

ESCUELA DE INGENIERÍAS
INDUSTRIALES

Máster en Electrónica Industrial y Automática

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

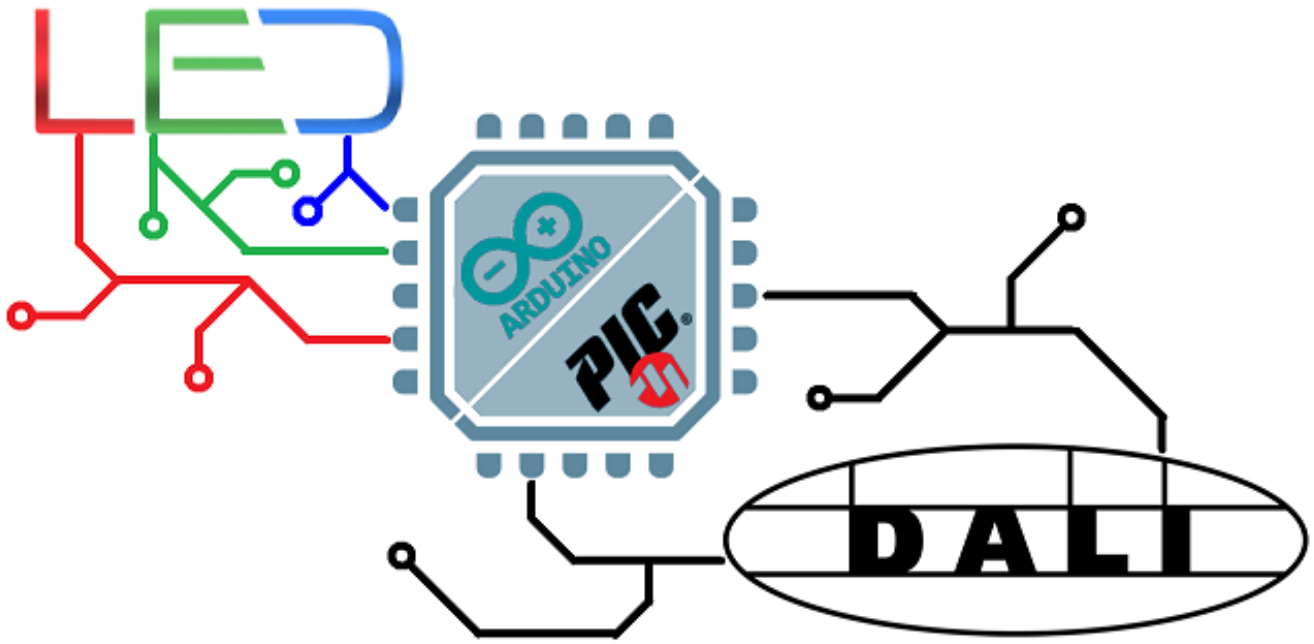
DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA

DESARROLLO DE UNA PLACA ELECTRÓNICA COMPATIBLE CON DALI PARA EL CONTROL DE LUMINARIAS

AUTOR: Alberto Encinas Elvira

DIRECTORA: María Isabel del Valle González

Valladolid, abril 2018



Resumen

Resumen

En el presente trabajo se va a desarrollar un sistema que permite el control de lámparas LED mediante un controlador DALI y crear diferentes escenas, tanto estáticas como dinámicas variando tanto el color como la intensidad de la luz. El desarrollo tendrá en cuenta la modularidad del sistema y la compatibilidad con diferentes microcontroladores de bajo coste como los ATmega utilizados en las placas de desarrollo Arduino o microcontroladores PIC, tanto discretos como montados en zócalos comerciales o de desarrollo propio. Además, se incluirán zócalos para la conexión de sensores compatibles que permitan aumentar la funcionalidad como sensores de iluminancia, memorias EEPROM, lectores de tarjetas micro SD, sensores del color de la luz y un módulo bluetooth que aumente la conectividad del sistema, permitiendo conectarse con un teléfono móvil, Tablet u ordenador para configurar el sistema de forma remota.

Palabras clave: Sistemas embebidos, microcontroladores, DALI, luminarias, comunicación.

Abstract

In this paper a DALI compatible controller device will be developed. The embedded system will allow the user to create and customize several light scenes by changing the light color and intensity. I keep in mind the modularity of the device, as well as the compatibility with some low-cost microcontrollers such as the ATmega family, used in the development Arduino boards or discrete PICs and socket mounted PICs. In addition, some sockets will be included in the printed circuit board to connect several sensors in order to increase the functionality of the system. The sensors used are illuminance sensor, EEPROM memory, microSD card reader, light color sensor and a Bluetooth module to increase the connectivity. With this module, you can connect the system through a smartphone, tablet or laptop to remotely configure it.

Keywords: Embedded systems, microcontrollers, DALI, light, communications.

Índice

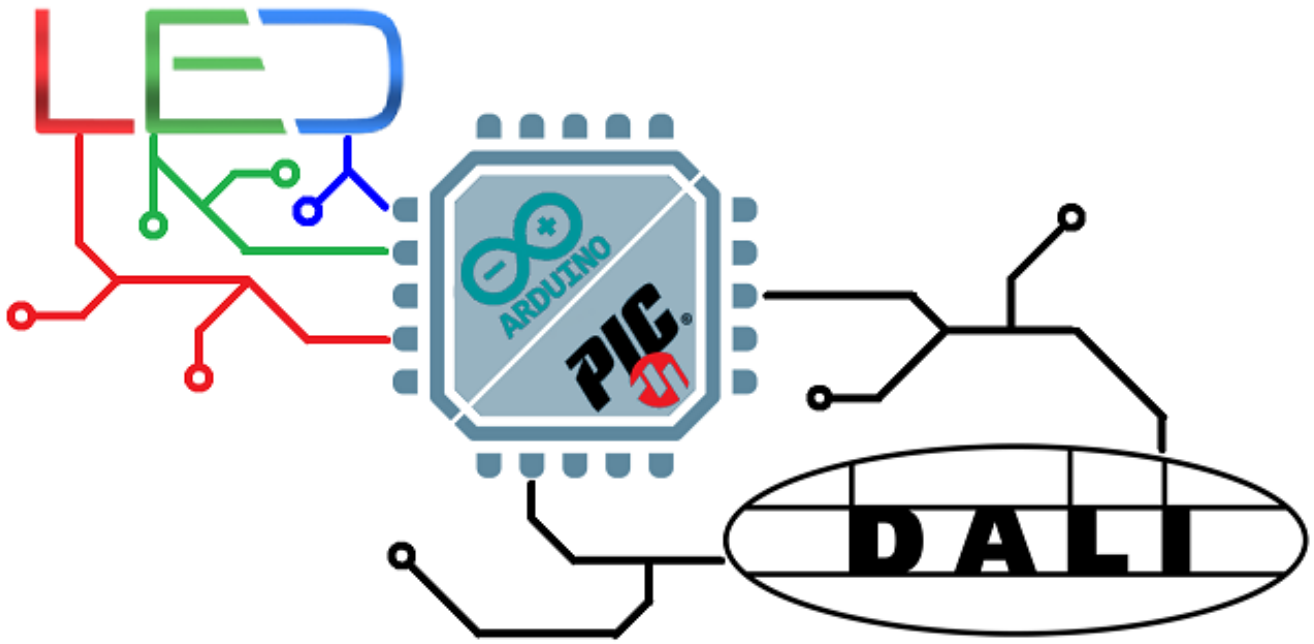
Resumen	4
Abstract	4
Índice de ilustraciones	7
1. Objetivos y Justificación	10
1.1 Antecedentes	12
1.2 Objetivos	12
1.3 Planteamiento	13
2. Estado del arte	16
2.1 Evolución de la iluminación	18
2.2 Sistemas electrónicos embebidos	21
2.3 Protocolo DALI	22
2.3.1 Descripción	22
2.3.2 Ventajas y aplicaciones	23
2.3.3 Características	24
2.3.4 Instrucciones	26
3. Desarrollo Hardware	28
3.1 Dispositivos hardware empleados	30
3.1.1 Sensor de luz	30
3.1.2 Sensor de color	31
3.1.3 Memoria EEPROM	34
3.1.4 Conexión al bus DALI	35
3.1.5 Lector de tarjetas micro SD	35
3.1.6 Módulo Bluetooth HC05	36
3.1.7 Arduino Nano	37
3.2 Diseño de la PCB	37
3.3 Implementación con DALI y leds RGB genéricos	44
3.4 Diseño de la envolvente	47
4. Desarrollo Software	54
4.1 Aplicación de configuración para Android	56
4.2 Funcionalidades del software de prueba desarrollado	61
5. Conclusiones	72
6. Líneas Futuras	76
7. Bibliografía	80
Anexos	84
ANEXO I. Tabla de comandos DALI	86

ANEXO II. Hojas de datos de los sensores empleados	88
ANEXO III. Esquema electrónico de la placa desarrollada y layout	95
ANEXO IV. Componentes DALI externos	101
ANEXO V. Presupuesto	103
ANEXO VI. Código	105

Índice de ilustraciones

Ilustración 1 Zócalo de Mikroelektronika con un PIC 18F87K22 compatible.....	13
Ilustración 2 Zócalo vacío de Mikroelektronika para soldar cualquiera de los microcontroladores compatibles	14
Ilustración 3 Microcontrolador Arduino Nano	14
Ilustración 4 Modelo de color CMYK.....	20
Ilustración 5 Modelo de color RGB	20
Ilustración 6 Círculo de matiz del modelo HSV	21
Ilustración 7 Cono cromático del modelo HSV.....	22
Ilustración 8 Incrementos de iluminación logarítmicos del protocolo DALI	23
Ilustración 9 Configuración de conexionado utilizando el protocolo DALI	24
Ilustración 10 Cableado para utilizar DALI	25
Ilustración 11 Rango de niveles de tensión para el bus DALI.....	26
Ilustración 12 Módulo Illuminance click.....	30
Ilustración 13 Tabla de registros del sensor TSL2561	31
Ilustración 14 Diagrama de funcionamiento del sensor de luz.....	31
Ilustración 15 Módulo color click	32
Ilustración 16 Diagrama de bloques del sensor de color TCS3471	32
Ilustración 17 Tabla de direcciones de los registros	33
Ilustración 18 Instrucción COMMAND necesaria para solicitar datos al sensor.....	34
Ilustración 19 Módulo EEPROM 3 click	35
Ilustración 20 Módulo DALI click que permite la conexión al bus DALI	35
Ilustración 21 Módulo genérico microSD.....	36
Ilustración 22 Módulo lector de tarjetas microSD de mikroelektronika.....	36
Ilustración 23 Módulo bluetooth HC05 empleado	37
Ilustración 24 pinout de los zócalos de MikroBus.....	38
Ilustración 25 Tamaños estandarizados de zócalos MikroBus.....	39
Ilustración 26 Zócalo universal compatible con Arduino y los PIC mencionados	40
Ilustración 27 Diseño final de la PCB desarrollada.....	42
Ilustración 28 Modelo 3D de la placa de circuito impreso desarrollada.....	43
Ilustración 29 Comparativa de tamaño entre la antigua placa y la nueva.....	44
Ilustración 30 Jumper de selección para el modo DALI o LED_RGB.....	45
Ilustración 31 Conector para LEDs RGB.....	46
Ilustración 32 Escena de luz blanca 4000K en la luminaria DALI	46
Ilustración 33 Escena dinámica Rojo-Azul en luminaria DALI y led RGB.....	47
Ilustración 34 Parte inferior de la envolvente diseñada	48
Ilustración 35 Parte superior de la envolvente	49
Ilustración 36 Trampilla conectores LED y bus DALI	50
Ilustración 37 Trampilla conector USB Arduino y puerto ICSP del PIC.....	50

