control del DALI es la comprobación del nivel medio de tensión en el bus (que se utiliza para la discretización de nivel alto y bajo) y la inicialización de los dispositivos conectados al bus. Estas tareas se realizan con llamadas a las funciones correspondientes en el setup, es decir, al conectar el microcontrolador a la alimentación.

La función "busTest" obtiene el valor de tensión en el bus en base a las respuestas obtenidas a la solicitud de estado a los esclavos conectados.

La función "initialisation" transmite comandos de inicialización de los balastos y generación aleatoria de dirección completa o "long address". Una vez generadas las direcciones aleatorias, se realiza una búsqueda en todo el espectro posible de direcciones para obtener los emparejamientos disponibles de cada esclavo con su dirección.

Dentro del bucle principal se encuentran las distintas opciones que afectan al bus DALI. Estas se resumen en los siguientes puntos, en función del número recibido por el enlace bluetooth:

- 5: entrada al modo de operación del bus DALI y apagado del resto de luminarias si están encendidas (tubos fluorescentes y tubo led).
- 6: encendido del dispositivo DALI por comando de difusión. El nivel de flujo luminoso es máximo en este caso.
- 7: apagado del led conectado al esclavo DALI por comando de difusión.
- 8: variación del flujo luminoso.

El siguiente fragmento de código muestra las acciones DALI dentro del bucle principal.

```
  if (incoming == 53) //5 en ASCII    DALI
        {
        digitalWrite(LED, LOW);
        digitalWrite(RELAY1, LOW);
        digitalWrite(RELAY2, LOW);
        opmode = 5;
        }

    if (incoming == 54) //6 en ASCII    DALI ON
    {
      dali.transmit(BROADCAST_C, ON_C);
      opmode = 6;
    }

    if (incoming == 55) //7 en ASCII    DALI OFF
    {
      dali.transmit(BROADCAST_C, OFF_C);
      opmode = 7;
    }
```

A continuación se incluyen las funciones que se utilizan para el envío de datos según el estándar del bus DALI. Este fragmento de código ya no está en el archivo principal (.ino), sino dentro de la librería de DALI.

```
void Dali::setupTransmit(uint8_t pin)
{
  TxPin = pin; // Se establece el pin como salida
  pinMode(TxPin, OUTPUT);
  digitalWrite(TxPin, LOW);
```

```
  delay1=416;
  delay2=416;
  period = delay1 + delay2;    //Se usa en la recepción (Dali.receive)
  }

void Dali::transmit(uint8_t cmd1, uint8_t cmd2) // Transmisión de
comandos (address byte, command byte)
{
  sendBit(1);
  sendByte(cmd1);
  sendByte(cmd2);
  digitalWrite(TxPin, LOW);
}

void Dali::sendByte(uint8_t b)
{
  for (int i = 7; i >= 0; i--)
  {
    sendBit((b >> i) & 1);
  }
}

void Dali::sendBit(int b)
{
 if (b) {
    sendOne();
  }
  else {
    sendZero();
  }
}

void Dali::sendZero(void)
{
  digitalWrite(TxPin, LOW);
  delayMicroseconds(delay2);
  digitalWrite(TxPin, HIGH);
  delayMicroseconds(delay1);

}

void Dali::sendOne(void)
{
  digitalWrite(TxPin, HIGH);
  delayMicroseconds(delay2);
  digitalWrite(TxPin, LOW);
  delayMicroseconds(delay1);
}
```

Para realizar la codificación Manchester a una velocidad de 1200 Bps, se siguen los siguientes pasos:

- Se establece el tiempo de espera entre un estado y su opuesto (416 µs).
- Se divide la trama de datos a enviar en dos bytes (direccionamiento y datos).
- Se recorre cada uno de los bytes identificando los bits que contiene.
- Se modifica el estado del bus DALI según el tipo de bit a enviar (0 ó 1).

En la librería DALI se incluyen otras funciones que no se utilizan de forma directa en la implementación del montaje final del sistema de control del banco de ensayos, pero que sí se

utilizaron en las pruebas previas de funcionamiento del bus DALI utilizando la comunicación con el usuario por puerto serie en vez de por bluetooth.

4.3. Programación de la aplicación

El dispositivo que el usuario dispondrá para controlar el banco de ensayos es una Tablet con sistema operativo Android, tal como ya se ha anticipado en apartados previos.

Android es un sistema operativo móvil desarrollado por Google y basado en el kernel de Linux. Es uno de los sistemas operativos más utilizados en dispositivos portátiles, con una cuota de mercado en el año 2017 superior al 80 % (wikipedia). Esto facilita la extrapolación de la app programada a otros proyectos, no necesariamente vinculados al uso del equipo disponible en el laboratorio.

Para la construcción de la aplicación se utiliza el entorno oficial de desarrollo Android Studio, basado en el IDE IntelliJ IDEA, desarrollado por JetBrains. El sistema base de compilación que utiliza es Gradle, lo que aporta gran flexibilidad a la hora de reutilizar códigos y generar diferentes APK (“Android Application Package”) a partir de un único proyecto (Android developers, 2020).

La estructura de Android Studio se divide en varias ventanas de visualización y barras de herramientas, tal como se muestra en la siguiente figura:

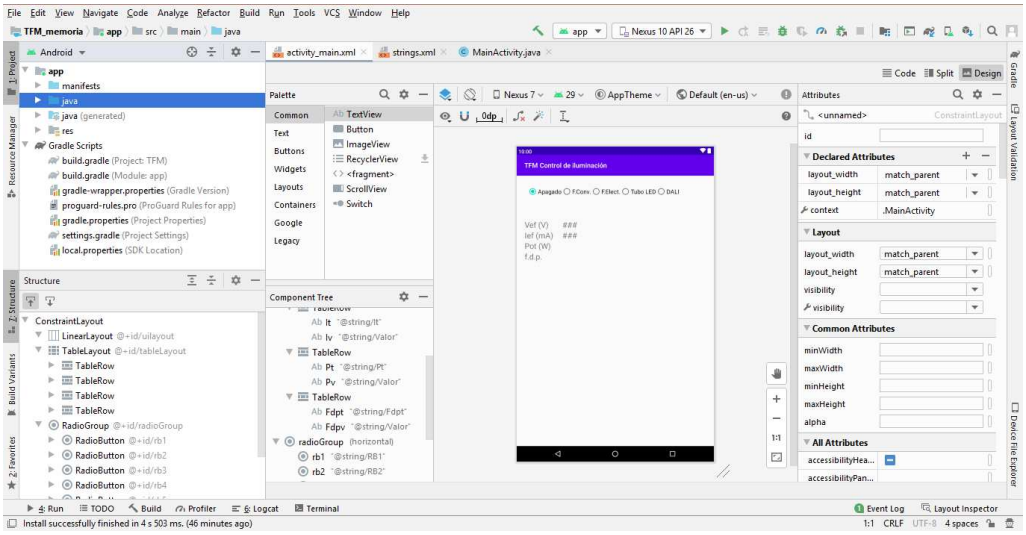


Figura 69. Android Studio IDE

No es objeto de este trabajo describir en detalle el funcionamiento del entorno Android Studio, por lo que sólo se comentarán las características más relevantes para comprender el proceso de programación a través de este IDE.

Al crear un nuevo proyecto en Android Studio es necesario especificar para qué versión de Android será compatible la aplicación y en qué lenguaje se desea programar. Una característica particular de este IDE es que admite la programación en diferentes lenguajes, tales como C++, Kotlin o Java. En concreto, en este trabajo se utilizará Java para la programación de la parte lógica del programa, ya que se trata del lenguaje más extendido para

la construcción de aplicaciones para Android. Por otra parte, se configura la compatibilidad de la app con dispositivos con una versión de Android igual o superior a 4.0.3[15].

Una vez creado el proyecto con una actividad vacía[16] se procede a programar la parte gráfica y la parte lógica de la aplicación. La interfaz gráfica se puede desarrollar a partir de un visor o directamente completando el código en lenguaje xml. La parte lógica se programa con el lenguaje Java.

Java es un lenguaje que utiliza el paradigma de la programación orientada a objetos. Los métodos (algoritmos asociados a un objeto) y atributos (características de una clase) están directamente relacionados, a diferencia de la programación estructurada tradicional, en la que datos y procedimientos se operan de forma separada. Esto condiciona la lógica de programación: no se definen funciones que procesan datos, sino que se instancian objetos a los que se envían mensajes que provocan la ejecución de sus métodos.

Otra característica del lenguaje Java, relacionada con la Máquina Virtual Java (JVM), es la posibilidad de ejecutar simultáneamente varios hilos o subrutinas, a través de la extensión de la clase "Thread". La interacción con la subrutina se realiza con un manejador o "Handler", que proporciona las instrucciones que debe ejecutar ("message"). Esto permite controlar de forma simultánea el entorno gráfico de la app y las comunicaciones por bluetooth (Sistemas Distribuidos:Hilos (Threads) en Java, 2014).

El conjunto del código que configura la aplicación, tanto la parte lógica como la parte gráfica, se incluyen en el anexo (apartado 9). En los siguientes apartados sólo se comentarán los aspectos más relevantes de la programación de la app.

4.3.1. Especificaciones de la aplicación

La aplicación constituye la plataforma a partir de la que el usuario puede controlar el banco de ensayos y recibir los datos de medida. Debe ser lo más sencilla e intuitiva posible para que el usuario no necesite ningún conocimiento previo para su utilización. Se recuerda que el objetivo final de este trabajo es el de servir de apoyo en la realización de prácticas didácticas, en las que la dificultad para los alumnos debe ser la comprensión de los fundamentos de diferentes tecnologías de iluminación, y no el uso de una herramienta informática.

El conjunto de acciones principales que debe gestionar la app se resumen en los siguientes puntos:

- **Control del encendido y apagado de los sistemas y regulación del flujo luminoso**. Se utilizarán opciones gráficas que no susciten incertidumbre sobre el resultado producido.
- **Visualización en tiempo real de las variables eléctricas del banco de ensayos**. Esto se debe realizar mostrando una gráfica con las formas de onda de tensión y corriente, así como los resultados numéricos de tensión y corriente eficaz, potencia y factor de potencia.
- Envío de órdenes al microcontrolador y recepción de las medidas muestreadas a partir del canal de **comunicación bluetooth**. Este proceso debe ser continuo y paralelo a la

[15] Las tablets del laboratorio disponen de la versión 4.2.
[16] Una actividad es cada una de las ventanas emergentes de una aplicación.

ejecución principal de la app, evitando los posibles conflictos entre envíos y recepciones simultáneas.

Independientemente de los requerimientos específicos, la aplicación debe proporcionar un funcionamiento estable y sin errores, que pudiesen provocar el deterioro de los equipos o afectar a la seguridad de las personas.

4.3.2. Interfaz gráfica

Constituye el entorno sobre el que se mostrarán los controles y los resultados de las medidas realizadas. Para esta aplicación sólo se utilizará una única ventana emergente o activity, que se programa de forma visual mediante la selección, arrastre y colocación en el lugar adecuado de cada componente a mostrar.

La siguiente imagen muestra el conjunto de menús y módulos de construcción de la parte gráfica de la app.

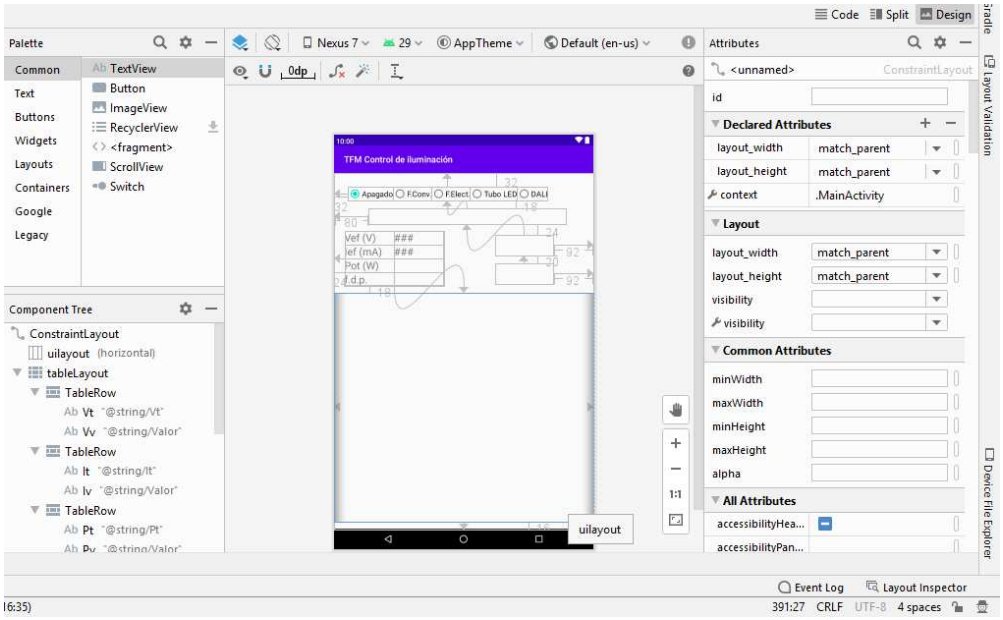


Figura 70. Programación de la interfaz gráfica en Android Studio

Una vez posicionado cada elemento con el tamaño y los márgenes que fijan su posición, se completan las características relevantes del mismo. En todos los componentes se debe especificar un identificador, que será el que se utilice para relacionar la parte gráfica con la parte lógica del programa.

Los elementos que se han utilizado en este trabajo son los siguientes:

- **"RadioGroup" con 5 "RadioButton"**. Establece la selección del modo de operación del banco de ensayos en base a la selección de un único botón simultáneamente. Esto asegura que no puedan estar encendidos varios dispositivos en el banco de ensayos, lo que provocaría una interpretación errónea de los datos capturados. Existen 5 estados posibles:
    - Apagado general
    - Encendido fluorescente con balasto convencional
    - Encendido fluorescente con balasto electrónico

- Encendido tubo led
- Operación bus DALI

- **"TableLayout"**. Contiene las etiquetas y los datos numéricos de las variables eléctricas a mostrar. Estas son tensión y corriente eficaz, potencia y factor de potencia. Las medidas de tensión y corriente se muestran en todos los modos de operación salvo en modo DALI[17]. La potencia y el factor de potencia se visualizan siempre excepto en modo apagado general y operación DALI.
- **"LinearLayout"**. Configura la ventana sobre la que se ubica la gráfica con las señales de tensión y corriente del banco de ensayos, que se construye en la parte lógica. Se visualiza en todos los modos de operación excepto en modo DALI.
- **"SeekBar"**. Barra de desplazamiento lateral que permite configurar el nivel de atenuación. Desplazamiento nulo (posición más a la izquierda) implica atenuación nula y flujo luminoso máximo, y desplazamiento máximo (posición derecha) supone atenuación máxima y flujo mínimo. Esta barra no se visualiza cuando está seleccionado el radiobutton de apagado o de fluorescente con balasto convencional (sin capacidad de regulación del flujo).
- **Botones de encendido y apagado de luminaria DALI**. Sólo se visualizan cuando está seleccionado el radiobutton de operación bus DALI. Permiten un encendido todo/nada de la luminaria, aparte de la regulación del flujo realizada con el seekbar.