Ilustración 38 Funcionamiento de la bisagra diseñada.....	51
Ilustración 39 Vista en corte de la trampilla cerrada y el mecanismo oculto.....	52
Ilustración 40 Resultado final de la envolvente.....	52
Ilustración 41 Relación entre los valores RGB y la temperatura de color obtenida.....	57
Ilustración 42 Diagrama de funcionamiento de la Aplicación para Android.....	58
Ilustración 43 icono de la aplicación desarrollada.....	59
Ilustración 44 Pantalla principal de la aplicación.....	59
Ilustración 45 Pantalla para configurar una escena.....	60
Ilustración 46 Pantalla de configuración de escenas.....	61
Ilustración 47 Diagrama básico de funcionamiento del programa.....	63
Ilustración 48 muestra del registro de funcionamiento.....	67
Ilustración 49 Aspecto de los datos en bruto generados por el sistema en el fichero AMBIENTE.LOG.....	68
Ilustración 50 Gráfico que representa la iluminancia y el color de la luz en cada momento.....	69



I. Objetivos y Justificación

1.1 Antecedentes

El presente TFM se desarrolla como evolución al trabajo titulado “CONTROL DE INTENSIDAD Y COLOR DE LUMINARIAS LED PARA LA REALIZACIÓN DE TERAPIAS” realizado por D. Jorge Gallud Cano. En dicho trabajo, su autor desarrolló un software de control para luminarias DALI, implementado en el microcontrolador PIC 18F87K22 y empleando una tarjeta de desarrollo Easy PIC Pro v7.

En el trabajo indicado se consiguió realizar la comunicación con las luminarias DALI y asignar las direcciones a los diferentes colores de la luminaria de pruebas, por lo que en este trabajo se mantendrán las direcciones establecidas para realizar el control de la luminaria.

También se implementó en lenguaje de programación C las funciones necesarias para enviar y recibir información del bus DALI, cumpliendo con la codificación Manhattan. Se aprovechará también en la medida de lo posible este código para implementar las funciones a realizar en este trabajo.

La finalidad sería desarrollar un sistema que tenga al menos las mismas capacidades que el actual, reduciendo en lo posible el tamaño de la placa y los requisitos del microcontrolador.

1.2 Objetivos

El presente trabajo fin de máster se plantea como una inmersión en el ámbito de la iluminación y el desarrollo de sistemas embebidos basados en microcontroladores. Se pretende desarrollar un sistema embebido modular y lo más universal posible, que permita la conexión de un microcontrolador a un bus DALI para realizar el control de la iluminación aprovechando este protocolo y a su vez, poder conectar a dicha placa otros elementos que puedan agregar valor al producto y añadir características interesantes de forma sencilla, económica y versátil como son las siguientes:

- Comunicación con sensores relacionados a través de un bus I2C
- Recogida de datos de estos sensores para poder ser almacenados, tratados y analizados con el fin de extraer conclusiones acerca de la situación de iluminación ambiente.
- Establecer una comunicación bluetooth con dispositivos móviles de forma que se pueda modificar el color o intensidad de la luz y aplicar escenas de forma inalámbrica.
- Realizar un sistema compatible con unos de los microcontroladores más extendidos actualmente como son los PIC y los ATmega.
- Permitir controlar luminarias LED RGB tradicionales sin necesidad de implementar un bus DALI.
- Desarrollo de una aplicación sencilla para dispositivos Android que permita el control de las luminarias vía bluetooth.
- Diseño de una envolvente para contener el sistema electrónico desarrollado.
- Migración del software desarrollado en el trabajo anterior para microcontroladores PIC a la plataforma Arduino.

Para llevar a cabo este trabajo, se realizará un estudio previo del protocolo DALI, así como de los sensores relacionados a la iluminación, como sensores del color de la luz o de la iluminancia, que proporcionarán una realimentación de lo que está sucediendo en la estancia. También se estudiarán otros sensores que, aunque no están directamente ligados al control de la iluminación, permiten aumentar la funcionalidad del diseño como pueden ser módulos bluetooth para aumentar la conectividad del controlador o módulos de memoria que permitan

almacenar escenas de iluminación, perfiles de usuario o registros del color de la iluminación y su intensidad en cada momento.

1.3 Planteamiento

En el presente proyecto se ha planteado el desarrollo de un sistema embebido, compatible con los microcontroladores PIC y ATmega, que contenga los conectores necesarios para poder utilizar un microcontrolador y algunos elementos auxiliares, como son un módulo bluetooth para aumentar la conectividad de forma inalámbrica y permitir conexión con otros dispositivos como pueden ser smartphones.

El desarrollo principal será la placa base donde se conectarán el resto de módulos. Se hará adaptada para la conexión de los módulos mikroBus de la empresa Mikroelektronika. Éstos están teniendo gran aceptación entre algunos fabricantes de microcontroladores como Microchip, que ya ofrece incluso placas de desarrollo compatibles con dichos módulos. Además, al ser módulos bastante estándares que utilizan para su conexión dos tiras de pines de distancia 0,1 pulgada entre dos pines consecutivos, resultaría sencillo en un futuro desarrollar otros módulos que encajen en la placa base implementada.

Por otro lado, disponemos de opciones interesantes a la hora de utilizar microcontroladores, como son microcontroladores discretos o módulos que integren un microcontrolador. Dentro de esta última clasificación encontramos placas como las Easy PIC Pro MCU Cards de Mikroelektronika, que son zócalos que incluyen prácticamente el microcontrolador en bruto soldado a una tarjeta de expansión de pines para facilitar su conexionado, o placas como Arduino, que además del microcontrolador incluyen una serie de componentes que facilitan la alimentación y la comunicación con él sin necesidad de programadores específicos.

Para el desarrollo del presente TFM se ha decidido realizar una placa de circuito impreso que tenga un zócalo lo más universal posible en el que se pueda conectar tanto las placas Easy Pic Pro MCU Cards de Mikroelektronika, como la placa de desarrollo Arduino Nano. Además, hace posible la implementación de zócalos personalizados, siempre que se cumplan con los factores de forma de ambas plataformas.

Al realizar este diseño, se obtiene compatibilidad, por tanto con el microcontrolador ATmega 328P, que es el utilizado por las placas Arduino Nano y los microcontroladores PIC18F8527-PT, PIC18F8622-PT, PIC18F8627-PT, PIC18F8722-PT, PIC18F8390-PT, PIC18F8490-PT, PIC18F8520-PT, PIC18F8620-PT, PIC18F8310-PT, PIC18F8410-PT, PIC18F8585-PT y el PIC18F8680-PT, entre otros, que pueden ser montados directamente en los zócalos de Mikroelektronika de 104 pines.



Ilustración 1 Zócalo de Mikroelektronika con un PIC 18F87K22 compatible

Además de poder adquirir los zócalos con un microcontrolador ya soldado, se pueden adquirir únicamente los zócalos, sin microcontrolador lo que nos permite utilizar cualquiera de los microcontroladores compatibles mencionados.

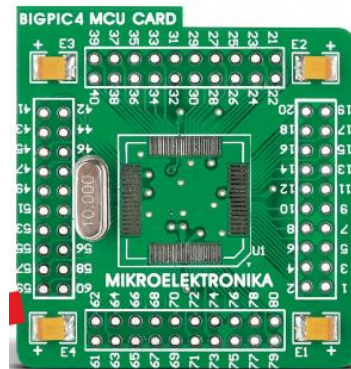
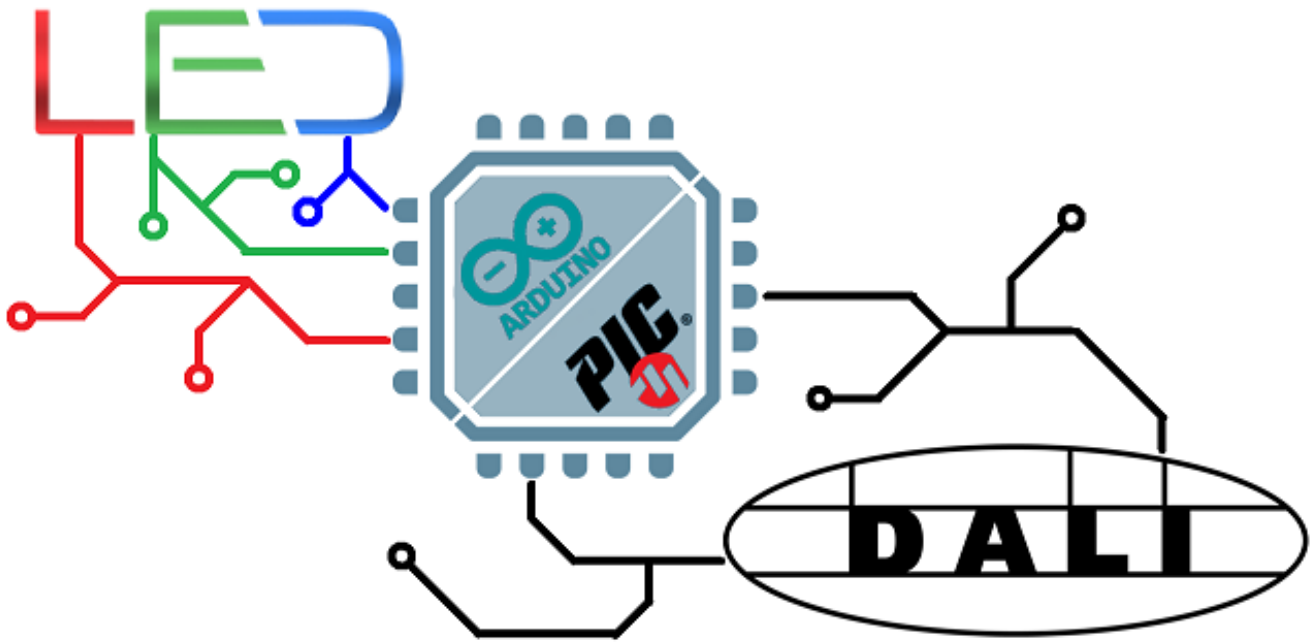


Ilustración 2 Zócalo vacío de Mikroelektronika para soldar cualquiera de los microcontroladores compatibles

Esto nos permite utilizar microcontroladores de bajo coste, como los PIC 18FXXXX y los ATmega, incluso las placas de desarrollo Arduino, para realizar un sistema básico de control de la iluminación y aumentar las características cuando sea necesario incorporando un microcontrolador superior, como los modelos de PIC mencionados. Esto junto a la memoria EEPROM externa, que permite tener almacenadas las escenas personalizadas o perfiles de usuario, hace que a la hora de cambiar el microcontrolador no sea necesario volver a configurar el sistema de nuevo.



Ilustración 3 Microcontrolador Arduino Nano



2. Estado del arte

2.1 Evolución de la iluminación

En el origen, el Sol era la única fuente de luz que alumbraba la Tierra. Todos los seres vivos, incluidos los humanos más primitivos, sincronizaban su actividad con la salida y la puesta de este astro. Cuando caía la noche, no quedaba ningún resplandor sobre la superficie terrestre, lo que obligaba a los seres vivos diurnos a refugiarse y esperar la salida del Sol al día siguiente. Posteriormente, el hombre descubre el fuego, fuente de luz y calor, lo que les permitía poder ver en la oscuridad de la noche, y alargar por tanto su actividad diaria.

Pasaron más de 700.000 años hasta que el ser humano desarrolló una fuente de luz artificial, la bombilla incandescente, inventada en la segunda mitad del siglo XIX. Hasta aquí, las fuentes de luz se caracterizaban por ser constantes, tanto en intensidad como en tonalidad de color, lo que se conoce con el nombre de temperatura de color. La temperatura de color se mide en grados Kelvin, en un rango que oscila entre los 2500K y 8000K. Cuanto menor es la temperatura del color, mayor es el tono amarillento de la luz y cuanto mayor es la temperatura del color, más blanca es la luz.

La temperatura del color de una llama de una vela se estima en torno a 1700K-2000K, lo que indica la alta tonalidad amarillenta. La luz de una bombilla incandescente tiene una temperatura de color de entre 2500K-3000K. Palidece algo el tono amarillento comparado con el fuego, pero sigue teniendo una fuerte tonalidad a este color, ya que lo que provoca la iluminación no es más que un metal incandescente.

Hasta el siglo XX, las personas nos hemos alumbrado con este tipo de luz, de tonalidad amarillenta, similar a la del fuego, que provoca unas reacciones en el cuerpo humano que describiremos más adelante, cuando se analice el impacto de la temperatura del color sobre los humanos.

No será hasta el siglo XXI, cuando se globalice una fuente de iluminación que revolucionará el mundo, consiguiendo algo que había sido impensable hasta el momento: cambiar el color de la luz con una única bombilla. Ha llegado la bombilla LED. Las características que presentan estas bombillas son un reducido consumo, comparándolas con las lámparas incandescentes, que supone un ahorro superior al 90% en determinados casos y la posibilidad de integrar diodos RGB, capaces de generar cualquier tonalidad de color combinando luz verde, roja y azul. En este momento toma más relevancia la temperatura del color ya que se podrá variar, tanto el color como la intensidad de la luz para adaptarla a cada momento del día.

Influencia de la temperatura del color

Hasta ahora hemos hablado de las diferentes temperaturas del color de la luz con la que nos hemos encontrado, ahora vamos a analizar el efecto de ésta sobre las personas.

Se ha demostrado que tanto la temperatura del color como la intensidad de la luz tienen un efecto psicológico en las personas. Es fácil comprender que cuando una persona se encuentra en un lugar con poca iluminación, inconscientemente tiende a hablar más bajo que cuando, aun estando en el mismo lugar, la intensidad de la luz es mucho más elevada. Así hay estudios que demuestran que intensidades lumínicas por debajo de 180lux-150lux hacen que las personas tiendan a hablar con un tono de voz más bajo. Del mismo modo, estar en lugares con una intensidad de luz superior a los 2000 lux hace que las personas estén más activas y hablen con un tono de voz más elevado.

Si evaluamos ahora la temperatura del color, tenemos tres tipos de luces, en función de su temperatura de color: luz cálida, luz neutra y luz fría.

La luz cálida, técnicamente se corresponde con temperaturas de color inferiores a 3300K. Su luminosidad invita a la relajación y aumenta la sensación de confort de las personas que la perciben. Este estado de predisposición se puede entender como que antiguamente, la luz del atardecer, con una temperatura de color reducida y un alto tono rojizo amarillento, invitaba a las personas a abandonar la frenética actividad del día y relajarse en casa, algo que llevaba marcando el ritmo de vida de las personas durante más de 700.000 años. Por ello una luz tenue rojiza se asocia con el final de la actividad y el cuerpo se prepara para el descanso.

Por su parte, la luz neutra se corresponde con temperaturas de color de entre 3300K y 5000K. Este tipo de luz tiene menos tonalidad rojiza amarillenta que en el caso anterior y por tanto no invita tanto a la relajación. Provoca una mayor actividad en las personas que la luz cálida.

Por último, el rango de temperatura de color de luz fría se corresponde con una temperatura de color superior a los 5000K. Este tipo de luz presenta un tono mucho más blanco que en los casos anteriores, con tonalidades incluso azuladas. El aumento de la cantidad de luz azul presente en este tipo de iluminación activa las personas. Algunos de los efectos de la luz azul son insomnio y mejora de la memoria. Según ha comprobado el investigador de la universidad de Montreal (Canadá) Gilles Vandewalle, la luz azul aumenta el ritmo cardíaco y, usando un encefalograma observó que cuando se somete a una persona a este tipo de luz mientras realizaba tareas de memorización, mejoraba la respuesta de la corteza prefrontal y parietal de su cerebro.

La luz fría principalmente se utiliza para ambientes activos, es decir, en aquellos que se requiera de un gran rendimiento y concentración. Por ello, las estancias que más se ajustan a este tipo de iluminación son lugares donde se desarrollan actividades que requieren concentración, como las oficinas o bibliotecas.

Todos estos efectos se pueden controlar gracias a la presencia de las bombillas LED y los balastos DALI, que permiten modificar tanto la temperatura del color ¹como la intensidad lumínica², permitiendo modificar y adaptar estos parámetros que hasta ahora había sido muy complicado realizar de forma sencilla y con una única bombilla.

El control y la modificación de estos parámetros se realiza con un controlador conectado al bus DALI que le envía los comandos al balastro DALI y éste actúa sobre la bombilla.

El color de la luz

Existen múltiples formas de representación de los colores. Cada una de estas formas se conoce con el nombre de modelo de color. Cada uno de los modelos emplea diferentes parámetros a la hora de definir cada color. A continuación, se explicarán brevemente algunos de estos modelos.

Modelo CMYK.

Este modelo es el que se emplea en la imprenta. Define cuatro colores básicos, que son el cyan, el magenta, el amarillo y el negro. Combinando estos cuatro colores básicos en diferentes cantidades se pueden generar el resto. Lo más normal es que cada color base tenga un valor de

¹ Para poder modificar la temperatura de color o el color de la luz es necesario disponer de una luminaria con LEDs RGB.

² Siempre que la bombilla LED sea compatible con el protocolo DALI y esté controlada por un balastro DALI

0 a 100, que representa el porcentaje de aportación de dicho color. Este modelo es un tipo de modelo sustractivo, ya que los colores se aplican sobre una superficie, generalmente blanca y cada color “sustrae” brillo de la superficie.

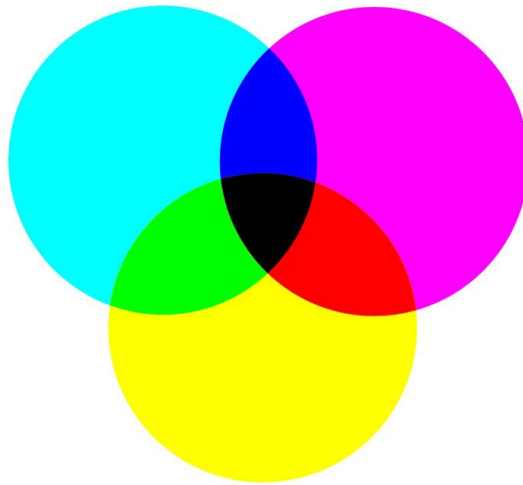


Ilustración 4 Modelo de color CMYK

Modelo RGB

Este modelo emplea únicamente tres colores básicos que son el rojo, el azul y el verde. Es el modelo que suele usarse para conseguir luz de diferentes colores. Cada componente básico puede tener un total de 255 valores diferentes, lo que conforman una profundidad de color de 24 bits (8 bits por cada color). A diferencia del modelo CMYK, este es aditivo, ya que se añade luz de un determinado color para combinarla con la existente y modificar el color apreciable. Cuando se aplica el valor máximo de cada uno de los colores básicos o primarios, el ojo humano detecta la luz resultante como blanca, por el contrario, cuando ninguno de ellos emite luz, se interpreta como el color negro (ausencia de luz).

Este es el modelo empleado por los monitores o las bombillas LED policromáticas.

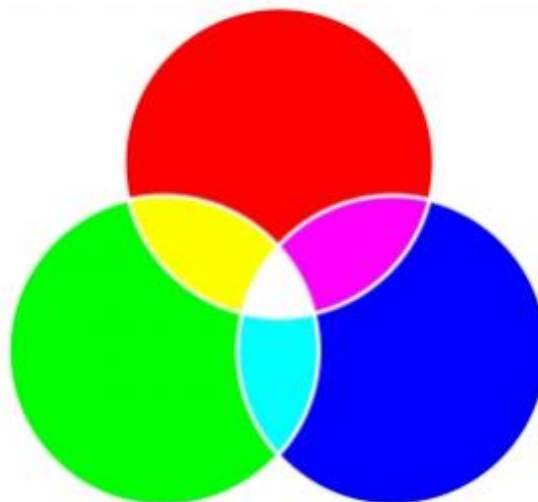


Ilustración 5 Modelo de color RGB

Modelo HSB o HSV

El modelo HSB utiliza los conceptos de matiz, saturación y brillo para definir los colores. El matiz es el parámetro que se encarga de describir el pigmento del color, es decir, el color en sí que se quiere representar. Este parámetro se expresa en grados (0°-360°) y hace referencia al color que se encuentra ubicado en esa posición en el círculo cromático. Así por ejemplo el color rojo se encuentra ubicado en esa posición en el círculo cromático. Así por ejemplo el color rojo se corresponde con los 0°, el amarillo estaría a 60° mientras que el azul estaría a unos 240°.