En las siguientes figuras se incluyen los elementos descritos que configuran la interfaz gráfica de la aplicación.

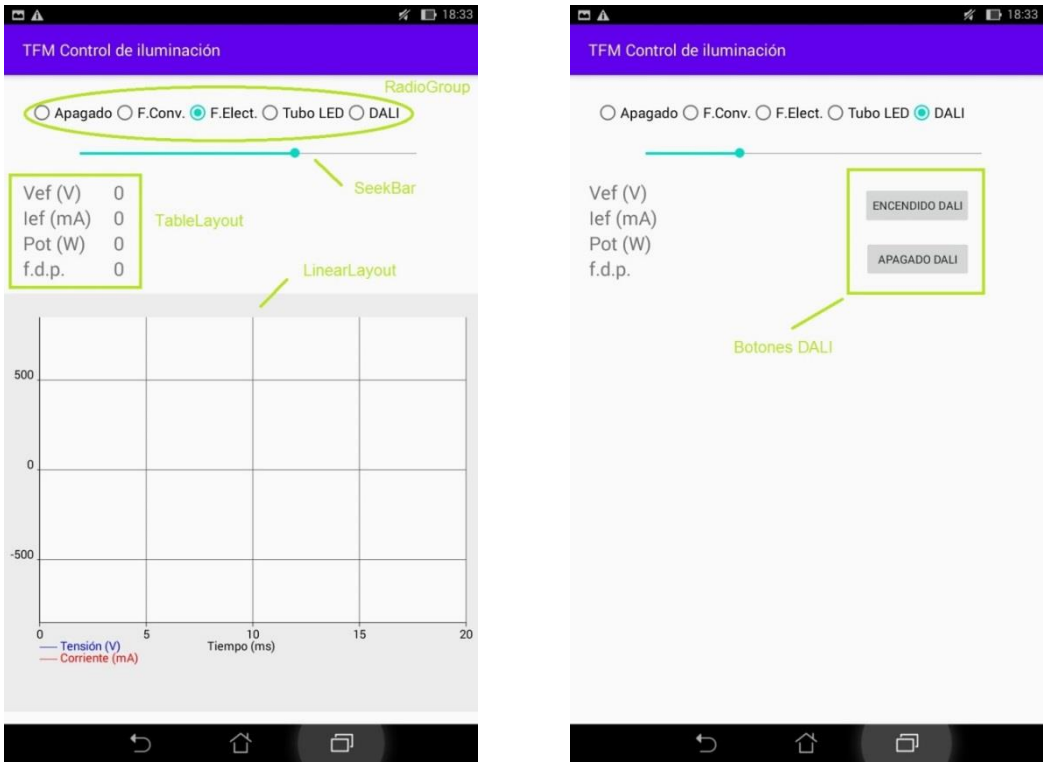


Figura 71. Pantallas de ejemplo de la interfaz gráfica

[17] Los sensores de tensión y corriente están en la entrada del circuito de potencia a los tubos. El bus DALI forma un circuito independiente, por lo que no tiene sentido visualizar estas variables cuando no son relevantes.

4.3.3. Parte lógica

La programación de la lógica funcional de la aplicación se realiza con Java. El conjunto de código se divide en clases, que permiten la realización de las distintas funcionalidades de la app. Estas clases se incluyen dentro de la actividad principal "MainActivity", que también es una clase pero con la condición especial de que se ejecuta siempre al iniciar el programa.

Las clases involucradas en la comunicación bluetooth se han desarrollado a partir de la aplicación "IMIEScope", utilizada en la asignatura de Instrumentación y Monitorización de Instalaciones Energéticas impartida en el Máster Universitario en Tecnología Energética para el Desarrollo Sostenible de la Universitat Politècnica de València. Se han realizado modificaciones puntuales para adaptar estas clases al entorno de este trabajo.

El primer punto en la programación consiste en importar el conjunto de **librerías** utilizadas a lo largo del código. Este proceso es relativamente sencillo, ya que a la hora de escribir código el propio IDE sugiere la librería necesaria, y no requiere una explicación detallada. Solo es interesante destacar que se usan librerías pertenecientes a tres familias distintas:

- Android: para el establecimiento del enlace y vinculación por bluetooth; referenciación de los elementos gráficos con la función realizada; incorporación del handler, etc
- Java: para la gestión de procesos try/catch con posibilidades de bloqueo; vinculación de variables estáticas, etc
- Achartengine: para la construcción de la gráfica de visualización de las señales en el Linearlayout asignado a tal efecto.

Toda la programación de esta aplicación se realiza dentro de la MainActivity, a excepción de la inclusión de las librerías correspondientes. Sólo hay una activity, ya que con una sola pantalla es posible mostrar toda la información y controles de la app.

Tras instanciar los objetos que se van a utilizar a lo largo del programa se incluye el método **"onCreate"**, que es el encargado de iniciar la activity con la interfaz programada en la parte gráfica. Dentro de este método se realizan una serie de acciones:

- Se crean los objetos correspondientes a los elementos de la parte gráfica.
- Se programa el funcionamiento de la seekbar, que devuelve un entero (0-100) cada vez que el usuario suelte la barra.
- Se genera la gráfica de visualización de las señales de tensión y corriente. Se configuran los colores, márgenes, rangos de escalas, títulos y etiquetas, etc.
- Se comprueba el enlace bluetooth. Si sólo hay un dispositivo vinculado con nombre adecuado, se inicia "mConnectedThread" (subrutina encargada del envío y lectura de datos por bluetooth) y se envía un mensaje al "handler" conforme se ha establecido la conexión. En caso contrario, se notifica al usuario un error en la conexión a través de un "Toast.makeText".

Es importante destacar que, para establecer la comunicación entre la Tablet usada y el microcontrolador, antes de iniciar la app se deben vincular ambos dispositivos a través de la

aplicación de ajustes del dispositivo Android. Este proceso sólo se realiza una única vez, ya que el emparejamiento queda guardado en la memoria de la Tablet de forma indefinida[18].

Fuera del método "onCreate", se programan un conjunto de clases que se detallan a continuación.

La primera clase es el manipulador o **"handler"**. En esta clase se realiza la interacción del hilo principal, que controla la interfaz gráfica, con la subrutina de envío y recepción de datos por bluetooth (objeto "mConnectedThread").

Cuando la subrutina bluetooth detecta datos disponibles por el enlace, envía un mensaje al manipulador con el conjunto de información almacenada. El manipulador interpreta estos datos y decodifica la información recibida.

La forma de mandar datos del ESP32 consiste en un mensaje inicial con el string "sampling finished". A continuación, procede a enviar los vectores de tiempos, medidas de tensión y corriente de forma sucesiva y elemento a elemento, con una separación con ";" entre vectores y un salto de línea entre posiciones consecutivas[19].

Una vez almacenados los datos recibidos por bluetooth, se convierten a niveles de tensión y corriente que se puedan visualizar en la gráfica, y se realiza el cálculo de valores eficaces, potencia y factor de potencia.

Dentro del bucle que recorre el "buffer" bluetooth, se convierten los enteros de las medidas digitalizadas de resolución 12 bits a niveles de tensión y de corriente, tal como se aprecia en el siguiente fragmento de código:

```
//Se obtienen los valores de tensión leídos
ch0f = ch0m*3.3/4095;
ch1f = ch1m*3.3/4095;
//Se realiza la conversión a los valores equivalentes del circuito
ch0f = ch0f * 214.20;  //Tensión del circuito en V
ch1f = ch1f * 1000/2; //Corriente primario en mA
```

La equivalencia entre la tensión leída en cada pin analógico del ESP32 y el valor real de tensión o corriente en el circuito de potencia se obtiene deshaciendo el divisor de tensión y utilizando la relación de reducción de corriente en el transformador respectivamente. A continuación, se incluyen las fórmulas de conversión:

$$R_{tensión} = \frac{2 \cdot 220 \cdot 10^3 + 1970\,\Omega}{1970}\frac{\Omega}{\Omega} = 224.35$$

$$R_{corriente} = \frac{1}{100}\frac{V}{\Omega} \cdot 50\frac{A}{A} \cdot 1000\frac{mA}{A}$$

En los primeros ensayos se ha comprobado que la tensión leída era superior al valor esperado. Se ha realizado una medida de las resistencias del divisor, obteniéndose el valor de 210 kΩ para ambas, inferior al nominal pero dentro de la tolerancia propia de la serie (±5 %). De esta forma, la equivalencia de tensión queda de la siguiente forma:

[18] Aunque se realice el reinicio de la Tablet o del ESP32, no es necesario volver a realizar el emparejamiento.

[19] Se recomienda consultar el apartado 4.2.5, en el que se ha explicado en detalle esta codificación.

$$R_{tensión} = \frac{2 \cdot 210 \cdot 10^3 + 1970}{1970} \frac{\Omega}{\Omega} = 214.20$$

Al realizar el muestreo de una variable de entrada, se pasa de una señal continua a una señal discreta. Según el teorema de muestreo de Nyquist-Shannon, se podrá hacer una reconstrucción de la señal a partir de los valores discretos si la tasa de muestreo es mayor o igual al doble de la frecuencia de la señal continua. Esto se cumple con elevado margen en este trabajo, ya que la señal de entrada tiene frecuencia de 50 Hz y la tasa de muestreo es 5000 Hz (100 muestras por periodo).

La definición de valor eficaz (RMS) se aplica a funciones continuas en el tiempo, como se muestra en la siguiente ecuación.

$$F_{señal\ continua} = \sqrt{\frac{1}{T} \cdot \int_0^T f(t)^2 \cdot dt}$$

La fórmula equivalente para muestras discretas es la siguiente (Eduardo Calle, Juan Gutiérrez, & Ángel Orozco, 2004):

$$F_{muestras\ discretas} = \sqrt{\frac{\sum_N {x_i}^2}{N}}$$

siendo F el valor eficaz, x_i el valor de las muestras discretas y N el número total de muestras.

La potencia activa se obtiene como el sumatorio del producto de las muestras discretas de tensión y corriente, dividido entre el número de muestras (González, Plata, Pérez, Gualdrón, & Morantes, 2009):

$$P = \frac{\sum_N v_i \cdot i_i}{N}$$

El factor de potencia se calcula como el cociente entre la potencia activa y la aparente:

$$FP = \frac{P}{S} = \frac{\frac{\sum_N v_i \cdot i_i}{N}}{V_{rms} \cdot I_{rms}}$$

La implementación de estas operaciones se muestra en el siguiente fragmento de código:

```
for(int i =0; i < samples ; i++){
    //Elimina el valor medio
    ch0rms += Math.pow((ch0s.getY(0)-ch0a),2);
    ch1rms += Math.pow((ch1s.getY(0)-ch1a),2);
    //El índice es 0 debido al remove
    xvalue = ch0s.getX(0);
    yvalue = ch0s.getY(0) - ch0a;
    yvalue2 = ch1s.getY(0) - ch1a;
    ch0s.remove(0);
    ch1s.remove(0);
    ch0s.add(xvalue, yvalue);
    ch1s.add(xvalue, yvalue2);
    //Se calcula el valor de potencia instantánea
    pot = pot + (yvalue * yvalue2 / 1000);   // se divide por 1000: corriente
en mA
}
//Divide entre el número de muestras porque es equivalente
```

```
//200 us entre muestras, periodo de 20000 us; Relación 1/100
ch0rms = Math.sqrt(ch0rms/samples);
ch1rms = Math.sqrt(ch1rms/samples);

pot /= samples;

if (ch0rms != 0 && ch1rms !=0) {
    fdp = pot / (ch0rms * ch1rms / 1000);    //corriente en mA
}
else fdp = 0;
```

Además de la interpretación de los mensajes recibidos por Bluetooth, el manipulador recibe las órdenes solicitadas por el usuario a través de la interacción con la interfaz gráfica. Después de decodificar los datos entrantes procedentes del microcontrolador y mandar esta información a la interfaz gráfica para su visualización, se comprueba si hay alguna solicitud de envío pendiente[20]. Si este es el caso, se traslada esa información a la subrutina de gestión del bluetooth, que es la encargada en última instancia de realizar el envío hacia el ESP32.

Este procedimiento se muestra en el siguiente fragmento de código:

```
if (!(order == 0))
{
    switch (order){
        case 1:     //Apagado
            mConnectedThread.write("1");
            order = 0;
            break;
        case 2:     //Fluorescente Convencional
            mConnectedThread.write("2");
            order = 0;
            break;
        case 3:     //Fluorescente B.Electrónico
            mConnectedThread.write("3");
            order = 0;
            break;
        case 4:     //LED
            mConnectedThread.write("4");
            order = 0;
            break;
        case 5:     //Dali
            mConnectedThread.write("5");
            order = 0;
            break;
        case 6:     //Dali ON
            mConnectedThread.write("6");
            order = 0;
            break;
        case 7:     //Dali OFF
            mConnectedThread.write("7");
            order = 0;
            break;
        case 8:     //Seekbar
            if (Rb3.isChecked() || Rb4.isChecked() || Rb5.isChecked()) {
                mConnectedThread.write("8");
                mConnectedThread.write(String.valueOf(level));
```

[20] Sólo se permite el envío de órdenes al ESP32 si se están recibiendo datos del mismo, es decir, si la comunicación es totalmente bidireccional.

```
                order = 0;
            } else order = 0;
            break;
    }
}
```

La segunda clase programada dentro de la "MainActivity" es **la subrutina de comunicación por bluetooth**, denominada "**ConnectedThread**". Esta clase, ejecutada en paralelo al hilo principal de la aplicación, realiza una lectura continua de la información disponible en el enlace bluetooth.

Para evitar que la lectura de datos pueda ser incompleta, es decir, no contener las 100 muestras capturadas en cada ciclo del ESP32, se debe ajustar el tiempo de espera entre comprobaciones sucesivas del enlace bluetooth. Esto se consigue si el "SystemClock.sleep" de la subrutina bluetooth es mayor o igual al "delay" entre capturas de las 100 muestras en el ESP32.