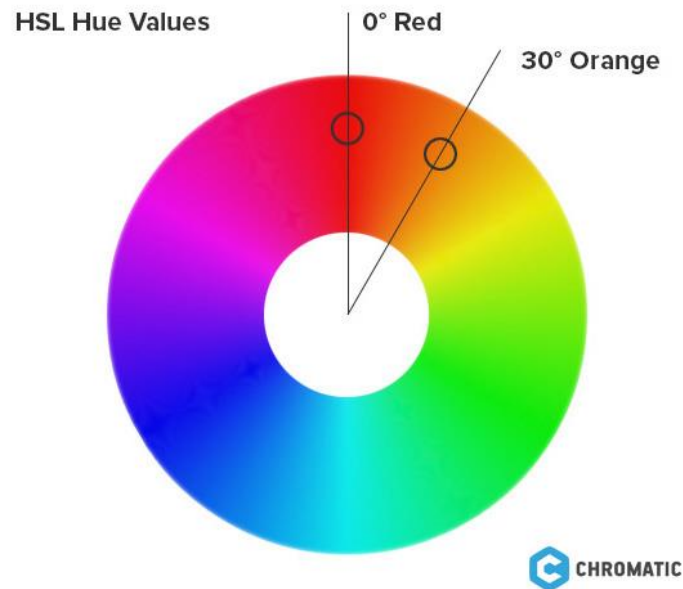


Ilustración 6 Círculo de matiz del modelo HSV

La saturación indica la intensidad del color. Esto indica si el color es muy vivo o tiene una tonalidad más tenue. Su valor va de 0 a 100, que representa el porcentaje de dicho color. Cuanto mayor sea su valor, más intenso será el color.

Finalmente, el parámetro de brillo indica la cantidad de blanco que contiene el color. Al igual que la saturación puede tomar valores entre 0 y 100, siendo 100 el correspondiente al color blanco (máximo brillo).

Para el desarrollo de este trabajo, se empleará el modelo RGB, al ser el más extendido en el ámbito de la iluminación LED y es el modelo para el que viene preparado el protocolo DALI, ya que se emplearán LEDs RGB.

2.2 Sistemas electrónicos embebidos

Un sistema embebido o sistema empotrado lo vamos a definir como un sistema electrónico diseñado específicamente para realizar unas determinadas funciones, habitualmente formando parte de un sistema de mayor entidad. La característica principal es que emplea para ello uno o varios procesadores digitales (CPUs) en formato microprocesador, microcontrolador o DSP lo que le permite aportar 'inteligencia' al sistema anfitrión al que ayuda a gobernar y del que forma parte. Otro aspecto importante de los sistemas embebidos es su capacidad para comunicarse con otros elementos, bien sea otros microcontroladores, sistemas o sensores.



Ilustración 7 Cono cromático del modelo HSV

Los sistemas embebidos se pueden encontrar actualmente en multitud de elementos a nuestro alrededor, ya que la gran mayoría de dispositivos electrónicos utilizan microcontroladores para realizar tareas. Este gran desarrollo de los sistemas electrónicos embebidos ha sido posible gracias a la producción masiva de los microcontroladores y la reducción de su precio.

También ha sido un factor determinante la aparición de plataformas de desarrollo basadas en microcontroladores como Arduino o Pingüino.

2.3 Protocolo DALI

2.3.1 Descripción

DALI (*Digital Adressable Lighning*) es un estándar para la comunicación entre balastos electrónicos que se incluye dentro del estándar IEC929. El sistema DALI surge como un avance al tradicional control analógico 1-10V. Presenta numerosas ventajas frente al sistema tradicional como una mayor flexibilidad y simplicidad de conexionado. Los campos de aplicación del protocolo van desde el control de luminarias en función de la hora del día para conseguir un ahorro energético hasta la creación de diferentes escenas para aumentar el confort en hoteles o salas de reuniones ya que permite la modificación de la intensidad de las luminarias de forma independiente.

El sistema DALI es un sistema Maestro-Esclavo, en la que el dispositivo Maestro (objeto de desarrollo en el presente TFM) se comunica con otros dispositivos esclavos (balastos) para gestionarlos tanto de forma individual como de forma grupal. Cada uno de los esclavos dispone de una dirección única lo que hace que el sistema tenga bastante inmunidad al ruido electromagnético.

Cada dispositivo maestro es capaz de controlar hasta 64 luminarias de forma individual, que pueden unirse en hasta 16 grupos diferentes si es necesario para crear diferentes zonas independientes. Esta versatilidad hace que pueda aplicarse el protocolo DALI en múltiples escenarios tanto en el mundo industrial como en el doméstico.

2.3.2 Ventajas y aplicaciones

El protocolo DALI, cuenta con una serie de características que lo hacen muy interesante y con grandes ventajas que están facilitando su implantación. A continuación, se van a indicar algunas de estas ventajas.

Posibilidad de modificar la intensidad de cada luminaria. El protocolo DALI dispone de 256 niveles de intensidad luminosa para cada luminaria. El protocolo DALI permite la gestión de estos niveles, tanto en orden ascendente como descendente en un ajuste logarítmico. Esto hace que percepción de iluminación para el ojo humano sea más lineal con cada incremento o decremento.

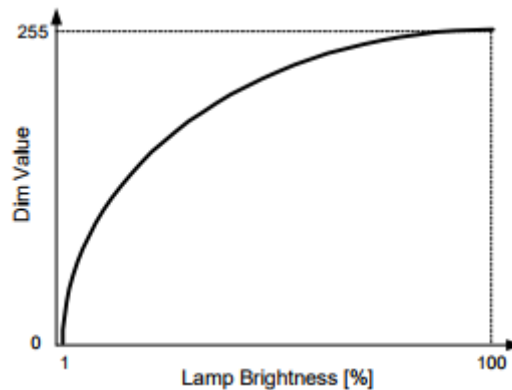


Ilustración 8 Incrementos de iluminación logarítmicos del protocolo DALI

Otra de las grandes de ventajas de utilizar este protocolo es conseguir un mayor ahorro energético. Como otros sistemas de control automático de iluminación, el uso del protocolo DALI permite emplear un microcontrolador para gestionar la iluminación de una estancia y adaptar la intensidad luminosa a las necesidades de cada momento. Además, al permitir varios niveles de intensidad luminosa presenta grandes ventajas frente a los controles ON-OFF tradicionales consiguiendo un mayor confort a la par que se ahorra energía.

Este ahorro se puede maximizar teniendo en cuenta la característica del protocolo DALI que permite direccionar de forma individual cada una de las luminarias, lo que hace posible que dentro de la misma estancia, cada una de las luminarias tenga un nivel de intensidad diferente, para que en su conjunto se mantenga la iluminación de la estancia lo más homogénea posible, independientemente de la ubicación de las fuentes de luz natural existentes como ventanas o claraboyas.

La implantación del protocolo DALI en multitud de balastos de diferentes fabricantes hace que aumente la competencia y bajen los precios en beneficio del consumidor, que podrá encontrar balastos compatibles a un precio menor. De la misma forma, aumenta la versatilidad del sistema al no estar ligado a un único tipo de balastro.

Este protocolo puede utilizarse no sólo para mejorar el confort en diferentes estancias, edificios o situaciones, si no para ayudar a corregir diferentes problemas en las personas como trastornos del sueño o mejorar el estado de ánimo. Como se ha comentado en puntos anteriores, la iluminación y los biorritmos del cuerpo humano están ligados. Gracias a la posibilidad de poder establecer una tonalidad de color de la luz determinada en cada una de las luminarias y ajustando el nivel de intensidad se pueden crear ambientes que inviten a la relajación, aliviar tensiones y preparar al cerebro para conciliar el sueño reduciendo en la medida de lo posible

problemas de insomnio o estrés. De igual forma se puede modificar la temperatura del color de la iluminación para generar iluminaciones que estimulen al cerebro y mejorar el rendimiento de los trabajadores.

Con esto vemos la gran cantidad de ventajas que se abren ante el uso de este protocolo, que no se centran sólo en mejorar la iluminación de una determinada si no que pueden aplicarse a otros campos como la luminoterapia.

2.3.3 Características

El funcionamiento del protocolo DALI se basa en el envío de comandos desde el dispositivo maestro hacia los balastos. El sistema se basa en la arquitectura maestro-esclavo, siendo el maestro un dispositivo de control y los esclavos los balastos electrónicos.

Por cada maestro puede haber hasta un máximo de 64 balastos, controlables individualmente y agrupables en hasta 16 conjuntos.

Para la conexión de los balastos únicamente se emplean dos cables por los que se envían los comandos DALI, lo que simplifica enormemente el conexionado de todas las luminarias, ya que con dos únicas líneas se pueden conectar hasta 64 balastos a un dispositivo maestro.

Además del bus de datos DALI, es necesario llevar los cables de alimentación a todos los balastos (fase, neutro y tierra), de forma que únicamente con 5 cables se puede realizar la conexión.

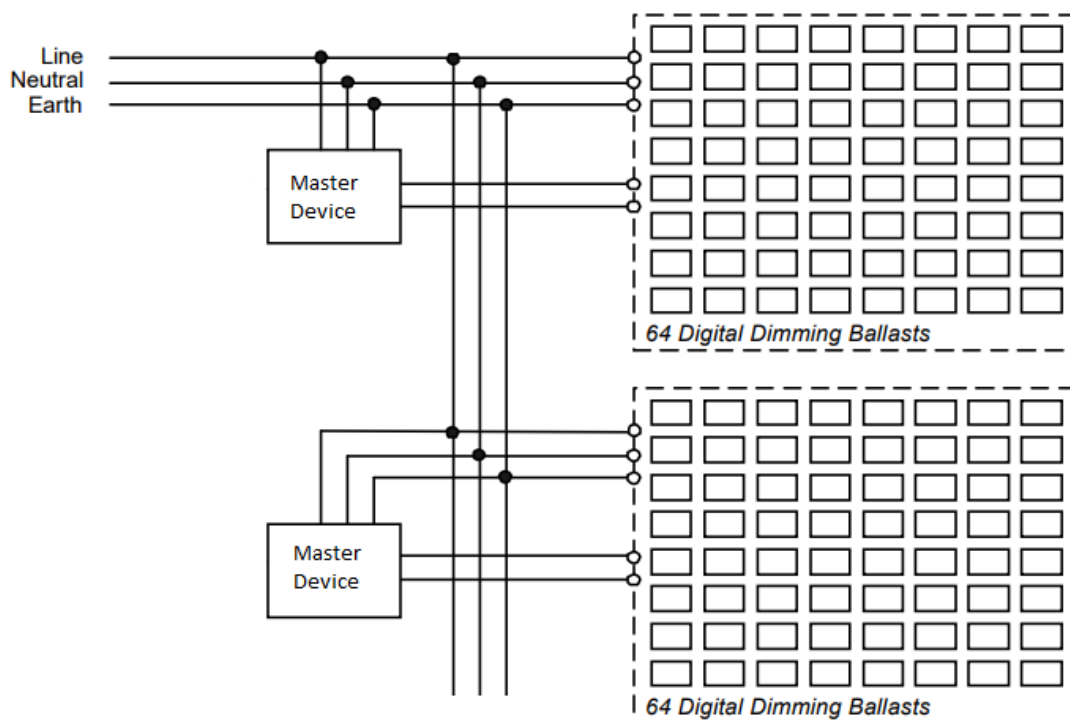


Ilustración 9 Configuración de conexionado utilizando el protocolo DALI

Además, una ventaja del cableado DALI es que el bus de datos no tiene polaridad, lo que facilita su conexión no sólo en nuevas instalaciones, si no cuando se desean añadir nuevos elementos a una red ya existente.

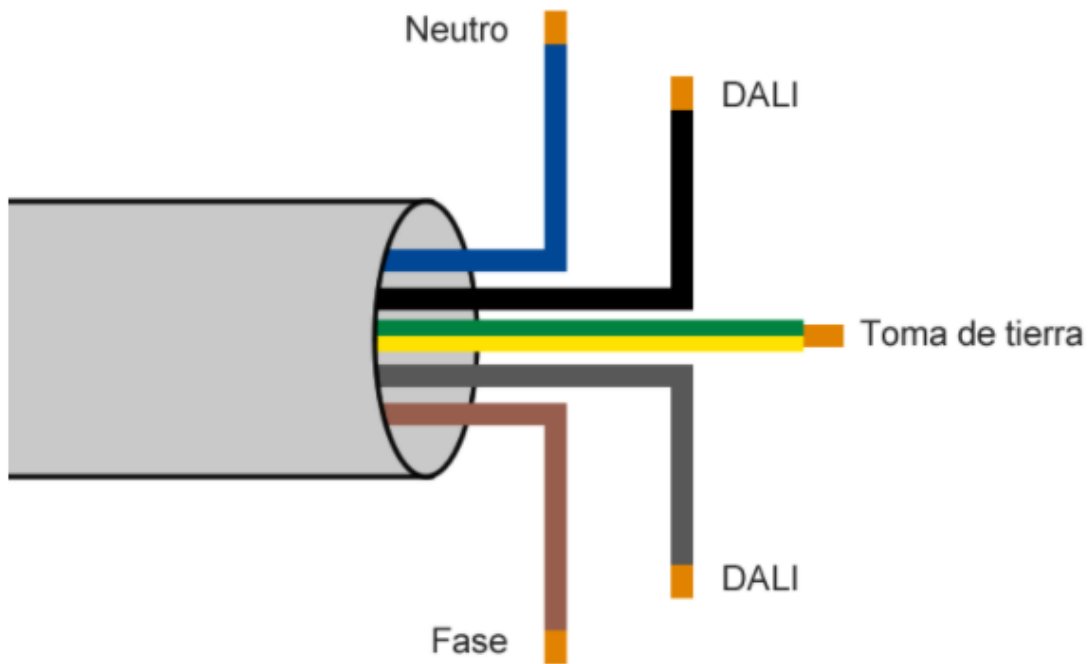


Ilustración 10 Cableado para utilizar DALI

Como restricciones en el cableado, indicar que la máxima caída de tensión en el bus DALI no puede ser mayor de 2V entre dos dispositivos cuando circula por él la máxima corriente (250mA), lo que limita la longitud del cableado. Aunque inicialmente esta longitud puede variar en función de la sección del cable elegido, no deberá superar los 300m de longitud por lo que típicamente se empleará un cable de $1,5\text{mm}^2$ de sección. No obstante, si la distancia entre dos dispositivos es muy inferior a 300m, se puede reducir la sección, siendo las más utilizadas las que se muestran en la siguiente tabla:

Longitud del cable	Sección
Hasta 100m	$0,5\text{mm}^2$
100m-150m	$0,75\text{mm}^2$
150m-300m	$1,5\text{mm}^2$

Tabla 1 Secciones de cable a emplear en función de la distancia del cableado

El protocolo DALI permite trabajar con velocidades de transmisión de hasta 1200bps, que, aunque pueda parecer reducido a priori es más que suficiente, teniendo en cuenta que las instrucciones son de pocos bits.

El rango de tensión del bus DALI debe estar comprendido entre 9,5V y 22,5V para el estado alto, y entre -6,5V y 6,5V para el estado bajo y no es necesario emplear resistencias de terminación del bus. La tensión típica de alimentación del bus es de 0V-16V.

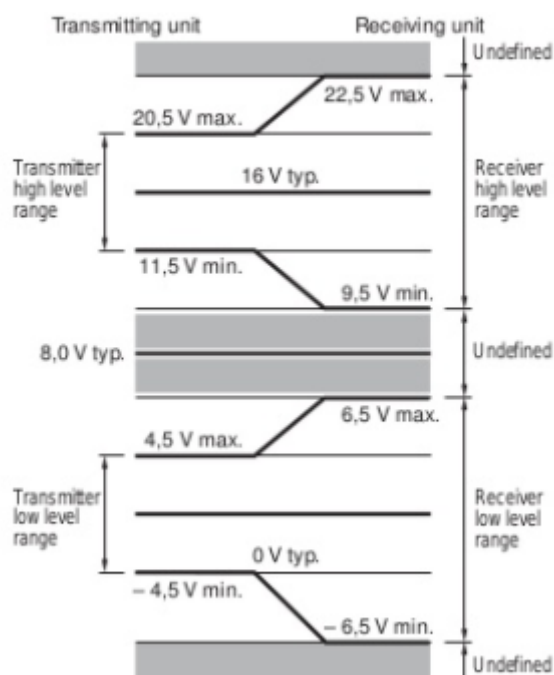


Ilustración 11 Rango de niveles de tensión para el bus DALI

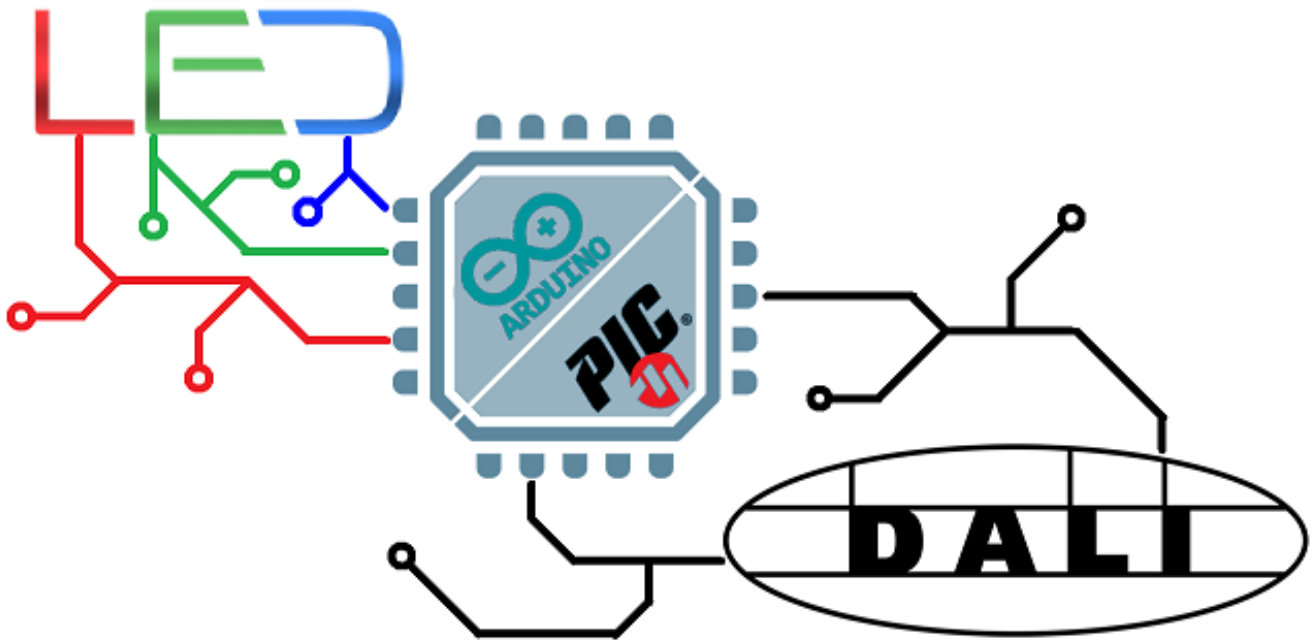
DALI se basa en el uso de direcciones para identificar y controlar cada uno de los dispositivos esclavos conectados al bus. Cada uno de ellos dispone de una dirección única, por lo que es posible enviar comandos únicamente a un dispositivo, aunque todos estén conectados al mismo sistema. Además, DALI permite realizar un broadcast con el fin de enviar información a todos los dispositivos conectados al mismo tiempo.

2.3.4 Instrucciones

Todo el funcionamiento del protocolo DALI es en base a instrucciones que el dispositivo maestro envía a los balastos para modificar su estado o realizar configuraciones. En total se pueden diferenciar 4 tipos de comandos o instrucciones.

- 1) Comandos de alimentación: Permiten establecer la alimentación del balastro.
- 2) Comandos de configuración: Permiten configurar el balastro. Este tipo de comandos deben ser enviados dos veces en menos de 100ms para que tengan efecto, en caso contrario serán ignorados.
- 3) Comandos de consulta: Con estas instrucciones el dispositivo maestro pide información a los balastos, como el nivel de tensión actual o el número de versión. Tras la emisión de un comando de este estilo el balastro correspondiente puede enviar las tramas con la información solicitada.
- 4) Comandos especiales: Se utilizan para inicializar y configurar los balastos. Estos comandos, al igual que sucede con los comandos de configuración deben ser enviados dos veces en menos de 100ms para ser aceptados y sólo serán considerados durante los primeros 15 minutos de la conexión del sistema después de haber enviado el comando "INITIALIZE".

Todas las instrucciones se encuentran detalladas en ANEXO I. Tabla de comandos DALI.



3. Desarrollo hardware

3.1 Dispositivos hardware empleados

A continuación, se mencionarán los componentes hardware que formarán parte del proyecto, explicando sus características y la funcionalidad que se le dará en el trabajo realizado.

3.1.1 Sensor de luz

El sensor de luz empleado es un TSL2561, incluido en el módulo illuminance click de Mikroelektronika.

Este sensor se encarga de captar la cantidad de luz visible e infrarroja que hay en el ambiente. Cuenta con dos receptores integrados, uno sensible a todo el espectro de la luz y otro sensible únicamente al espectro infrarrojo. La salida del sensor se proporciona a través de un puerto I2C e indica únicamente la cantidad de luz visible, aplicando fórmulas para su obtención. La resolución del sensor es de 16 bits. Este módulo funciona únicamente a una tensión de 3,3V y en el módulo se incluye un LED indicador que nos permite saber si el dispositivo está encendido.



Ilustración 12 Módulo Illuminance click

Se utilizará este sensor para determinar el nivel ambiente de iluminación y tomar medidas como aumentar o reducir el brillo de las luminarias para obtener la situación deseada.

El sensor TSL2561 que integra este módulo cuenta con la siguiente tabla de registros. En ella aparecen todos los datos que se pueden solicitar al sensor, así como la dirección en la que se almacenan. Esta dirección habrá que indicarla en el momento de realizar la petición de datos al sensor a través del bus I2C.

De todos los datos que aparecen en la tabla, nos interesarán los datos que hacen referencia a los valores del convertidor ADC de iluminancia.