Por otra parte, la subrutina bluetooth se encarga del envío de datos por el enlace bluetooth, cuando se recibe el mensaje de escritura gestionado por el manipulador o "handler".

Además de las dos clases comentadas, dentro de la "MainActivity" se implementan las funciones que detectan la interacción del usuario con la interfaz gráfica.

Cuando el usuario realiza una acción sobre la aplicación, se comprueba si no hay órdenes pendientes de envío al ESP32 y si ese es el caso, se reescribe el valor de la variable que codifica la orden solicitada. Esta modificación es interpretada por el manipulador, que envía el mensaje de escritura correspondiente al hilo de comunicación bluetooth.

Estas funciones gestionan la visualización u ocultación de los diferentes elementos de la interfaz gráfica en función del modo de operación vigente.

En el siguiente fragmento de código se incluye la implementación de estas funciones:

```
public void onRadioButtonClicked(View view) {
    switch(view.getId()) {
        case R.id.rb1:
            if (order == 0){
                order = 1;
            }
                B1.setVisibility(View.INVISIBLE);
                B2.setVisibility(View.INVISIBLE);
                seekBar.setVisibility(View.INVISIBLE);
                layout.setVisibility((View.VISIBLE));
                rms0.setVisibility(View.VISIBLE);
                rms1.setVisibility(View.VISIBLE);
                potencia.setVisibility(View.INVISIBLE);
                factorpotencia.setVisibility(View.INVISIBLE);
                break;
        case R.id.rb2:
            if (order == 0) {
                order = 2;
            }
                B1.setVisibility(View.INVISIBLE);
                B2.setVisibility(View.INVISIBLE);
                seekBar.setVisibility(View.INVISIBLE);
                layout.setVisibility((View.VISIBLE));
                rms0.setVisibility(View.VISIBLE);
```

```
                rms1.setVisibility(View.VISIBLE);
                potencia.setVisibility(View.VISIBLE);
                factorpotencia.setVisibility(View.VISIBLE);
                break;
        case R.id.rb3:
            if (order == 0){
                order = 3;
            }
                B1.setVisibility(View.INVISIBLE);
                B2.setVisibility(View.INVISIBLE);
                seekBar.setVisibility(View.VISIBLE);
                layout.setVisibility((View.VISIBLE));
                rms0.setVisibility(View.VISIBLE);
                rms1.setVisibility(View.VISIBLE);
                potencia.setVisibility(View.VISIBLE);
                factorpotencia.setVisibility(View.VISIBLE);
                break;
        case R.id.rb4:
            if (order == 0){
                order = 4;
            }
                B1.setVisibility(View.INVISIBLE);
                B2.setVisibility(View.INVISIBLE);
                seekBar.setVisibility(View.VISIBLE);
                layout.setVisibility((View.VISIBLE));
                rms0.setVisibility(View.VISIBLE);
                rms1.setVisibility(View.VISIBLE);
                potencia.setVisibility(View.VISIBLE);
                factorpotencia.setVisibility(View.VISIBLE);
                break;
        case R.id.rb5:
            if (order == 0){
                order = 5;
            }
                B1.setVisibility(View.VISIBLE);
                B2.setVisibility(View.VISIBLE);
                seekBar.setVisibility(View.VISIBLE);
                layout.setVisibility((View.INVISIBLE));
                rms0.setVisibility(View.INVISIBLE);
                rms1.setVisibility(View.INVISIBLE);
                potencia.setVisibility(View.INVISIBLE);
                factorpotencia.setVisibility(View.INVISIBLE);
                break;}}

public void Encender(View view){
    if (Rb5.isChecked()){
        if (order == 0){
            order = 6;
        }
    } else {}
}

public void Apagar(View view){
    if (Rb5.isChecked()){
        if (order == 0){
            order = 7;
        }
    } else {}
}
```

5. RESULTADOS

En este capítulo se presentan los resultados obtenidos tras el diseño e implementación del sistema de control de iluminación aplicado a un banco de ensayos. La estructura de este capítulo se resume en tres puntos principales:

- Comprobación del funcionamiento del ESP32 controlado por el usuario a partir de la aplicación para dispositivos Android desarrollada. Se utilizan capturas dentro de la app para mostrar el comportamiento real del sistema diseñado.
- Conclusiones de este trabajo.
- Futuras líneas de mejora y evolución del diseño acometido en este proyecto.

5.1. Ensayos finales

En este apartado se incluyen muestras del funcionamiento del sistema de control en los diferentes modos de operación posibles. Se realiza además un breve comentario sobre las características de cada tecnología de iluminación presente en el banco de ensayos, comprobando que los resultados obtenidos son coherentes con el funcionamiento real de estos equipos.

5.1.1. Apagado general

Al entrar a la aplicación se comprueba la conexión bluetooth. Si el vínculo es correcto aparece un mensaje temporal indicando que la app está conectada al ESP32. En caso contrario, se visualiza un mensaje indicando que no hay ningún dispositivo enlazado.

Si la conexión por bluetooth es correcta, se muestran en la pantalla los resultados de las medidas de tensión en V y corriente en mA, tal como se puede apreciar en la siguiente figura:

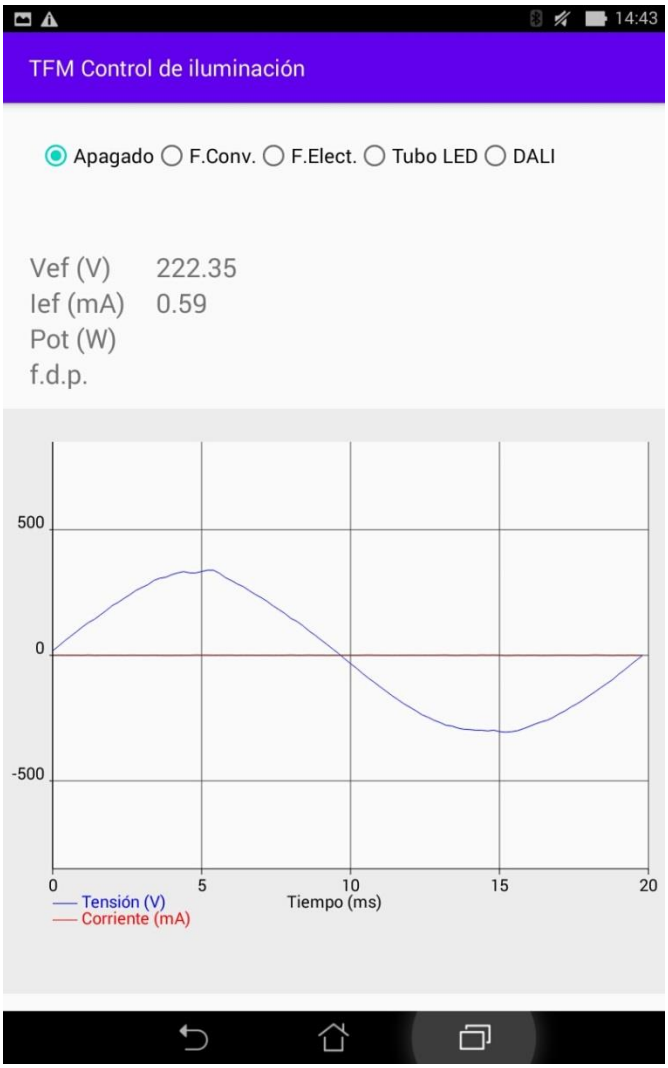


Figura 72. Aplicación en modo apagado general

Se aprecia un error de offset en la medida de la corriente. En todo caso, esta desviación es menor a 1 mA eficaz, por lo que se puede considerar despreciable.

La señal de tensión observada es prácticamente una senoidal pura, con un valor eficaz de 222.35 V (dentro del margen permitido de ±7 % sobre la tensión nominal de 230 V[21]).

5.1.2. Tubo fluorescente con balasto convencional

Una vez iniciada la aplicación y comprobada la conexión bluetooth con el dispositivo se procede al encendido del tubo fluorescente con balasto convencional. La siguiente figura muestra las medidas obtenidas en este modo de operación:

[21] El Real Decreto 1955/2000 especifica que los aspectos de calidad de producto (suministro eléctrico) seguirán los criterios establecidos en la norma UNE-EN 50.160.

Figura 73. Aplicación con el tubo fluorescente con balasto convencional en funcionamiento

Se puede apreciar con claridad la particular forma de la onda de corriente de este tipo de conjunto lámpara-balasto convencional. La corriente eficaz asciende a un valor 441.44 mA, registrándose una medida de potencia de 66.03 W y un factor de potencia de 0.67.

Se comprueba que el valor de corriente eficaz es cercano al que indica el fabricante del balasto como referencia (0.43 A).

5.1.3. Tubo fluorescente con balasto electrónico

En este apartado se comprueba el funcionamiento del tubo fluorescente con balasto electrónico para tres niveles de flujo luminoso diferente, ya que este equipo permite regulación del flujo a diferencia de lo que sucede con el tubo con balasto ferromagnético.

El primer caso analizado es el de flujo máximo (atenuación mínima).

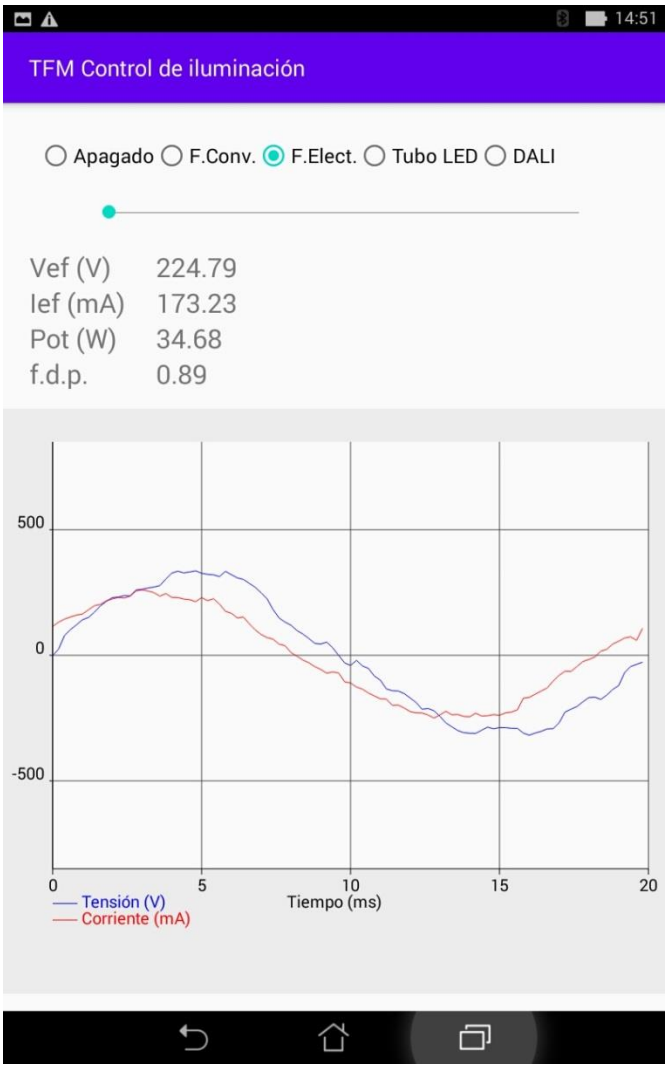


Figura 74. Aplicación con el tubo con balasto electrónico sin atenuación en funcionamiento

Se puede observar como la onda de corriente alcanza valores inferiores a los registrados para el fluorescente con balasto ferromagnético. La corriente eficaz presenta un valor aproximado de 173 mA, la potencia un valor cercano a 35 W y el factor de potencia un valor de 0.89.

Según el fabricante del balasto, el consumo de corriente debería estar en torno a 0.17 A, por lo que se podría concluir que los resultados de este ensayo se ajustan en gran medida a la realidad.

En comparación con el tubo con balasto convencional, el consumo de potencia se reduce más del 30 %, aunque la mayor reducción se produce en términos de corriente con una reducción superior al 60 %. Esto es un resultado conjunto de la reducción de potencia y del aumento del factor de potencia.

El siguiente caso que se presenta corresponde con una regulación del flujo luminoso al 75 % (atenuación del 25 %).

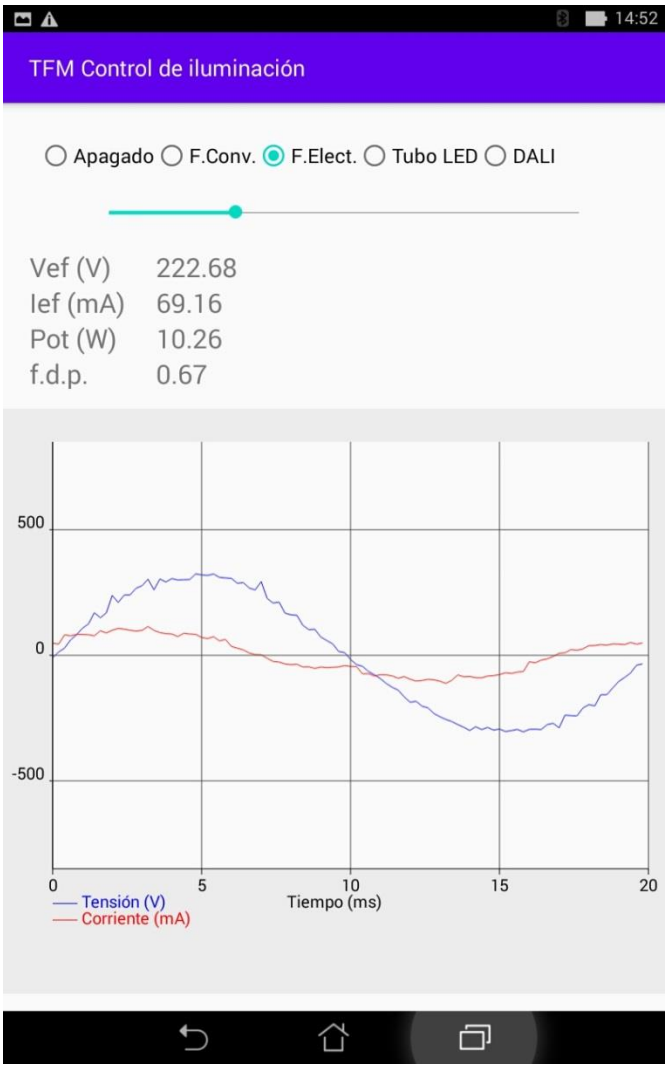


Figura 75. Aplicación con el tubo fluorescente con balasto electrónico y atenuación del 25 %

Se obtienen unos valores redondeados de corriente eficaz de 69 mA, potencia de 10 W y un factor de potencia de 0.67. Se puede apreciar con claridad la no linealidad entre la percepción del flujo luminoso (en torno al 75 % del total) y la potencia consumida (cercana al 30 % de la de flujo máximo).