ADDRESS	REGISTER NAME	REGISTER FUNCTION
---	COMMAND	Specifies register address
0h	CONTROL	Control of basic functions
1h	TIMING	Integration time/gain control
2h	THRESHLOWLOW	Low byte of low interrupt threshold
3h	THRESHLOWHIGH	High byte of low interrupt threshold
4h	THRESHHIGHLOW	Low byte of high interrupt threshold
5h	THRESHHIGHHIGH	High byte of high interrupt threshold
6h	INTERRUPT	Interrupt control
7h	---	Reserved
8h	CRC	Factory test — not a user register
9h	---	Reserved
Ah	ID	Part number/ Rev ID
Bh	---	Reserved
Ch	DATA0LOW	Low byte of ADC channel 0
Dh	DATA0HIGH	High byte of ADC channel 0
Eh	DATA1LOW	Low byte of ADC channel 1
Fh	DATA1HIGH	High byte of ADC channel 1

Ilustración 13 Tabla de registros del sensor TSL2561

Como vemos, existen dos canales del convertidor que nos dan información acerca de la iluminancia. Esto se debe a que el propio sensor toma medidas de iluminancia de radiación infrarroja solamente o de radiación visible e infrarroja.

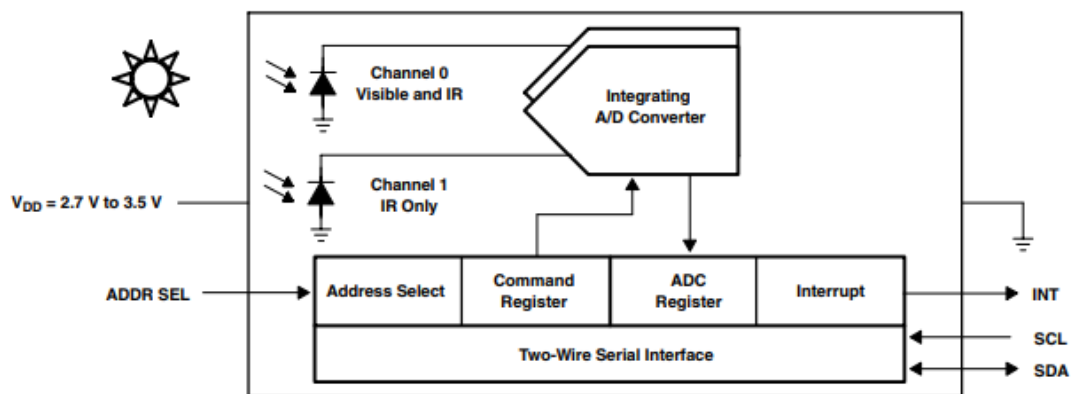


Ilustración 14 Diagrama de funcionamiento del sensor de luz

En este caso se podrán tomar ambos valores por separado, según la dirección que se indique al enviar el byte de COMMAND.

3.1.2 Sensor de color

El sensor de color es el TCS3471 que permite estimar el color de la luz que incide sobre el sensor.

El módulo empleado es el "color click" de Mikroelektronika. Al igual que el sensor de luz, sólo acepta alimentación a 3,3V y la salida del sensor se obtiene por comunicación I2C.

Con este sensor se puede determinar, en escenas que empleen iluminación con colores, si el color de la estancia es el deseado y corregir pequeñas desviaciones. El sensor cuenta con

detectores a los colores rojo, verde, azul y blanco y posee una resolución de 16 bits en las medidas.



Ilustración 15 Módulo color click

Además, incluye un LED RGB que permitiría estimar colores de superficies próximas en situaciones de escasa iluminación.

TCS3471 Block Diagram

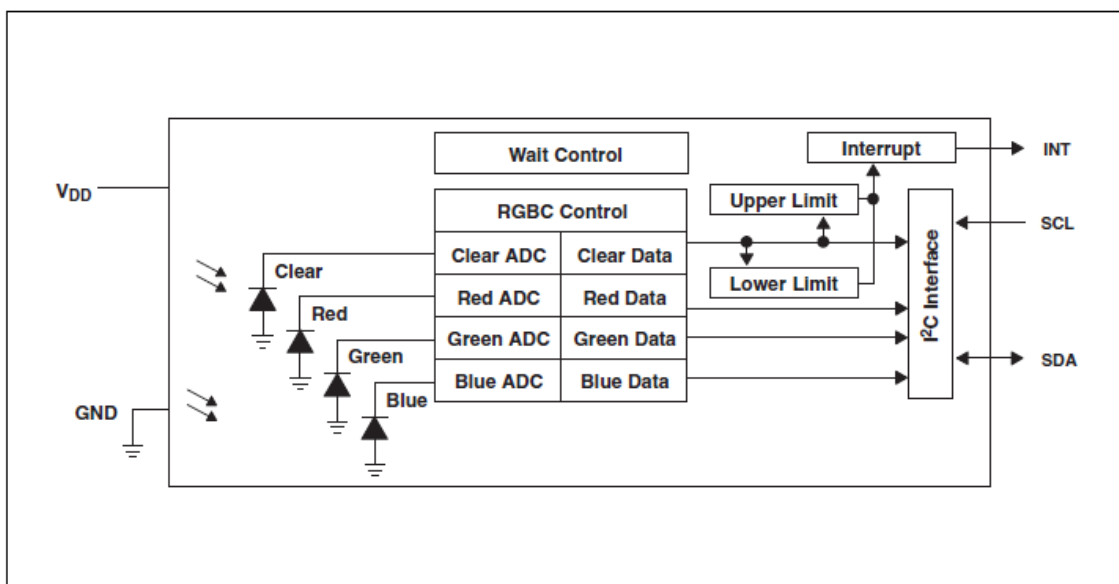


Ilustración 16 Diagrama de bloques del sensor de color TCS3471

El sensor TCS3471 que integra dispone de la siguiente tabla de registros, en la que se almacenan los datos que obtiene. Las direcciones son de 5 bytes y es necesario especificar la dirección a la que se desea acceder cuando se solicitan datos al sensor a través del bus I2C.

Address	Register Name	R/W	Register Function	Reset Value
--	COMMAND	W	Specifies register address	0x00
0x00	ENABLE	R/W	Enables states and interrupts	0x00
0x01	ATIME	R/W	RGBC ADC time	0xFF
0x03	WTIME	R/W	Wait time	0xFF
0x04	AILTL	R/W	RGBC interrupt low threshold low byte	0x00
0x05	AILTH	R/W	RGBC interrupt low threshold high byte	0x00
0x06	AIHTL	R/W	RGBC interrupt high threshold low byte	0x00
0x07	AIHTH	R/W	RGBC interrupt high threshold high byte	0x00
0x0C	PERS	R/W	Interrupt persistence filters	0x00
0x0D	CONFIG	R/W	Configuration	0x00
0x0F	CONTROL	R/W	Gain control register	0x00
0x12	ID	R	Device ID	ID
0x13	STATUS	R	Device status	0x00
0x14	CDATA	R	Clear ADC low data register	0x00
0x15	CDATAH	R	Clear ADC high data register	0x00
0x16	RDATA	R	Red ADC low data register	0x00
0x17	RDATAH	R	Red ADC high data register	0x00
0x18	GDATA	R	Green ADC low data register	0x00
0x19	GDATAH	R	Green ADC high data register	0x00
0x1A	BDATA	R	Blue ADC low data register	0x00
0x1B	BDATAH	R	Blue ADC high data register	0x00

Ilustración 17 Tabla de direcciones de los registros

Estas direcciones se especifican dentro del byte de la instrucción COMMAND, en concreto son los últimos 5 bits de dicha instrucción.

De todos los datos recogidos en la tabla de registros, serán de interés los que hacen referencia a los convertidores ADC CRGB, que se ubican a partir de la dirección 0x14.

7	6	5	4	3	2	1	0
COMMAND	TYPE		ADD				

Field	Bits	Description	
COMMAND	7	Select Command Register. Must write as 1 when addressing COMMAND register.	
TYPE	6:5	Selects type of transaction to follow in subsequent data transfers:	
		FIELD VALUE	INTEGRATION TIME
		00	Repeated byte protocol transaction
		01	Auto-increment protocol transaction
		10	Reserved — Do not use
		11	Special function — See description below
		Byte protocol will repeatedly read the same register with each data access. Block protocol will provide auto-increment function to read successive bytes.	
ADD	4:0	Address field/special function field. Depending on the transaction type, see above, this field either specifies a special function command or selects the specific control-status-register for following write and read transactions. The field values listed below apply only to special function commands:	
		FIELD VALUE	READ VALUE
		00000	Normal — no action
		00110	RGBC interrupt clear
		other	Reserved — Do not write
		RGBC Interrupt Clear. Clears any pending RGBC interrupt. This special function is self clearing.	

Ilustración 18 Instrucción COMMAND necesaria para solicitar datos al sensor

3.1.3 Memoria EEPROM

Se utilizará también una memoria EEPROM externa que servirá para poder almacenar algunas configuraciones y escenas predeterminadas, independientemente del microcontrolador utilizado. Si bien hay microcontroladores que disponen de memoria EEPROM interna, al sustituir el microcontrolador cambiaríamos la memoria EEPROM y sería necesario volver a guardar todos los ajustes en la memoria EEPROM del nuevo dispositivo, lo que reduce la versatilidad y flexibilidad. Por ello se ha decidido emplear una memoria EEPROM externa, en concreto el modelo de Atmel AT24CM02, de 256KB, a la que se accede por I2C, como en los sensores anteriores y que permite una alimentación de 3,3V o 5V. En este trabajo se optará por la alimentación de 3,3V para poder utilizar la misma línea de alimentación para todos los sensores.



Ilustración 19 Módulo EEPROM 3 click

Los 256KB de memoria serán suficientes para poder almacenar algunas escenas predeterminadas y algunos parámetros de configuración que no queremos perder tras apagar el sistema.

3.1.4 Conexión al bus DALI

La conexión al bus se realiza a través del módulo DALI click. En este módulo se incluye todo lo necesario para realizar la conexión al bus y dispone de optoacopladores para proporcionar aislamiento entre el bus y nuestro circuito. Al igual que los módulos anteriores permite la alimentación a 3,3V. En el módulo se integran algunos LEDs que nos permiten conocer el estado de la comunicación y del módulo.