Por otra parte, el factor de potencia disminuye hasta 0.67, valor idéntico al obtenido para el caso de tubo con balasto convencional. Esto es una característica propia de los balastos electrónicos. Cuando se realiza atenuación sobre el flujo luminoso a través de la regulación analógica 1-10 V, el factor de potencia empeora.

El último caso analizado se corresponde con una mayor atenuación, en torno al 85 % (flujo luminoso del 15 % sobre el máximo).

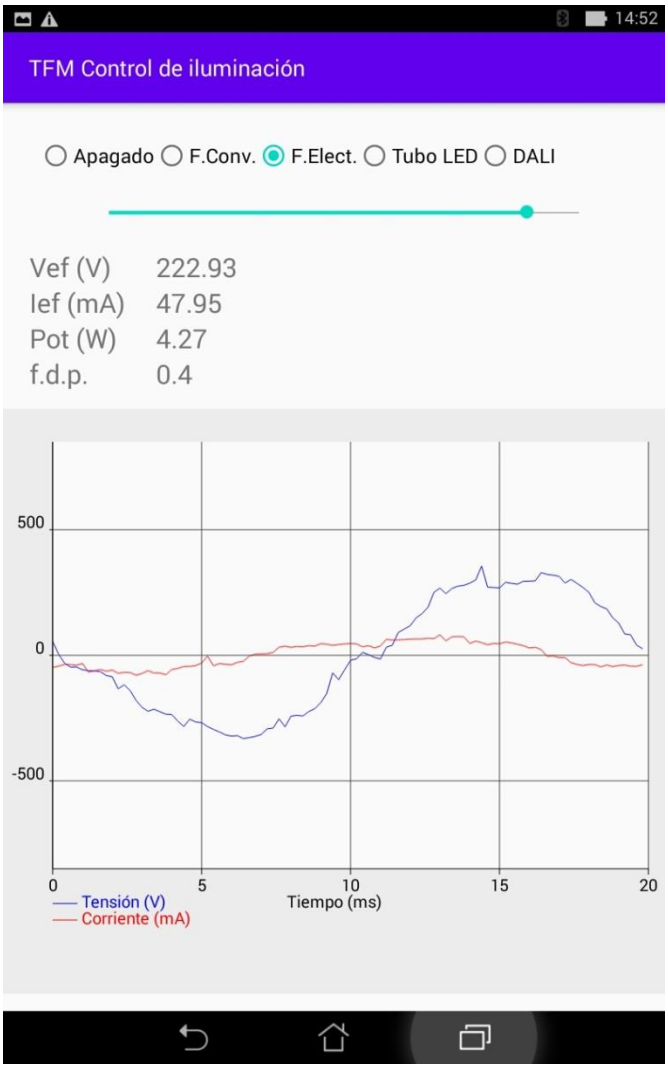


Figura 76. Aplicación con el tubo fluorescente con balasto electrónico y alta atenuación

En comparación con el ensayo con baja atenuación, tanto corriente como potencia y factor de potencia disminuyen su valor. La corriente presenta un valor de 47.95 mA, la potencia 4.27 W y el factor de potencia 0.4. Este hecho confirma el descenso en la corriente que circula por el tubo, así como la potencia consumida, al incrementar la atenuación, si bien la relación no es lineal con la percepción del flujo luminoso.

5.1.4. Tubo led

En este apartado se analizan los resultados obtenidos para tres niveles distintos de flujo luminoso, de forma análoga a lo realizado para el caso de tubo fluorescente con balasto electrónico.

El primer ensayo consiste en el funcionamiento del tubo led sin regulación.

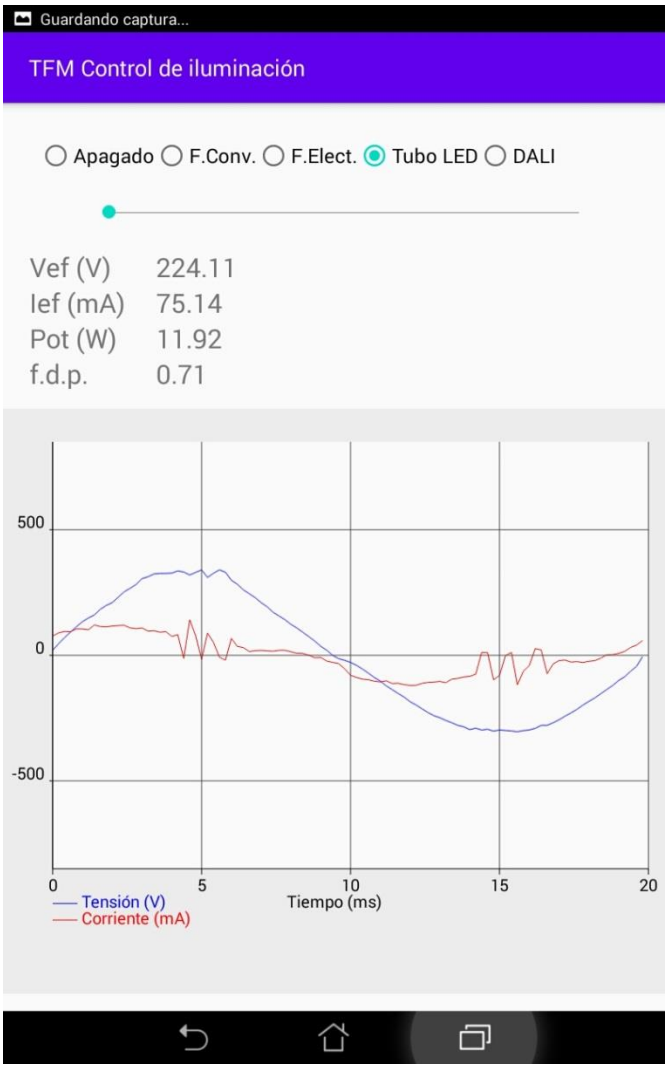


Figura 77. Aplicación con tubo led sin regulación en funcionamiento

Los valores registrados son una corriente eficaz de 75.14 mA, una potencia de 11.92 W y un factor de potencia de 0.71. La potencia teórica del tubo es 12 W, por lo que los resultados son coherentes con la realidad.

Aunque no es objeto de este trabajo el estudio de la eficiencia en la iluminación de las diferentes tecnologías, sí se puede comprobar que la corriente que circula por el conjunto tubo-dimmer led y la potencia consumida para el caso de flujo máximo presentan valores inferiores a los de la tecnología de iluminación por tubo fluorescente, para rangos de iluminancia similares entre los tres tubos analizados. Esto implica que la tecnología led proporciona una mayor eficiencia[22], sin entrar a valorar otros aspectos como puede ser el índice de rendimiento del color (IRC) o la fiabilidad.

También es importante señalar que el tubo led, por la limitación de potencia y corriente del driver led, se ha limitado a 1/3 de su capacidad (anulando un 33% de los chips led instalados, ver 2.2.3). Con un flujo luminoso especificado igual al del tubo fluorescente se tendría un consumo aproximadamente un 50% superior al registrado, es decir, unos 18W (aun así,

[22] Entendida como potencia luminosa de salida entre potencia eléctrica de entrada.

prácticamente la mitad del consumo del tubo florescente de igual flujo luminoso nominal con balasto electrónico).

Una característica que se aprecia en la señal de corriente son los picos que produce el controlador del led en cada semiciclo. El driver led actúa como una fuente de alimentación de intensidad constante. No obstante, la entrada de corriente sufre importantes variaciones producidas por la electrónica del dimmer. Este comportamiento no es un hecho aislado, puesto que se repite siempre que el tubo led funciona sin atenuación.

El segundo ensayo realizado se corresponde con un flujo luminoso cercano al 80 % (20 % de atenuación).

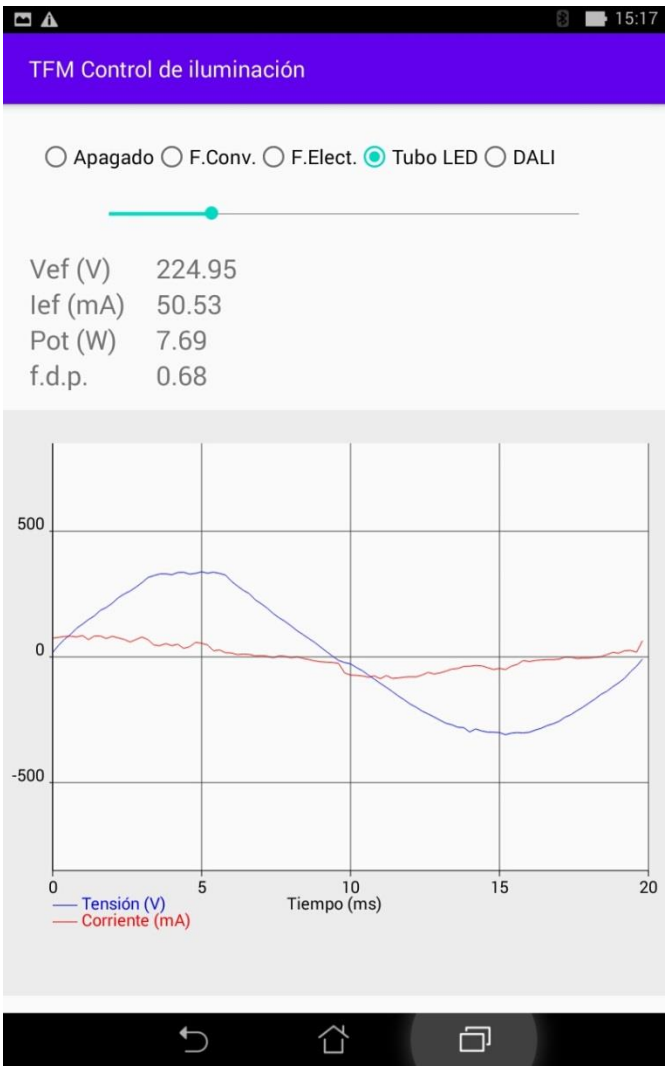


Figura 78. Aplicación con el tubo led con baja atenuación

Se registran valores de 50.53 mA de corriente eficaz, 7.69 W de potencia consumida con un factor de potencia de 0.68. El porcentaje de potencia consumida frente a la de flujo máximo es cercana al 65 %, por lo que se puede concluir que la relación entre flujo luminoso percibido y potencia no es perfectamente lineal.

El último caso presentado corresponde a una atenuación cercana al 85 % (15 % de flujo luminoso).

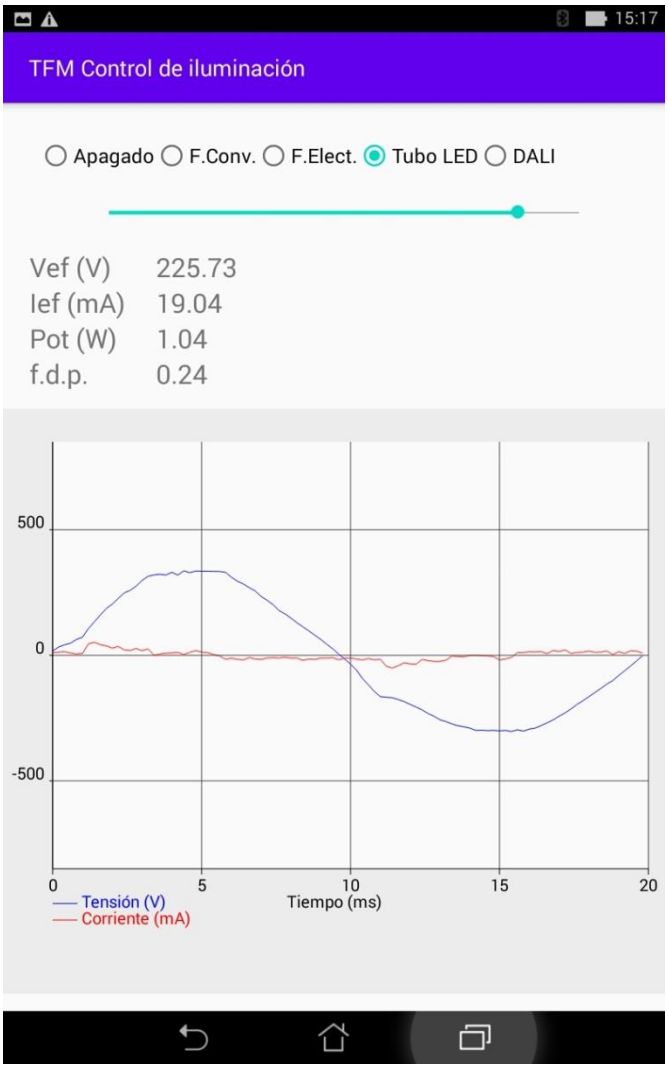


Figura 79. Aplicación con el tubo led con alta atenuación

Se obtienen unos valores aproximados de corriente eficaz de 19 mA y potencia de 1 W, con un factor de potencia de 0.24. Se cumple que, al reducir el flujo luminoso, la intensidad de entrada al controlador led y la potencia consumida disminuyen (aunque la relación no es perfectamente lineal). Por otra parte, el factor de potencia disminuye, al igual que sucedía al aumentar la atenuación en el tubo fluorescente con balasto electrónico.

Aunque para este nivel de flujo luminoso la corriente es tan reducida que apenas se identifica su forma de onda en la gráfica, sí que se aprecia el comportamiento oscilante de la misma.

5.1.5. Bus DALI

El resultado del ensayo DALI no se puede apreciar a partir de capturas de la aplicación. Cuando se selecciona el modo de operación DALI, la app deja de mostrar la gráfica de formas de onda y los resultados numéricos de las variables registradas. Esto sucede de esta forma porque los sensores de corriente y de tensión están ubicados en la entrada al banco de ensayos donde se

encuentran los tres tubos. La comunicación con la luminaria DALI se realiza a través de un bus totalmente independiente, en el que no tiene sentido medir consumos de potencia[23].

La siguiente figura muestra la interfaz gráfica durante el modo de operación DALI.