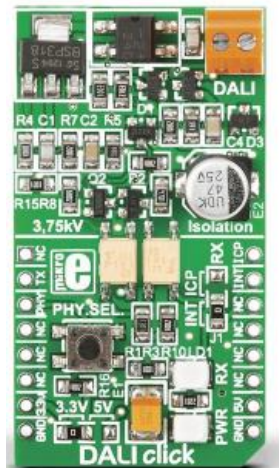


Ilustración 20 Módulo DALI click que permite la conexión al bus DALI

3.1.5 Lector de tarjetas micro SD

Se empleará un lector de tarjetas microSD que permitirá almacenar datos para poder procesarlos posteriormente. En concreto para este TFM se ha empleado un módulo genérico, pero también se podría emplear el módulo microSD click de Mikroelektronika utilizando uno de los zócalos de mikroBus de la placa desarrollada. La ventaja de utilizar un módulo genérico es la posibilidad de enchufarlo en el zócalo de comunicación SPI y dejar libre un módulo MikroBus para conectar otro tipo de sensores. Además se puede conectar a través de unos cables a dicho

conector y tener una mayor versatilidad a la hora de ubicar el módulo dentro de una carcasa para dejar libre la ranura de la tarjeta microSD. El módulo empleado se puede ver en la Ilustración 21

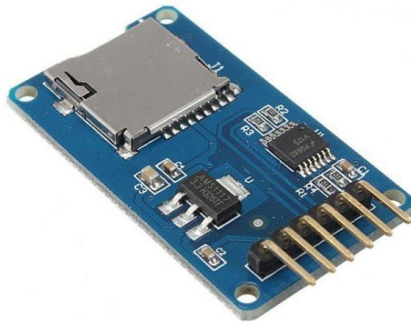


Ilustración 21 Módulo genérico microSD

El módulo utiliza una comunicación SPI, que es el estándar para este tipo de lectores.



Ilustración 22 Módulo lector de tarjetas microSD de mikroelektronika

En este caso se empleará la tarjeta microSD como medio de almacenamiento para la generación de los registros de funcionamiento y para la recopilación de datos de interés con el fin de poder procesarlos posteriormente en un ordenador. Estos datos de interés puede ser información acerca del nivel de iluminación, temperatura del color, consumo y otros datos en periodos de pocos minutos, para poder estudiar la evolución detallada de la estancia a lo largo de uno o varios días.

La generación de registros permitirá detectar fallos y guardarlos para poder realizar un seguimiento de los mismos posteriormente o conocer las condiciones en las que se produjo un fallo determinado.

3.1.6 Módulo Bluetooth HC05

Se empleará un módulo bluetooth, en concreto el módulo HC05 para proporcionar comunicación inalámbrica y compatibilidad con otros dispositivos como smartphones. De esta forma se abre la puerta a realizar algunas configuraciones del equipo a través de estos dispositivos de manera inalámbrica, como la carga de escenas o el control de algunos balastos.

HC-05 FC-114



Ilustración 23 Módulo bluetooth HC05 empleado

Gracias a este módulo, se puede desarrollar una aplicación para teléfonos móviles que interactúe con el sistema haciendo uso de la comunicación Bluetooth para realizar las acciones mencionadas anteriormente.

3.1.7 Arduino Nano

Se podrá emplear el microcontrolador Arduino Nano, uno de los más compactos de la familia Arduino para realizar este control. Este microcontrolador (Ilustración 3) cuenta con 13 pines de entrada/salida digital, de los cuales, 6 poseen la capacidad de generar PWM, 8 canales DAC y reguladores de tensión que pueden proporcionar salidas de tensión a 3,3V y 5V.

Arduino nano lleva en su interior un microcontrolador ATmega 328, que cuenta con 32KB de memoria flash para el programa, 2KB de memoria SRAM para variables y funciona a una frecuencia de 16MHz gracias a un cristal oscilador externo soldado en la placa de Arduino.

Como se ha visto, es un microcontrolador bastante modesto, y que ya nos permite realizar esta funcionalidad.

3.2 Diseño de la PCB

Para el desarrollo del presente trabajo se va a diseñar una placa de circuito impreso (PCB) que permita integrar todos los elementos hardware mencionados anteriormente y sustituya la placa de desarrollo Easy Pic Pro v7, que es la utilizada previamente para la realización del control de la luminaria RGB a través de DALI.

La PCB se desarrollará en material FR4, con espesor de cobre de 35 micras y de doble cara.

En ella, se integrarán los siguientes elementos:

- 3 zócalos MikroBus S
- 1 Zócalo para el módulo bluetooth HC-05
- 1 Zócalo para MikroBus L
- 1 Zócalo universal para MCU
- 1 Zócalo para lector de tarjetas microSD
- 1 Puerto de programación ICSP para MCU PIC
- 1 Regulador integrado de 3,3V
- 1 Puerto micro-USB para alimentación de la tarjeta a 5V.

Los zócalos MikroBus S servirán para poder conectar a la tarjeta el sensor de luz, de color y la memoria EEPROM externa, entre otros.

En el zócalo MikroBus L servirá para conectar el adaptador DALI.

Además, es posible crear nuevos módulos personalizados con otros sensores compatibles con el bus I2C siempre que se adapten al factor de forma de los zócalos de MikroBus.

Mikroelektronika, detalla en el documento adjunto “MikroBus Standard Specifications.pdf” las características que deben tener los módulos desarrollados para que sean compatibles con el estándar MikroBus.

Pinout specification

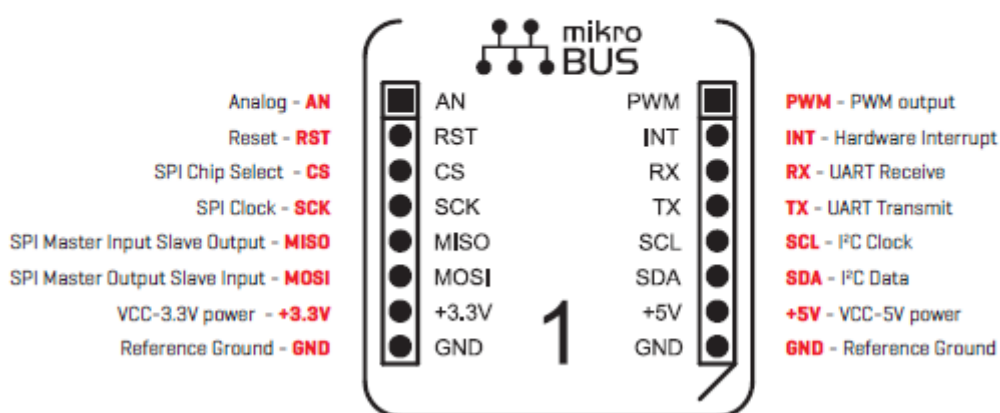


Ilustración 24 pinout de los zócalos de MikroBus

Siguiendo estas especificaciones, es posible crear módulos personalizados que puedan ser añadidos al sistema y ser compatibles con ellos.

En este mismo documento, se indican también las medidas que deben tener los zócalos de cada uno de los tamaños estandarizados de MikroBus.

El zócalo universal permitirá conectar la MCU a la tarjeta. Como ya se ha comentado, será posible conectar tanto microcontroladores PIC como un Arduino Nano. Esto hace que el desarrollo del zócalo se complique.

Para realizar el desarrollo de este zócalo, se ha tenido en cuenta lo siguiente:

Sólo se debe permitir la conexión de uno de los dos dispositivos de forma simultánea, lo que lleva a la colocación de los zócalos superpuestos, total o parcialmente, de forma que una vez colocada una MCU no se pueda conectar otra porque se produce una oclusión total o parcial de los pines del otro zócalo.

Ambos zócalos deben ser lo suficientemente diferentes como para evitar una confusión en su inserción. De esta forma se evita que se inserte el microcontrolador PIC en el zócalo y Arduino y viceversa.

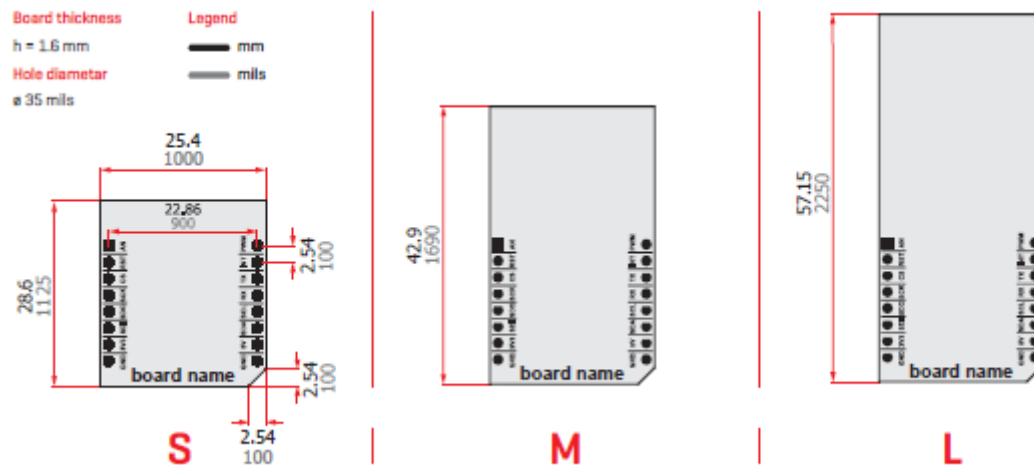


Ilustración 25 Tamaños estandarizados de zócalos MikroBus

En caso de solapamiento de los zócalos, se deberá respetar, en la medida de lo posible, que las características de los pines superpuestos sea la misma, o en su defecto lo más parecida posible en ambos microcontroladores. Esto quiere decir que si se produce un solapamiento de un pin de Arduino que tenga función PWM, es deseable que el pin del PIC que coincida en esa posición tenga dicha función también para evitar incompatibilidades de los periféricos conectados a ese pin.

En base a todas estas premisas, se desarrolló un zócalo universal, que se puede ver en la Ilustración 26.

Finalmente se ha optado por un zócalo compartido parcialmente por ambos microcontroladores, de forma que una vez colocado uno de ellos no permite la inserción del otro. Además, con esta distribución se consigue ahorrar espacio en la placa de circuito impreso, ya que ambos ocupan casi el mismo espacio físico y se produce una coincidencia de funciones de los pines solapados, lo que no restringe la funcionalidad de los dispositivos que se conecten en dichos pines.

Al utilizar el zócalo desarrollado, se produce solapamiento de los pines 20, 21, 10 y 11 de Arduino con los pines 35, 36, 24 y 23 del PIC respectivamente.

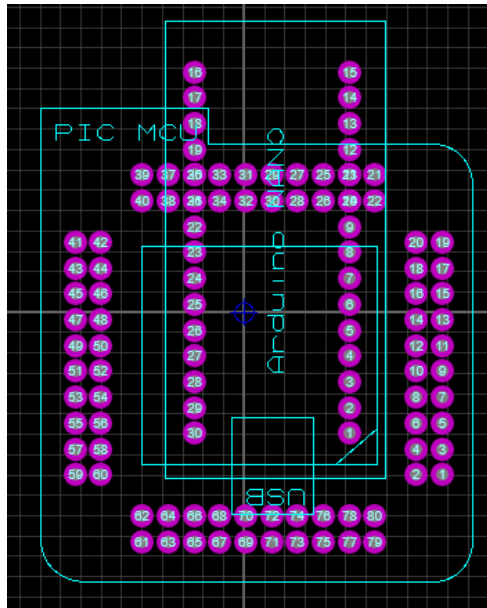


Ilustración 26 Zócalo universal compatible con Arduino y los PIC mencionados

Analizando las características de dichos pines nos encontramos con lo siguiente:

Pin de Arduino	Pin del equivalente PIC	Descripción del pin de Arduino	Función del pin del PIC
10	24	A6	RF0
11	23	A7	RF1
20	35	D2	RC1
21	36	D3	RC0

Analizando los microcontroladores compatibles con el zócalo instalado, como es por ejemplo el PIC 18F8520 encontramos las siguientes funciones para los pines especificados anteriormente:

RF0 coincide con AN5. Esto indica que está conectada a un canal del convertidor analógico-digital (ADC), por lo que la función del pin de Arduino y del pin del PIC coincide en ese aspecto.