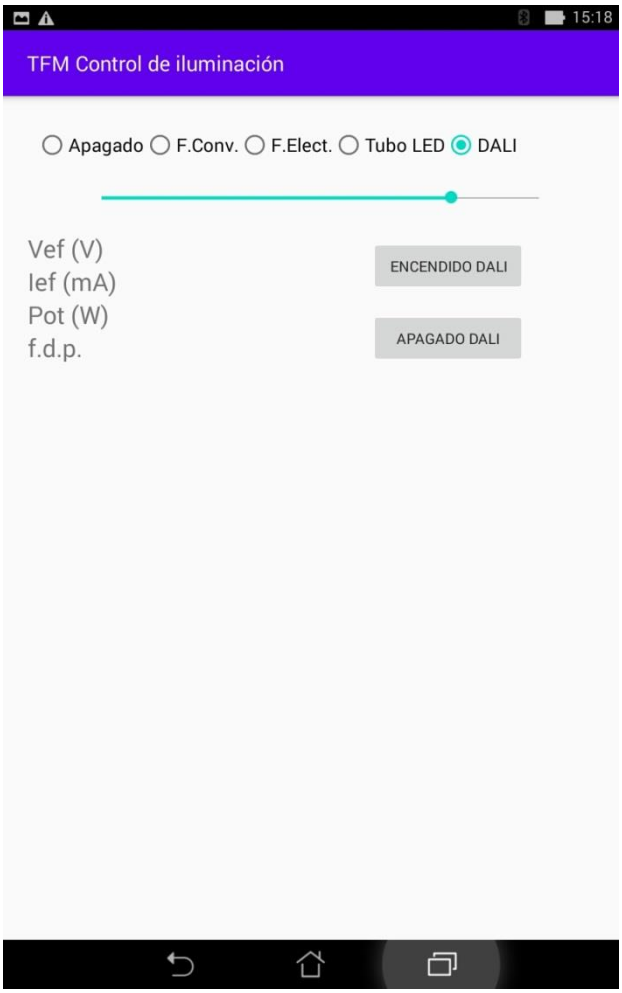


Figura 80. Aplicación en modo de operación DALI

La comprobación del correcto funcionamiento del sistema se realiza verificando que el esclavo DALI modifica la salida en el led de prueba, ante las peticiones por parte del usuario de encender, apagar o regular el flujo luminoso.

5.1.6. Resumen de los resultados obtenidos en cada ensayo

En la siguiente tabla se recoge el conjunto de medidas registradas en los diferentes ensayos:

	F. Conv	F. B. Electrónico			Led		
Flujo luminoso	100 %	100 %	75 %	15 %	100 %	80 %	15 %
Ief (mA)	441,44	173,23	69,16	47,95	75,14	50,53	19,04
Pot (W)	66,03	34,68	10,26	4,27	11,92	7,69	1,04
fdp	0,67	0,89	0,67	0,4	0,71	0,68	0,24

Tabla 4. Resumen de medidas obtenidas en los diferentes ensayos realizados

[23] La alimentación de los esclavos DALI se realiza a través de fuentes independientes, y no a partir del propio bus de comunicación.

5.2. Conclusiones

En este trabajo se ha completado el diseño e implementación de un sistema de control de iluminación de bajo coste, aplicado a un banco de ensayos de luminarias para su uso en actividades docentes.

El sistema desarrollado materializa el siguiente conjunto de funcionalidades básicas:

- Control del encendido y apagado de las diferentes luminarias instaladas en el banco de ensayos.
- Regulación del flujo luminoso mediante protocolos analógicos (0-1/10 V) y digitales (protocolo de comunicación DALI).
- Captura de medidas de tensión y corriente que permiten analizar las características eléctricas básicas de cada tecnología presente en el banco de ensayos.
- Comunicación por bluetooth entre el ESP32 y el dispositivo Android a través de la aplicación programada.
- Ejecución de operaciones de control sobre el banco de ensayos (a través del microcontrolador) de forma totalmente remota y sin ningún tipo de accionamiento mecánico directamente manipulado por el usuario.
- Interfaz gráfica de interacción con el usuario (mediante la app Android) para la selección de los modos de operación del banco y la visualización en tiempo real de las medidas capturadas, tanto de las formas de onda como de los valores numéricos representativos.

Los resultados obtenidos en los diversos ensayos de funcionamiento constituyen la prueba de que el sistema diseñado es funcional y cumple con los requerimientos establecidos en el encargo solicitado.

Es importante destacar que, aunque inicialmente se había planteado realizar el montaje final sobre una placa de circuito impreso (PCB), la situación derivada de la pandemia de COVID-19 supuso la desestimación de esta propuesta. Esto se fundamenta en el hecho de que no se haya recibido una formación mínima en soldadura que asegure la correcta operación de ensamblaje del circuito en condiciones de seguridad y que se haya tenido que realizar el montaje en la vivienda particular del alumno firmante de este trabajo, al continuar las restricciones de acceso al laboratorio a fecha de redacción del presente documento.

5.3. Futuras líneas de desarrollo

El sistema diseñado en este trabajo está directamente pensado para ser usado en prácticas de laboratorio, mostrando el funcionamiento de diversas tecnologías de iluminación. No obstante, se podría extrapolar al ámbito de la domótica[24] o incluso a un ámbito industrial (extremando las garantías de fiablididad en estos entornos agresivos para los sistemas electrónicos).

De forma más concreta, sobre el producto final obtenido en este proyecto, se presentan una serie de mejoras e incorporaciones adicionales que podrían incrementar las funcionalidades del sistema, más allá de las especificaciones solicitadas en este trabajo:

[24] Se entiende por este concepto al conjunto de sistemas capaces automatizar una vivienda, aportando servicios de gestión energética (wikipedia, 2020).

- **Generación de informes** con los datos capturados y guardado en la memoria del dispositivo o envío por email. Este añadido permitiría disponer de la información capturada sin necesidad de realizar capturas de la aplicación, favoreciendo a los alumnos en su proceso de realización del informe de prácticas.
- **Implementación completa de las posibilidades de control del bus DALI**. Este protocolo permite el envío de una gran cantidad de comandos de configuración de los esclavos, así como la recepción de información ante solicitudes de consulta del estado de funcionamiento. Si bien en la aplicación final es posible el encendido y regulación de la luminaria DALI, no se puede enviar cualquier tipo de comando ni visualizar la respuesta de los esclavos.

  Este añadido se podría implementar a partir de una nueva activity dentro de la app al seleccionar el modo de operación DALI.
- **Incorporación de nuevos sensores que permitan visualizar las tramas de comunicación en el bus DALI**. Esta medida posibilitaría el estudio del comportamiento del bus sin necesidad de utilizar osciloscopios, de forma adicional al propio conjunto banco de ensayos-microcontrolador-aplicación Android.
- **Implementación de sensores de medida del flujo luminoso**. Esta incorporación permitiría analizar y comparar las diferentes tecnologías incorporadas en el banco de ensayos desde el punto de vista de la eficiencia energética global, entendida como potencia luminosa dividida por potencia eléctrica consumida.

6. PRESUPUESTO

En este apartado se incluye el presupuesto asociado al diseño e implementación del sistema de control de iluminación desarrollado en este trabajo.

6.1. Consideraciones generales

El presupuesto está constituido como si el autor del presente trabajo desempeñase un cargo de ingeniero industrial dentro de una oficina técnica, a la que se realiza el encargo de este proyecto.

En el coste de materiales, se incluye el asociado a todos los elementos que forma parte directamente del sistema de control y del banco de ensayos. El coste asociado al uso de equipos auxiliares (multímetro, osciloscopio, soldador, etc) se computa dentro de los costes directos complementarios, que se fijan en el 5 %.

Por la naturaleza singular y de ejecución única de este trabajo, no se incluye un estado de mediciones, ya que cada unidad de obra se ejecuta una sola vez.

Los gastos de carácter general se estiman en un 13% y el beneficio industrial en un 6 %, ambos aplicados sobre el presupuesto de ejecución material (PEM). El impuesto sobre el valor añadido (IVA) es del 21 % en la fecha de redacción de este documento.

6.2. Mano de obra

En la realización de este proyecto sólo ha intervenido de forma directa el autor del trabajo. La retribución asociada a su actividad se calcula a partir de lo establecido en el XIX Convenio colectivo del sector de empresas de ingeniería y oficinas de estudios técnicos.

La siguiente tabla describe el cálculo de la retibución en base a la que se ha redactado este presupuesto.

Puesto	Salario bruto anual (€/año)	Salario bruto por hora (€/hora)
Ingeniero industrial	30000	14,42

Tabla 5. Restribución mano de obra

6.3. Materiales

Los materiales que forman parte del diseño final del sistema de control y del banco de ensayos se desglosan en la siguiente tabla. El coste asociado a estos elementos se incluye en la unidad de obra nº5 (Montaje y ensayos finales).

Descripción	Cantidad (ud.)	Precio unitario (€/ud)	Total (€)
Tubo fluorescente 36 W	2	2,20	4,40
Balasto convencional Inecsa	1	4,95	4,95
Balasto electrónico Quicktronic	1	14,41	14,41
Cebador	1	1,20	1,20
Tubo led 18 W	1	6,95	6,95
Dimmer led	1	19,95	19,95
Luminaria DALI	1	29,99	29,99
Fuente 24 V DC (DALI)	1	24,19	24,19
Fuente 12 V DC	1	2,95	2,95
Tablet Android	1	79,99	79,99
Placa de prototipos	4	2,49	9,96
Lote cables Dupont	1	5,75	5,75
ESP32 DevKit v1	1	7,00	7,00
Relé PCH-105L2M	3	1,70	5,10
Transistor NPN BC337-40	6	0,42	2,52
Transformador Talema AX-0500	1	2,49	2,49
Resistencia	18	0,10	1,80
Condensador electrolítico	2	0,59	1,18
		Total	**224,78**

Tabla 6. Materiales utilizados

6.4. Unidades de obra y precios descompuestos

Las siguientes tablas contienen el coste asociado a cada unidad de obra de este proyecto.

Nº de Orden		**Descripción de la unidad de obra**		
U.O.1		ANÁLISIS DEL ESTADO DEL ARTE Revisión bibliográfica y análisis de las tecnologías de iluminación a incorporar en el banco de ensayos y sus antecedentes.		
Costes directos				
Rendimiento		**Descripción**	**Precio**	**Importe**
	15 h	Ingeniero industrial	14,42 €/h	216,30 €
	5%	Costes directos complementarios		10,82 €
TOTAL PRECIO UNIDAD DE OBRA				**227,12 €**

Tabla 7. Unidad de obra nº1. Análisis del estado del arte

Nº de Orden		Descripción de la unidad de obra		
U.O.2		DISEÑO DEL HARDWARE		
		Diseño de los circuitos e interfaces de control de las luminarias del banco de ensayos, atendiendo a las especificaciones marcadas en el encargo inicial. Montajes preliminares de verificación de funcionamiento.		
Costes directos				
Rendimiento		**Descripción**	**Precio**	**Importe**
	75 h	Ingeniero industrial	14,42 €/h	1.081,50 €
	5%	Costes directos complementarios		54,08 €
TOTAL PRECIO UNIDAD DE OBRA				**1.135,58 €**

Tabla 8. Unidad de obra nº2. Diseño del hardware

Nº de Orden		Descripción de la unidad de obra		
U.O.3		ESTUDIO DEL PROTOCOLO DALI		
		Revisión de normativas de funcionamiento del protocolo DALI, diseño del bus de comunicación y ensayos con control vía puerto USB.		
Costes directos				
Rendimiento		**Descripción**	**Precio**	**Importe**
	50 h	Ingeniero industrial	14,42 €/h	721,00 €
	5%	Costes directos complementarios		36,05 €
TOTAL PRECIO UNIDAD DE OBRA				**757,05 €**

Tabla 9. Unidad de obra nº3. Estudio del protocolo DALI

Nº de Orden		Descripción de la unidad de obra		
U.O.4		PROGRAMACIÓN ESP32 Y APP ANDROID		
		Diseño del código ejecutado por el microcontrolador y la aplicación Android. Verificación del funcionamiento adecuado y comunicación efectiva vía bluetooth.		
Costes directos				
Rendimiento		**Descripción**	**Precio**	**Importe**
	100 h	Ingeniero industrial	14,42 €/h	1.442,00 €
	5%	Costes directos complementarios		72,10 €
TOTAL PRECIO UNIDAD DE OBRA				**1.514,10 €**

Tabla 10. Unidad de obra nº4. Programación ESP32 y app Android

Nº de Orden		Descripción de la unidad de obra		
U.O.5		MONTAJE Y ENSAYOS FINALES		
		Integración de los montajes previos en el montaje final y verificación del funcionamiento global del sistema diseñado.		
Costes directos				
Rendimiento		**Descripción**	**Precio**	**Importe**
	20 h	Ingeniero industrial	14,42 €/h	288,40 €
	1 ud	Montaje final	224,78 €/ud	224,78 €
	5%	Costes directos complementarios		25,66 €
TOTAL PRECIO UNIDAD DE OBRA				**538,84 €**

Tabla 11. Unida de obra nº5. Montaje y ensayos finales

Nº de Orden		Descripción de la unidad de obra		
U.O.6		REDACCIÓN DOCUMENTACIÓN TÉCNICA		
		Registro por escrito de las tareas realizadas y de los resultados obtenidos en el conjunto de este trabajo.		
Costes directos				
Rendimiento		**Descripción**	**Precio**	**Importe**
	40 h	Ingeniero industrial	14,42 €/h	576,80 €
	5%	Costes directos complementarios		28,84 €
TOTAL PRECIO UNIDAD DE OBRA				**605,64 €**

Tabla 12. Unidad de obra nº6. Redacción documentación técnica

6.5. Presupuesto de ejecución por contrata

Análisis del estado del arte ……	227,12 €
Diseño del hardware ……	1.135,58 €
Estudio del protocolo DALI ……	757,05 €
Programación del ESP32 y app Android ……	1.514,10 €
Montaje y ensayos finales ……	538,84 €
Redacción documentación técnica ……	605,64 €
Presupuesto de Ejecución Material ……	4.778,33 €
Gastos generales (13 %) ……	621,18 €
Beneficio industrial (6%) ……	286,70 €
Base imponible ……	5.686,21 €
IVA (21 %) ……	1.194,10 €
Presupuesto de Ejecución por Contrata	6.880,32 €

El presupuesto de ejecución por contrata asciende a la cantidad expresada de:

SEIS MIL OCHOCIENTOS OCHENTA EUROS CON TREINTA Y DOS CÉNTIMOS

En Valencia, a 20 de Julio de 2020

7. BIBLIOGRAFÍA

AENOR. (s.f.). *EN 62386-101:2014.*

Android developers. (2020). *Developers.* Recuperado el 2020, de Android Studio: https://developer.android.com/studio/intro?hl=es-419

Arteche. (2018). *Teoría y tecnología de los transformadores de medida.*

Blanco Espinosa, P. Á. (2016). *Apuntes de Iluminación, 3ºCurso , Grado en Ingeniería Eléctrica.* Universitat Politècnica de València.

Departamento de Informática. (2014). *Sistemas Distribuidos:Hilos (Threads) en Java.* Universidad de Valladolid.

Departamento de Ingeniería Eléctrica. (s.f.). *Práctica 6. Alumbrado. Medidas de iluminación con balasto electrónico.* Universitat Politècnica de València.

Digital Illumination Interface Alliance. (2020). *Digital Illumination Interface Alliance*. Obtenido de https://www.digitalilluminationinterface.org/

Eduardo Calle, J., Juan Gutiérrez, J., & Ángel Orozco, Á. (2004). *Medición de variables eléctricas utilizando señales digitalizadas.* Scientia et Technica.

Egido Nieto, D., Frías Marín, P., & Egido Cortés, I. (2015). *Diseño y construcción de un equipo de medida de energía eléctrica de bajo coste.* Universidad Pontificia Comillas ICAI-ICADE.

Ela Nsogo Nsa, S., & Parra Rodríguez, F. (2018). *Despliegue de una red Manet para la captura de la señal radiada por emisores bluetooth LE en entronos de microlocalización.* Escuela Politécnica Superior de Linares. Universidad de Jaén.

Encinas Elvira, A., & del Valle González, M. I. (2018). *Desarrollo de una placa electrónica compatible con DALI para el control de luminarias.* Universidad de Valladolid.

Espressif Systems. (2016). *ESP32-DevKit_v1_Schematics.*

Espressif Systems. (2020). *ESP32 Datasheet V3.3.*

Espressif Systems. (2020). *ESP32 Technical Reference Manual V4.2.*

Esteve Bosch, R., & Toledo Alarcón, J. F. (2005). *Fundamentos de Eletrónica Digital.* Universitat Politècnica de València.

Falcón Oubiña, P., & Armesto Quiroga, J. I. (2017). *Curso complementario de lenguaje C y Arduino.* Escola de Enxeñería Industrial. Universidade de Vigo.

García del Buey, D., Martínez Román, J. A., & Sapena Bañó, Á. (2019). *Desarrollo de un sistema de realimentación de corriente vectorial para banco de ensayos de accionamientos eléctricos.* Universitat Politècnica de València.

GEALed. (2016). *Control y Regulación SolidPowerSSD.*

giangrandi. (2017). *giangrandi.org.* Obtenido de Some measurements on a fluorescent tube and its magnetic ballast: https://www.giangrandi.org/electronics/fluorescenttubes/fluorescenttubes.shtml

González, E., Plata, Ó., Pérez, B., Gualdrón, D., & Morantes, B. (2009). *Medición de las magnitudes de potencia y energía eléctrica bajo las nuevas condiciones de los sistemas eléctricos.* Dialnet.

Martínez La Osa, P., Sapena Bañó, Á., & Martínez Román, J. A. (2018). *Diseño e implementación de un banco de ensayo de técnicas de iluminación industrial y su eficiencia energética para uso en prácticas de laboratorio.*

Martínez Pelayo, J., & Mena Rodríguez, J. M. (2019). *Módulo inalámbrico de señalización luminosa en vehículos de emergencias.* Universidad de Valladolid.

Martínez Pérez, P., & López Antón, A. (2016). *Creación de un MASTER DALI. Control de una/varias luminarias basado en un microcontrolador Cortex M4.* Universitat Oberta de Catalunya.

Muriel Stefanuto, I., Melquiades dos Santos, J. A., & Taveira Torres, C. (2016). *Evolução das Redes Sem Fio: Comparativo Entre Wi-Fi e Bluetooth.* Facultade de Tecnologia de Bauru.

NabiyevTR. (2017). *create.arduino.cc.* Recuperado el Julio de 2020, de Simple DALI Controller: https://create.arduino.cc/projecthub/NabiyevTR/simple-dali-controller-506e44

OSRAM. (2014). *Hoja de características QUICKTRONIC DE LUXE HF DIM T8.*

Pérez García, M. Á. (s.f.). *Instrumentación electrónica.* Paraninfo.

ProyectoArduino. (2020). Obtenido de https://proyectoarduino.com/arduino-uno-r3/

randomnerdtutorials. (2020). *esp32-pinout-reference-gpios*. Obtenido de https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

Roal Electronics. (s.f.). *Application Note 4 Meso50 DALI & PWM Dimming.*

Tocci, R. J., & Widmer, N. S. (2003). *Sistemas digitales. Principios y aplicaciones.* Prentice Hall.

Tridonic. (s.f.). *DALI manual.* Obtenido de https://www.tridonic.es/

Villarubia, M. (2000). *Seguridad Eléctrica: efectos de la corriente eléctrica sobre el cuerpo humano.* Facultad de Física. Universidad de Barcelona.

wikipedia. (2020). *wikipedia.org.* Recuperado el 2020, de Android: https://es.wikipedia.org/wiki/Android

Zhongshan dimmable lighting Electronics Co., Ltd. (2019). *Hoja de características PE18AA 18W Dimming Led Driver.*

ZVEI-Division Luminaires. (2001). *DALI AG Manual.*

8. ANEXO LIBRERÍA DALI

8.1. Archivo de cabecera: Dali.h

```
#ifndef dali_h
#define dali_h
// Comandos dali
#define BROADCAST_DP 0b11111110 //Broadcast Direct Power Arc Control
#define BROADCAST_C 0b11111111  //Broadcast Command
#define ON_DP 0b11111110        //Nivel máximo 254
#define OFF_DP 0b00000000       //Nivel mínimo 0
#define ON_C 0b00000101         //Nivel máximo por comandos
#define OFF_C 0b00000000        //Off
#define MIN_LEVEL 0b00000110    //Nivel mínimo por comandos
#define QUERY_STATUS 0b10010000
#define QUERY_LEVEL 0b10100000
#define RESET 0b00100000

#if defined(ARDUINO) && ARDUINO >= 100
  #include "Arduino.h"
#else
  #include "WProgram.h"
  #include <pins_arduino.h>
#endif

class Dali
{
  public:
  Dali(); //Constructor
  void setupTransmit(uint8_t pin);
  void setupAnalogReceive(uint8_t pin);
  void transmit(uint8_t cmd1, uint8_t cmd2);
  void scanShortAdd();
  void busTest();
  void initialisation();
  bool cmdCheck(String &input, int &cmd1, int &cmd2);
  uint8_t receive();
  int minResponseLevel();
  int maxResponseLevel();
  uint16_t delay1;
  uint16_t delay2;
  uint16_t period;
  String errorMsg; //Mensaje de error en la última operación
  bool msgMode;
  bool getResponse;
  uint8_t RxAnalogPin;
  long daliTimeout = 20000; //us, DALI response timeout   Tiempo entre
forward y backward (incluyendo trama backward)
                            //Se utiliza en varias funciones,
incluyendo Dali.receive
  int analogLevel = 870;    //Límite estado lógico (por debajo - "0";
por encima - "1")

  private:
  void sendByte(uint8_t b);
  void sendBit(int b);
  void sendZero(void);
  void sendOne(void);
  void splitAdd(long input, uint8_t &highbyte, uint8_t &middlebyte,
uint8_t &lowbyte);
  uint8_t TxPin;
```

```
  uint8_t rxAnalogPin;

};//Fin de la clase Dali

// Se utiliza para que el compilador de C ++ no agregue información de
tipo argumento / parámetro
extern "C"
{
   }
extern Dali dali;          //Permite disponer de un objeto dali externo
a usar por cualquier programa
#endif
```

8.2. Archivo de código fuente: Dali.cpp

```
#include "Dali.h"

Dali::Dali() //constructor
{
}

void Dali::setupAnalogReceive(uint8_t pin)
{
  RxAnalogPin = pin; // Se establece el pin como entrada
}

void Dali::setupTransmit(uint8_t pin)
{
  TxPin = pin; // Se establece el pin como salida
  pinMode(TxPin, OUTPUT);
  digitalWrite(TxPin, LOW);
  delay1=416;
  delay2=416;
  period = delay1 + delay2;   //Se usa en la recepción (Dali.receive)
  }

void Dali::transmit(uint8_t cmd1, uint8_t cmd2) // Transmisión de
comandos (address byte, command byte)
{
  sendBit(1);
  sendByte(cmd1);
  sendByte(cmd2);
  digitalWrite(TxPin, LOW);
}

void Dali::sendByte(uint8_t b)
{
  for (int i = 7; i >= 0; i--)
  {
    sendBit((b >> i) & 1);
  }
}

void Dali::sendBit(int b)
{
 if (b) {
    sendOne();
  }
  else {
    sendZero();
  }
}
```

```
void Dali::sendZero(void)
{
  digitalWrite(TxPin, LOW);
  delayMicroseconds(delay2);
  digitalWrite(TxPin, HIGH);
  delayMicroseconds(delay1);

}

void Dali::sendOne(void)
{
  digitalWrite(TxPin, HIGH);
  delayMicroseconds(delay2);
  digitalWrite(TxPin, LOW);
  delayMicroseconds(delay1);
}

void Dali::busTest() //Bus test para establecer nivel medio de tensión
{
  int maxLevel;
  int minLevel;

  //Encendido y apagado de luminarias para comprobar conexión
  delay(100);
  dali.transmit(BROADCAST_C, OFF_C); //Broadcast ON
  delay(500);
  dali.transmit(BROADCAST_C, ON_C); //Broadcast OFF
  delay(100);
  while (!Serial);

  //Recibe niveles máximos y mínimos de tensión en el bus dali
  dali.transmit(BROADCAST_C, QUERY_STATUS);
  maxLevel = dali.maxResponseLevel();
  dali.transmit(BROADCAST_C, QUERY_STATUS);
  minLevel = dali.minResponseLevel();
  dali.analogLevel = (int)(maxLevel + minLevel) / 2;
}


// Obtención del nivel mínimo
int Dali::minResponseLevel()
{
  const uint8_t dalistep = 40; //us
  uint16_t rxmin = 4095;
  uint16_t dalidata;
  long idalistep;
  for (idalistep = 0; idalistep < dali.daliTimeout; idalistep =
idalistep + dalistep) {
    dalidata = analogRead(RxAnalogPin);
    if (dalidata < rxmin) {
      rxmin = dalidata;
    };
    delayMicroseconds(dalistep);
  }
  Serial.println("minResponseLevel");
  Serial.println(rxmin);
  return rxmin;
}

// Obtención del nivel máximo
```

```
int Dali::maxResponseLevel()
{
  const uint8_t dalistep = 40; //us
  uint16_t rxmax = 0;
  uint16_t dalidata;
  long idalistep;
  for (idalistep = 0; idalistep < dali.daliTimeout; idalistep =
idalistep + dalistep) {
    dalidata = analogRead(dali.RxAnalogPin);
    if (dalidata > rxmax) {
      rxmax = dalidata;
    };
    delayMicroseconds(dalistep);
  }
  Serial.println("maxResponseLevel");
  Serial.println(rxmax);
  return rxmax;
}

//Escanea short addresses
void Dali::scanShortAdd()
{
  const int delayTime = 10;   //Tiempo entre tramas consecutivas
  const uint8_t start_ind_adress = 0;
  const uint8_t finish_ind_adress = 127;
  uint8_t add_byte;
  uint8_t device_short_add;
  uint8_t response;

  dali.transmit(BROADCAST_C, OFF_C); // Broadcast Off
  delay(delayTime);

  if (dali.msgMode) {
    Serial.println("Short addresses:");
  }

  for (device_short_add = start_ind_adress; device_short_add <= 63;
device_short_add++) {
    add_byte = 1 + (device_short_add << 1); // convierte short address
a address byte para usar con comandos (no DPL)
    dali.transmit(add_byte, 0xA1);   //Maximum lighting control
level    Query genérico para comprobar respuesta
    response = dali.receive();

    if (dali.getResponse) {
      dali.transmit(add_byte, ON_C); // Encendido
      delay(1000);
      dali.transmit(add_byte, OFF_C); // Apagado
      delay(1000);
    }
    else {
      response = 0;
    }

    if (dali.msgMode) {
      Serial.print("BIN: ");
      Serial.print(device_short_add, BIN);
      Serial.print(" ");
      Serial.print("DEC: ");
      Serial.print(device_short_add, DEC);
      Serial.print(" ");
```

```
      Serial.print("HEX: ");
      Serial.print(device_short_add, HEX);
      Serial.print(" ");
      if (dali.getResponse) {
        Serial.print("Get response");
      }
      else {
        Serial.print("No response");
      }
      Serial.println();
    }
    else {
      if (dali.getResponse) {
        Serial.println(255, BIN);
      }
      else {
        Serial.println(0, BIN);
      }
    }
  }
  dali.transmit(BROADCAST_C, ON_C); // Broadcast On
  Serial.println();
  delay(delayTime);
}

bool Dali::cmdCheck(String &input, int &cmd1, int &cmd2) //Comprueba
si el formato es el correcto
{
  bool test = true;
  input.replace(" ", "");   // Elimina espacios
  //Se comprueba longitud igual a 17 y no 16
  //Al pasar datos por serial se incluye \n
  if (input.length() != 17) {
    Serial.println("Fallo longitud");
    test = false; //Comprueba si hay 16 bits
  }
  else {
    //comprueba si los caracteres son ceros y unos (ASCII)
    for (int i = 0; i <= input.length() - 2; i++) {
      if (input.charAt(i) == '0' or input.charAt(i) == '1') {}
      else {
        test = false;
        Serial.println("Fallo caracter en la posición");
        Serial.println(i);
      };
    };
  };
  if (test) {
  int base2=1;
  int i=7;
  cmd1=0;
  cmd2=0;
  while (input.substring(0, 8).c_str()[i]!='\0' && i>=0){
  if (input.substring(0, 8).c_str()[i]=='1'){
    cmd1=cmd1+base2;
  }
  base2=base2*2;
  i--;
  }
  i=7;
  base2=1;
```

```
  while (input.substring(8, 16).c_str()[i]!='\0' && i>=0){
  if (input.substring(8, 16).c_str()[i]=='1'){
    cmd2=cmd2+base2;
  }
  base2=base2*2;
  i--;
  }
  }
  return test;
}

//Fragmentación de la long address
void Dali::splitAdd(long input, uint8_t &highbyte, uint8_t
&middlebyte, uint8_t &lowbyte)
{
  highbyte = input >> 16;
  middlebyte = input >> 8;
  lowbyte = input;
}

void Dali::initialisation()                //Inicialización de los
esclavos conectados al bus
{
  const int delaytime = 10; //ms
  long low_longadd = 0x000000;
  long high_longadd = 0xFFFFFF;
  long longadd = (long)(low_longadd + high_longadd) / 2;
  uint8_t highbyte;
  uint8_t middlebyte;
  uint8_t lowbyte;
  uint8_t short_add = 0;
  uint8_t cmd2;

  delay(delaytime);
  dali.transmit(BROADCAST_C, RESET);
  delay(delaytime);
  dali.transmit(BROADCAST_C, RESET);
  delay(delaytime);
  dali.transmit(BROADCAST_C, OFF_C);
  delay(delaytime);
  dali.transmit(0b10100101, 0b00000000); //Inicialización
  delay(delaytime);
  dali.transmit(0b10100101, 0b00000000); //Inicialización
  delay(delaytime);
  dali.transmit(0b10100111, 0b00000000); //Randomise
  delay(delaytime);
  dali.transmit(0b10100111, 0b00000000); //Randomise

  if (dali.msgMode) {
    Serial.println("Searching for long addresses:");
  }
  while (longadd <= 0xFFFFFF - 2 and short_add <= 64) {
    while ((high_longadd - low_longadd) > 1) {
      dali.splitAdd(longadd, highbyte, middlebyte, lowbyte); //divide
24bit adress into three 8bit adresses
      delay(delaytime);
      dali.transmit(0b10110001, highbyte); //Specifies the higher 8
bits of the search address
      delay(delaytime);
      dali.transmit(0b10110011, middlebyte); //search MB
      delay(delaytime);
```

```
      dali.transmit(0b10110101, lowbyte); //search LB
      delay(delaytime);
      dali.transmit(0b10101001, 0b00000000); //compare

      if (minResponseLevel() > dali.analogLevel)
      {
        low_longadd = longadd;  //No hubo respuesta, la random address
es mayor que la de búsqueda
        Serial.println("Sin respuesta a la comparación");
        Serial.println("daliAnalogLevel");
        Serial.println(dali.analogLevel);
      }
      else
      {
        high_longadd = longadd;
        Serial.println("daliAnalogLevel");
        Serial.println(dali.analogLevel);
        Serial.println("Slave ha contestado a la comparación");
      }

      longadd = (low_longadd + high_longadd) / 2; //center

      if (dali.msgMode) {
        Serial.print("BIN: ");
        Serial.print(longadd + 1, BIN);
        Serial.print(" ");
        Serial.print("DEC: ");
        Serial.print(longadd + 1, DEC);
        Serial.print(" ");
        Serial.print("HEX: ");
        Serial.print(longadd + 1, HEX);
        Serial.println();
      }
      else {
        Serial.println(longadd + 1);
      }
    } // Segundo while

    if (high_longadd != 0xFFFFFF)       //Comprueba que se haya
encontrado long random address
    {
      Serial.println("Se ha entrado al if de programación short
address");
      splitAdd(longadd + 1, highbyte, middlebyte, lowbyte);
      dali.transmit(0b10110001, highbyte); //search HB
      delay(delaytime);
      dali.transmit(0b10110011, middlebyte); //search MB
      delay(delaytime);
      dali.transmit(0b10110101, lowbyte); //search LB
      delay(delaytime);
      dali.transmit(0b10110111, 1 + (short_add << 1)); //program short
adress
      delay(delaytime);
      dali.transmit(0b10101011, 0b00000000); //withdraw
      delay(delaytime);
      dali.transmit(1 + (short_add << 1), ON_C);    //enciende slave
con short address
      delay(1000);
      dali.transmit(1 + (short_add << 1), OFF_C);
      delay(delaytime);
      Serial.println(short_add,BIN);
```

```
      short_add++;                                    //aumenta el short
address para asignárselo a un nuevo slave
      if (dali.msgMode) {
      Serial.println("Assigning a short address");
      }
      high_longadd = 0xFFFFFF;
      longadd = (low_longadd + high_longadd) / 2;
    }
    else {
      if (dali.msgMode) {
        Serial.println("End");     //Ha realizado la búsqueda completa
en todo el rango
                                    // de long address y no ha
encontrado nuevos slaves
      }
    }
  } // first while

  dali.transmit(0b10100001, 0b00000000);  //terminate
  delay(delaytime);
  dali.transmit(BROADCAST_C, ON_C);       //broadcast on
  delay(5000);
  dali.transmit(BROADCAST_C, OFF_C);
}

uint8_t Dali::receive()             //Recepción de señales
{
  unsigned long startFuncTime = 0;
  bool previousLogicLevel = 1;
  bool currentLogicLevel = 1;
  uint8_t arrLength = 20;
  int  timeArray[arrLength];            //Suficiente entero,
limitación por Dali.timeout 20ms=20000us
  int i = 0;
  int k = 0;
  bool logicLevelArray[arrLength];
  int response = 0;
  dali.getResponse = false;
  startFuncTime = micros();
  while (micros() - startFuncTime < dali.daliTimeout and i <
arrLength)
  {
    // obteniendo respuestas
    if (analogRead(dali.RxAnalogPin) >
dali.analogLevel)          //dali.RxAnalogPin se define en
Dali.setRxAnalogPin
    {
      currentLogicLevel = 1;
    }
    else {
      currentLogicLevel = 0;
    }

    if (previousLogicLevel != currentLogicLevel) {
      timeArray[i] = micros() - startFuncTime;
      logicLevelArray[i] = currentLogicLevel;
      previousLogicLevel = currentLogicLevel;
      dali.getResponse = true;
      i++;
    }
  }
```

```
  arrLength = i;
  //Modifica timeArray y logicArray para que contenga todos los
estados lógicos cada T/2
  for (i = 0; i < arrLength - 1; i++) {
    if ((timeArray[i + 1] - timeArray[i]) > 0.75 * dali.period) {
      for (k = arrLength; k > i; k--) {
        timeArray[k] = timeArray[k - 1];
        logicLevelArray[k] = logicLevelArray[k - 1];
      }
      arrLength++;
      timeArray[i + 1] = (timeArray[i] + timeArray[i + 2]) / 2;
      logicLevelArray[i + 1] =
logicLevelArray[i];                    //Se podría eliminar, igual a
logicLevelArray[k]
    }
  }
  k = 8;
  for (i = 1; i < arrLength; i++) {
    int y=round((timeArray[i] - timeArray[0]) / (0.5 * dali.period));
    if (logicLevelArray[i] == 1) {
      if (y & 1) {
        //Anota las posiciones impares, es decir el
estado

        //lógico de la segunda parte del paquete de
bits

        //coincide con el flanco Manchester
        //Round redondea al entero más cercano
        //& comparación bit a
bit


        response = response + (1 << k);
      }
      k--;
    }
   }
  response = (uint8_t)response;     //Elimina bit de inicio
  Serial.println("Respuesta sin bit de inicio");
  Serial.println(response,BIN);
  return response;
}

  Dali dali;
```

8.3. Archivo principal: Comunicación por puerto serie (USB)

```
#include "Dali.h"

const int DALI_TX = 25;
const int DALI_RX_A = 26;

#define BROADCAST_DP 0b11111110 //Broadcast Direct Power Arc Control
#define BROADCAST_C 0b11111111  //Broadcast Command
#define ON_DP 0b11111110        //Nivel máximo 254
#define OFF_DP 0b00000000       //Nivel mínimo 0
#define ON_C 0b00000101         //Nivel máximo por comandos
#define OFF_C 0b00000000        //Off
#define MIN_LEVEL 0b00000110    //Nivel mínimo por comandos
#define QUERY_STATUS 0b10010000
#define QUERY_LEVEL 0b10100000
```

```
#define RESET 0b00100000

void setup() {

  Serial.begin(115200);
  dali.setupTransmit(DALI_TX);
  dali.setupAnalogReceive(DALI_RX_A);
  dali.busTest();
  dali.msgMode = true;
  help(); //Mostrar ayuda
  Serial.println("dali.analogLevel");
  Serial.println(dali.analogLevel);
}


void help() {
  Serial.println("Enter 16 bit command or another command from
list:");
  Serial.println("help -  command list");
  Serial.println("on -  broadcast on 100%");
  Serial.println("off -  broadcast off 0%");
  Serial.println("scan -  device short address scan");
  Serial.println("initialise -  start process of initialisation");
  Serial.println("query - consulta de estado");
  Serial.println("level- consulta el nivel actual");
  Serial.println("min - nivel mínimo");
  Serial.println();
}

  int a=0;
  const int delaytime = 5000;
  int i;
  int cmd1;
  int cmd2;
  String comMsg;

void loop() {

  delay(delaytime);

 while (Serial.available() > 0)
    {
        char recieved = Serial.read();
        comMsg += recieved;

        // Se detecta recepción de mensaje cuando lee carácter de
salto de línea
        if (recieved == '\n')
        {
            Serial.print("Arduino Received: ");
            Serial.print(comMsg);
        }
    }

  if (comMsg == "scan\n") {
    dali.scanShortAdd();
    Serial.println("Recibido scan");
  };

  if (comMsg == "on\n") {
    dali.transmit(BROADCAST_C, ON_C);
```

```
    Serial.println("Recibido on");
  };

  if (comMsg=="off\n") {
    dali.transmit(BROADCAST_C, OFF_C);
    Serial.println("Recibido off");
  };

  if (comMsg == "initialise\n" or comMsg == "ini\n") {
    dali.initialisation();
    Serial.println("Recibido ini");
  };

  if (comMsg == "help\n") {
    help();
    Serial.println("Recibido help");
  };

  if (dali.cmdCheck(comMsg, cmd1, cmd2)) {
    Serial.println("Command check");
    Serial.println(cmd1,BIN);
    Serial.println(cmd2,BIN);
    dali.transmit(cmd1, cmd2);  // command in binary format: (address
byte, command byte)
  }

  if (comMsg == "query\n") {
    Serial.println("Recibido query");
    dali.transmit(BROADCAST_C,QUERY_STATUS);
    uint8_t response=dali.receive();
    Serial.println("Respuesta recibida");
    Serial.println(response,BIN);
  }

  if (comMsg == "min\n") {
    Serial.println("Recibido min");
    dali.transmit(BROADCAST_C,MIN_LEVEL);
  }

  if (comMsg == "level\n") {
    Serial.println("Recibido level");
    dali.transmit(BROADCAST_C,QUERY_LEVEL);
    uint8_t response=dali.receive();
    Serial.println("Respuesta recibida");
    Serial.println(response,BIN);
  }

  delay(delaytime);
  a=a+1;
  Serial.println("Contador del serial es:");
  Serial.println(a);
  comMsg = ""; // Para limpiar el buffer
};
```

9. ANEXO CÓDIGO APLICACIÓN ANDROID

```
package com.example.tfm;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.os.SystemClock;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.RadioButton;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
import android.view.View;
import androidx.appcompat.app.AppCompatActivity;
import org.achartengine.ChartFactory;
import org.achartengine.GraphicalView;
import org.achartengine.model.XYMultipleSeriesDataset;
import org.achartengine.model.XYSeries;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.lang.ref.WeakReference;
import java.util.Set;
import java.util.UUID;

public class MainActivity extends AppCompatActivity {

    private double ch0[];
    private double us[];

    private XYMultipleSeriesDataset mDataset;
    private XYMultipleSeriesRenderer mRenderer;
    private XYSeries ch0s, ch1s;
    private XYSeriesRenderer ch0sr, ch1sr;
    private GraphicalView mChartView;

    private BluetoothAdapter mBTAdapter;
    private Set<BluetoothDevice> mPairedDevices;        //Conjunto de objetos
BluetoothDevice

    private final MyHandler mHandler = new MyHandler(this);
    private ConnectedThread mConnectedThread; // hilo de transmisión y
emisión de datos por bluetooth
    private BluetoothSocket mBTSocket = null; // enlace de comunicación
bidireccional

    private final static int REQUEST_ENABLE_BT = 1; // identifica nuevos
nombres en bluetooth
    private final static int MESSAGE_READ = 2; // actualización de mensaje en
```

```
el handler
    private final static int CONNECTING_STATUS = 3; // estado del mensaje en
el handler
    private int order = 0;
    private int level = 0;

    private static final String TAG = "debugging bluetooth";
    private static final UUID BTMODULEUUID = UUID.fromString("00001101-0000-
1000-8000-00805F9B34FB"); // identificador aleatorio

    private TextView rms0, rms1, factorpotencia, potencia;
    private RadioButton Rb1, Rb2, Rb3, Rb4, Rb5;
    private SeekBar seekBar;
    private Button B1,B2;
    private LinearLayout layout;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ch0 = new double[100];
        us = new double[100];

        Rb1 = (RadioButton) findViewById(R.id.rb1);
        Rb2 = (RadioButton) findViewById(R.id.rb2);
        Rb3 = (RadioButton) findViewById(R.id.rb3);
        Rb4 = (RadioButton) findViewById(R.id.rb4);
        Rb5 = (RadioButton) findViewById(R.id.rb5);
        B1 = (Button) findViewById(R.id.button);
        B2 = (Button) findViewById(R.id.button2);
        seekBar = (SeekBar) findViewById(R.id.seekBar);

        seekBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
                                          boolean fromUser) {
                if (order == 0) {
                    order = 8;
                    level = progress;
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });

        //Se ponen a cero los textview de rms
        rms0 = (TextView) findViewById(R.id.Vv);
        rms0.setText("0");
        rms1 = (TextView) findViewById(R.id.Iv);
        rms1.setText("0");
        potencia = (TextView) findViewById(R.id.Pv);
```

```
        potencia.setText("0");
        factorpotencia = (TextView) findViewById(R.id.Fdpv);
        factorpotencia.setText("0");

        //Se genera la interfaz gráfica dentro del LinearLayout
        mDataset = new XYMultipleSeriesDataset();
        mRenderer = new XYMultipleSeriesRenderer();
        ch0s = new XYSeries("Tensión (V)" +
                "
");
        mDataset.addSeries(ch0s);
        ch0sr = new XYSeriesRenderer();
        mRenderer.addSeriesRenderer(ch0sr);

        ch1s = new XYSeries("Corriente (mA)" +
                "
");
        mDataset.addSeries(ch1s);
        ch1sr = new XYSeriesRenderer();
        mRenderer.addSeriesRenderer(ch1sr);

        layout = (LinearLayout) findViewById(R.id.uilayout);
        mChartView = ChartFactory.getLineChartView(this, mDataset,
mRenderer);
        layout.addView(mChartView, new
LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,LinearLayout
.LayoutParams.WRAP_CONTENT));

        for(int i = 0; i< 100; i++){
            ch0s.add(us[i],ch0[i]);
            ch1s.add(us[i],ch0[i]);
        }
        mRenderer.setMargins(new int[]{40, 60, 10, 20});
        mRenderer.setMarginsColor(Color.rgb(236,236,236));
        mRenderer.setAxesColor(Color.BLACK);
        mRenderer.setXLabelsColor(Color.BLACK);
        mRenderer.setYLabelsColor(0, Color.BLACK);
        mRenderer.setYLabelsAlign(Paint.Align.RIGHT, 0);
        mRenderer.setLabelsTextSize(20);
        mRenderer.setYLabelsPadding(10);
        mRenderer.setXTitle("Tiempo (ms)");
        mRenderer.setAxisTitleTextSize(20);
        mRenderer.setLabelsColor(Color.BLACK);
        mRenderer.setYAxisMax(850.0);
        mRenderer.setYAxisMin(-850.0);
        mRenderer.setXAxisMax(20.0);
        mRenderer.setXAxisMin(0.0);
        mRenderer.setGridColor(Color.DKGRAY);
        mRenderer.setShowGrid(true);
        mRenderer.setLegendTextSize(20);
        ch0sr.setColor(Color.BLUE);
        ch1sr.setColor(Color.RED);
        mChartView.repaint();

        //Gestión del bluetooth, identificar enlaces creados y establecer
conexión
        mBTAdapter = BluetoothAdapter.getDefaultAdapter();
        mPairedDevices = mBTAdapter.getBondedDevices();
        //Hay dos variables imiedevices y imiedevice
```

```
        int imiedevices = 0;
        BluetoothDevice imiedevice=null;
        for (BluetoothDevice device : mPairedDevices) {
            if(device.getName().contains("IMIE")){
                Toast.makeText(getApplicationContext(), device.getAddress(),
Toast.LENGTH_SHORT).show();
                imiedevices++;
                imiedevice = device;
            }
            SystemClock.sleep(500);
        }
        if(imiedevices ==1){
            Toast.makeText(getApplicationContext(), "Connecting to " +
imiedevice.getName(), Toast.LENGTH_SHORT).show();
            final String address = imiedevice.getAddress();
            final String name = imiedevice.getName();
            new Thread()
            {
                public void run() {
                    boolean fail = false;
                    BluetoothDevice device =
mBTAdapter.getRemoteDevice(address);
                    try {
                        mBTSocket = createBluetoothSocket(device);
                    } catch (IOException e) {
                        fail = true;
                        Toast.makeText(getBaseContext(), "Socket creation
failed", Toast.LENGTH_SHORT).show();
                    }
                    // Establece el enlace bluetooth mBTSocket
                    try {
                        mBTSocket.connect();
                    } catch (IOException e) {
                        try {
                            fail = true;
                            mBTSocket.close();
                            mHandler.obtainMessage(CONNECTING_STATUS, -1, -1)
                                    .sendToTarget();
                        } catch (IOException e2) {
                            Toast.makeText(getBaseContext(), "Socket creation
failed", Toast.LENGTH_SHORT).show();
                        }
                    }
                    if(!fail) {
                        mConnectedThread = new ConnectedThread(mBTSocket);
                        mConnectedThread.start();
                        mHandler.obtainMessage(CONNECTING_STATUS, 1, -1,
name)
                                .sendToTarget();
                    }
                }
            }.start();
        }
        else if(imiedevices ==0){
            Toast.makeText(getApplicationContext(), "No instrument paired",
Toast.LENGTH_SHORT).show();
        }
        else{
            Toast.makeText(getApplicationContext(), "More than one instrument
```

```
paired", Toast.LENGTH_SHORT).show();
        }
    }

    private class MyHandler extends Handler {
        private final WeakReference<MainActivity> mActivity;

        public MyHandler(MainActivity activity) {
            mActivity = new WeakReference<MainActivity>(activity);
        }

        @Override
        public void handleMessage(Message msg) {
            MainActivity activity = mActivity.get();
            if (activity != null) {
                if(msg.what == MESSAGE_READ){
                    String readMessage = null;
                    try {
                        readMessage = new String((byte[])
msg.obj,0,msg.arg1,"UTF-8");
                    } catch (UnsupportedEncodingException e) {
                        e.printStackTrace();
                    }
                    //Busca el primer espacio o salto de línea en el mensaje
                    int last = readMessage.indexOf("\r\n");
                    String header = readMessage.substring(0,last);
                    String remains = readMessage.substring(last+2);     //+2:
salto de línea + ; al inicio de la línea

                    double ch0rms, ch1rms, ch0a, ch1a, ch0f, ch1f;
                    double pot, fdp;
                    double xvalue, yvalue,  yvalue2;
                    pot=fdp=0;
                    ch0rms=ch1rms=ch0a=ch1a=0;

                    if(header.equals("sampling finished")){
                        ch0s.clear();
                        ch1s.clear();
                        boolean scontinue = false;  //Se usa para comprobar
si queda información pendiente
                        int samples = 1;

                        do{
                            int nextlf = remains.indexOf("\r\n");
                            String line = remains.substring(0,nextlf);
                            remains = remains.substring(nextlf+2);
                            scontinue = (remains.indexOf("\r\n")>0);
                            line = line.substring(1);  //elimina el ; del
inicio de línea
                            int nextsc = line.indexOf(";");
                            //Transforma a double el substring de line
                            double timems =
Double.valueOf(line.substring(0,nextsc))/1000.0;
                            //Se divide entre 1000 para pasar de us a ms
                            line = line.substring(nextsc+1);
                            nextsc = line.indexOf(";");
                            double ch0m =
Double.valueOf(line.substring(0,nextsc));
                            line = line.substring(nextsc+1);
```

```
                            nextsc = line.indexOf(";");
                            double ch1m =
Double.valueOf(line.substring(0,nextsc));
                            line = line.substring(nextsc+1);
                            nextsc = line.indexOf(";");
                            //Se obtienen los valores de tensión leídos
                            ch0f = ch0m*3.3/4095;
                            ch1f = ch1m*3.3/4095;
                            //Se realiza la conversión a los valores
equivalentes del circuito
                            ch0f = ch0f * 214.20;  //Tensión del circuito en
V
                            ch1f = ch1f * 1000/2; //Corriente primario en mA
                            ch0s.add(timems,ch0f);
                            ch1s.add(timems,ch1f);
                            ch0a+=ch0f;
                            ch1a+=ch1f;
                        }while(scontinue);

                        samples = ch0s.getItemCount();      //Número de
muestras recibidas

                        //Calcula le media de los valores obtenidos en cada
canal
                        ch0a/= samples;
                        ch1a/= samples;

                        for(int i =0; i < samples ; i++){
                            //Elimina el valor medio
                            ch0rms += Math.pow((ch0s.getY(0)-ch0a),2);
                            ch1rms += Math.pow((ch1s.getY(0)-ch1a),2);
                            //El índice es 0 debido al remove
                            xvalue = ch0s.getX(0);
                            yvalue = ch0s.getY(0) - ch0a;
                            yvalue2 = ch1s.getY(0) - ch1a;
                            ch0s.remove(0);
                            ch1s.remove(0);
                            ch0s.add(xvalue, yvalue);
                            ch1s.add(xvalue, yvalue2);
                            //Se calcula el valor de potencia instantánea
                            pot = pot + (yvalue * yvalue2 / 1000);   // se
divide por 1000: corriente en mA
                        }
                        //Divide entre el número de muestras porque es
equivalente
                        //200 us entre muestras, periodo de 20000 us;
Relación 1/100
                        ch0rms = Math.sqrt(ch0rms/samples);
                        ch1rms = Math.sqrt(ch1rms/samples);

                        pot /= samples;

                        if (ch0rms != 0 && ch1rms !=0) {
                            fdp = pot / (ch0rms * ch1rms / 1000);
//corriente en mA
                        }
                        else fdp = 0;

                        //Ajustar número de decimales en la visualización
```

```
                    ch0rms = Math.round(ch0rms *1000);
                    ch0rms = ch0rms/1000;
                    ch1rms = Math.round(ch1rms * 1000);
                    ch1rms = ch1rms/1000;
                    pot = Math.round(pot * 1000);
                    pot = pot/1000;
                    fdp = Math.round(fdp * 1000);
                    fdp = fdp/1000;

                }
                mChartView.repaint();

                rms0.setText(String.valueOf(ch0rms));
                rms1.setText(String.valueOf(ch1rms));
                potencia.setText(String.valueOf(pot));
                factorpotencia.setText(String.valueOf(fdp));

                if (!(order == 0))
                {
                    switch (order){
                        case 1:     //Apagado
                            mConnectedThread.write("1");
                            order = 0;
                            break;
                        case 2:     //Fluorescente Convencional
                            mConnectedThread.write("2");
                            order = 0;
                            break;
                        case 3:     //Fluorescente B.Electrónico
                            mConnectedThread.write("3");
                            order = 0;
                            break;
                        case 4:     //LED
                            mConnectedThread.write("4");
                            order = 0;
                            break;
                        case 5:     //Dali
                            mConnectedThread.write("5");
                            order = 0;
                            break;
                        case 6:     //Dali ON
                            mConnectedThread.write("6");
                            order = 0;
                            break;
                        case 7:     //Dali OFF
                            mConnectedThread.write("7");
                            order = 0;
                            break;
                        case 8:     //Seekbar
                            if (Rb3.isChecked() || Rb4.isChecked() ||
Rb5.isChecked()) {
                                mConnectedThread.write("8");

mConnectedThread.write(String.valueOf(level));
                                order = 0;
                            } else order = 0;
                            break;
                    }
                }
```

```
                }
                if(msg.what == CONNECTING_STATUS){
                    if(msg.arg1 == 1);
                    else ;
                }
            }
        }
    }

    private BluetoothSocket createBluetoothSocket(BluetoothDevice device)
throws IOException {
        return  device.createRfcommSocketToServiceRecord(BTMODULEUUID);
        //Crea una conexión saliente con el bluetooh
    }

    public void onRadioButtonClicked(View view) {
        switch(view.getId()) {
            case R.id.rb1:
                if (order == 0){
                    order = 1;
                }
                    B1.setVisibility(View.INVISIBLE);
                    B2.setVisibility(View.INVISIBLE);
                    seekBar.setVisibility(View.INVISIBLE);
                    layout.setVisibility((View.VISIBLE));
                    rms0.setVisibility(View.VISIBLE);
                    rms1.setVisibility(View.VISIBLE);
                    potencia.setVisibility(View.INVISIBLE);
                    factorpotencia.setVisibility(View.INVISIBLE);
                    break;
            case R.id.rb2:
                if (order == 0) {
                    order = 2;
                }
                    B1.setVisibility(View.INVISIBLE);
                    B2.setVisibility(View.INVISIBLE);
                    seekBar.setVisibility(View.INVISIBLE);
                    layout.setVisibility((View.VISIBLE));
                    rms0.setVisibility(View.VISIBLE);
                    rms1.setVisibility(View.VISIBLE);
                    potencia.setVisibility(View.VISIBLE);
                    factorpotencia.setVisibility(View.VISIBLE);
                    break;
            case R.id.rb3:
                if (order == 0){
                    order = 3;
                }
                    B1.setVisibility(View.INVISIBLE);
                    B2.setVisibility(View.INVISIBLE);
                    seekBar.setVisibility(View.VISIBLE);
                    layout.setVisibility((View.VISIBLE));
                    rms0.setVisibility(View.VISIBLE);
                    rms1.setVisibility(View.VISIBLE);
                    potencia.setVisibility(View.VISIBLE);
                    factorpotencia.setVisibility(View.VISIBLE);
                    break;
            case R.id.rb4:
                if (order == 0){
                    order = 4;
```

```
            }
                B1.setVisibility(View.INVISIBLE);
                B2.setVisibility(View.INVISIBLE);
                seekBar.setVisibility(View.VISIBLE);
                layout.setVisibility((View.VISIBLE));
                rms0.setVisibility(View.VISIBLE);
                rms1.setVisibility(View.VISIBLE);
                potencia.setVisibility(View.VISIBLE);
                factorpotencia.setVisibility(View.VISIBLE);
                break;
        case R.id.rb5:
            if (order == 0){
                order = 5;
            }
                B1.setVisibility(View.VISIBLE);
                B2.setVisibility(View.VISIBLE);
                seekBar.setVisibility(View.VISIBLE);
                layout.setVisibility((View.INVISIBLE));
                rms0.setVisibility(View.INVISIBLE);
                rms1.setVisibility(View.INVISIBLE);
                potencia.setVisibility(View.INVISIBLE);
                factorpotencia.setVisibility(View.INVISIBLE);
                break;
    }
}

public void Encender(View view){
    if (Rb5.isChecked()){
        if (order == 0){
            order = 6;
        }
    } else {
    }
}

public void Apagar(View view){
    if (Rb5.isChecked()){
        if (order == 0){
            order = 7;
        }
    } else {
    }
}

//Gestión de la entrada y salida de datos por el bluetooth
//Método para la comunicación por bluetooth, recepción de datos
private class ConnectedThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

    public ConnectedThread(BluetoothSocket socket) {
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;
        // Cadenas de entrada y salida con objetos temporales
        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
```

```
            } catch (IOException e) { }
            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }

        public void run() {
            byte[] buffer = new byte[4000];  // buffer para datos de entrada
            int bytes; // bytes obtenidos de read()
            // Escucha continua salvo que suceda interrupción
            while (true) {
                try {
                    // Lectura del input mmInStream
                    bytes = mmInStream.available();
                    if(bytes != 0) {
                        SystemClock.sleep(500);                 //Pausa para
recibir datos
                        bytes = mmInStream.available();             // Número
de bytes disponibles oara lectura
                        bytes = (bytes <4001)? bytes : 4000;
//Operador ternario
                        bytes = mmInStream.read(buffer, 0, bytes);
                        mHandler.obtainMessage(MESSAGE_READ, bytes, -1,
buffer)
                                .sendToTarget(); // Envía los datos a la UI
Activity
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                    break;
                }
            }
        }

        //Método para enviar datos por enlace bluetooth a través de
ConnectedThread
        public void write(String input) {
            byte[] bytes = input.getBytes();           //convierte string a
byte
            try {
                mmOutStream.write(bytes);
            } catch (IOException e) { }
        }

        //Cierre enlace bluetooth
        public void cancel() {
            try {
                mmSocket.close();
            } catch (IOException e) { }
        }
    }

}
```