RF1 coincide con AN6, por lo que de igual modo que en el caso anterior, ambos pines pueden ser utilizados como entradas a un canal del ADC en ambos microcontroladores.

Finalmente, para los pines RC0, coincidente con D3 y RC1, coincidente con D2, pueden usarse todos ellos como entrada/salida digital.

Con esto vemos que con este zócalo se aprovechan al máximo las características de ambos microcontroladores al ubicarlos sobre el mismo zócalo.

Otra de las consideraciones a tener en cuenta es el tema de las interrupciones hardware de los dispositivos. Para realizar la comunicación con el bus DALI, es necesario disponer de al menos un pin con interrupción hardware, ya que se empleará para identificar las respuestas de las luminarias en algunos comandos. Debido a que en el protocolo DALI sólo puede ser el maestro el microcontrolador, el bus utiliza únicamente un pin para la transmisión de datos (TX) y el pin de interrupción (INT) para indicar al microcontrolador que alguien conectado al bus ha respondido a un mensaje. Por ello, es necesario que dicho pin del módulo esté conectado a un pin que soporte interrupciones hardware. Para ello se ha estudiado qué pines de los

microcontroladores soportan este tipo de interrupciones en ambos microcontroladores para realizar el rutado teniendo en cuenta esta consideración y se ha visto que para el Arduino Nano únicamente los pines D2 y D3, que son los que coinciden con el zócalo del microcontrolador PIC, pueden ser utilizados para interrupciones externas. Al ser pines coincidentes se ha contrastado si los pines RC1 y RC0 (pines coincidentes del microcontrolador PIC) soportan este tipo de interrupciones, pero en la hoja de datos de los microcontroladores PIC compatibles no se puede asegurar que estos pines cuenten con esa función. Para el caso del microcontrolador PIC 18F8520, los pines que soportan interrupciones son RB0, RB1 y RB2. En este caso, se ha decidido conectar el pin RB1 del zócalo PIC al pin de interrupción del zócalo mikroBus L, que es donde se conectará el módulo DALI click. Para el caso de Arduino, se ha decidido tomar el pin D2 para recibir las interrupciones del módulo DALI click.

De esta forma, las interrupciones del módulo DALI click están conectadas al pin D2 de Arduino y al pin RB1 del PIC. Esto deberá ser tenido en cuenta a la hora de realizar el programa correspondiente en cada microcontrolador, para garantizar que las interrupciones saltan y se tratan de manera adecuada.

El diseño final de la placa queda como se muestra en la Ilustración 27. En ella se puede ver la distribución de todos los componentes y la presencia del regulador de tensión integrado en la placa junto con el conector micro USB en la esquina inferior derecha. Estos componentes integrados en la placa permiten alimentar el circuito cuando se encuentra conectado el microcontrolador PIC, que al no disponer de un puerto USB como Arduino, es necesario alimentar la placa de otra forma. La misión del regulador es proveer de una alimentación de 3,3V a los módulos que necesiten esa tensión, como es el caso del sensor de luz, que sólo acepta hasta 3,3V en la alimentación por lo que no ha sido posible alimentarlo directamente con la tensión del USB.

Con el fin de que el proceso de fabricación sea sencillo, se ha empleado un grosor y un espaciado de las pistas muy superior al recomendado por el estándar IPC2221B, para la fabricación de placas de circuito impreso. Según dicho estándar, la separación mínima de dos conductores en una placa de tipo B2, que es el que aplica a nuestro desarrollo (placas con conductores externos sin aislante para utilizarse a alturas comprendidas entre el nivel del mar y 3050m), deberá ser como mínimo de 0,1mm cuando la diferencia de tensión entre ambas es menor de 30V. En este caso, se ha establecido como regla de diseño que la distancia entre dos conductores no sea inferior a 0,385mm, lo que supera con creces la recomendación mínima. Además, la distancia entre una pista y un pad ó dos pads se ha establecido en 0,254mm, más del doble del mínimo recomendado.

En cuanto a la anchura de las pistas, este estándar establece unas recomendaciones en función de la corriente que circula por ella, el incremento de temperatura máximo que se desea tener en la pista y el grosor de la capa de cobre. En este caso, la máxima tensión a la que estará sometido el circuito serán 5V, y la máxima corriente que circulará por una sola pista no superará los 2A. Para este caso, el grosor de la pista deberá de ser 24mils. Por ello se ha establecido el valor mínimo para la anchura de las pistas en 25 mils, siendo las líneas de alimentación de 40mils, muy por encima de los requisitos. Esto facilita la fabricación de la PCB.

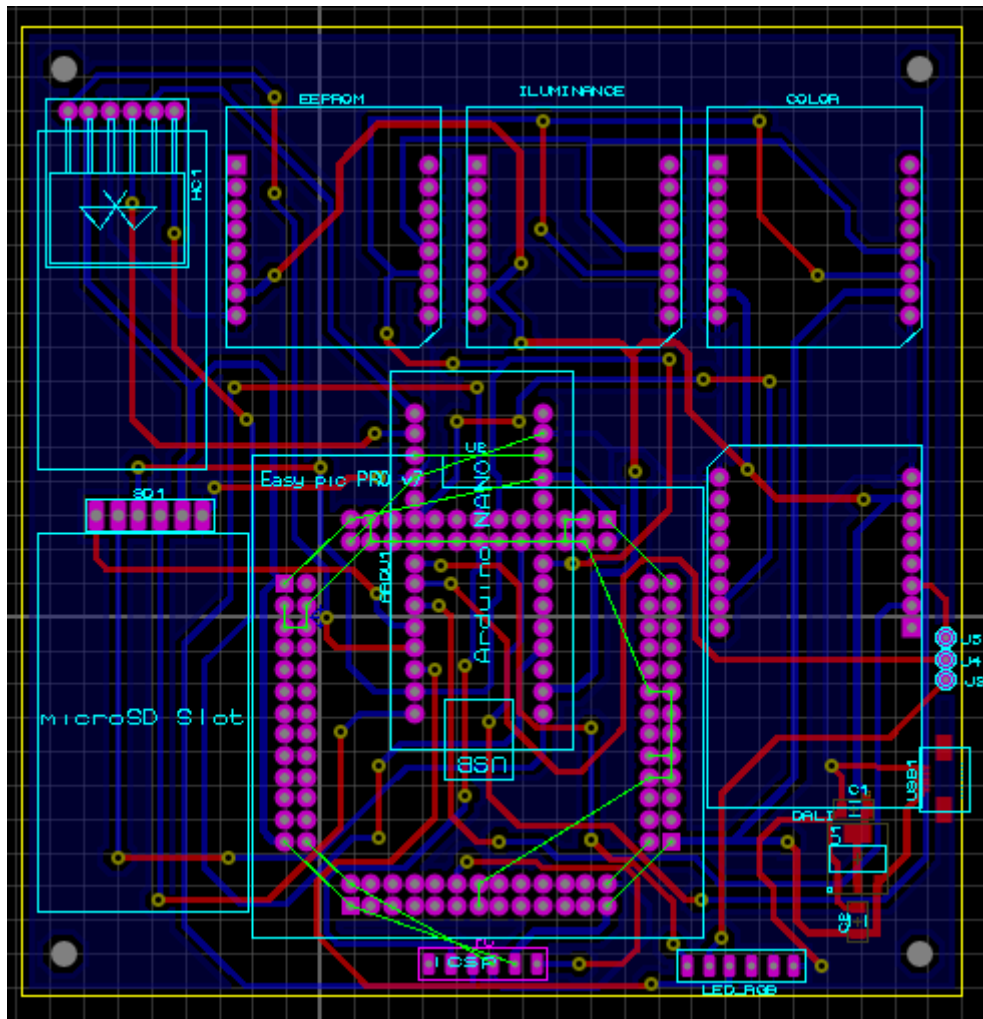


Ilustración 27 Diseño final de la PCB desarrollada

Para comprobar la disposición de los elementos en la placa y verificar que no se producen colisiones entre ellos y que las huellas de los dispositivos creados respetan el espacio físico de los mismos se han creado los modelos tridimensionales de cada uno de los componentes que se utilizan para poder obtener una previsualización 3D de la placa. Algunos de los dispositivos ya contaban con un modelo 3D previo, por lo que no ha sido necesaria la realización de esos modelos. Los dispositivos que ya contaban con un modelo 3D son:

- Regulador 3,3V LD1117S33
- Condensadores SMD C1 y C2
- Tira de pines para conexión de elementos ICSP y conexión de módulos de tarjetas microSD.

Para el resto de elementos se ha creado el componente empleando la herramienta de diseño 3D Autodesk Inventor Profesional 2017 y se ha exportado en un formato compatible con el software de desarrollo electrónico.

Los modelos creados han sido los siguientes:

- Módulo Mikro Bus S
- Módulo Mikro Bus L
- Módulo Bluetooth HC-05

Finalmente se han importado los modelos restantes (Arduino Nano y conector microUSB) de una página de diseño de elementos 3D al ser elementos genéricos ha sido sencillo encontrar modelos bastante completos de esos componentes.

El aspecto de la placa en el visualizador 3D se puede ver en la Ilustración 28. En este modelo se ha podido comprobar que ningún elemento colisiona con otro una vez conectados en la placa. Algunos de los componentes que tenían este riesgo eran el conector micro USB y el módulo Mikro Bus L, que corresponde con el módulo DALI click. Al estar ubicado el conector micro USB bajo dicho módulo existía una posibilidad de que se produjese un choque entre el módulo y el conector y gracias al modelo 3D se ha podido verificar que no existe contacto.

Otras de las funcionalidades del modelo 3D permite comprobar visualmente el emplazamiento de los componentes en ambas caras de la placa de circuito impreso, que de otra forma puede resultar más confuso. De esta forma se puede ver claramente como el conector ISCP tiene el acceso por la parte inferior de la placa de circuito impreso. Esto se ha decidido así para dejar despejada la entrada USB del Arduino cuando éste esté conectado.

Finalmente, se realizará un análisis de las dimensiones de la placa desarrollada, comparándola con la Easy Pic Pro V7 utilizada anteriormente para comprobar si se ha conseguido reducir el tamaño de la placa de forma notoria.

Las dimensiones de la placa Easy Pic Pro v7 son de 266x220mm, mientras que la placa desarrollada tiene unas dimensiones de 111x103mm. La comparativa visual del tamaño se puede observar en la Ilustración 29.

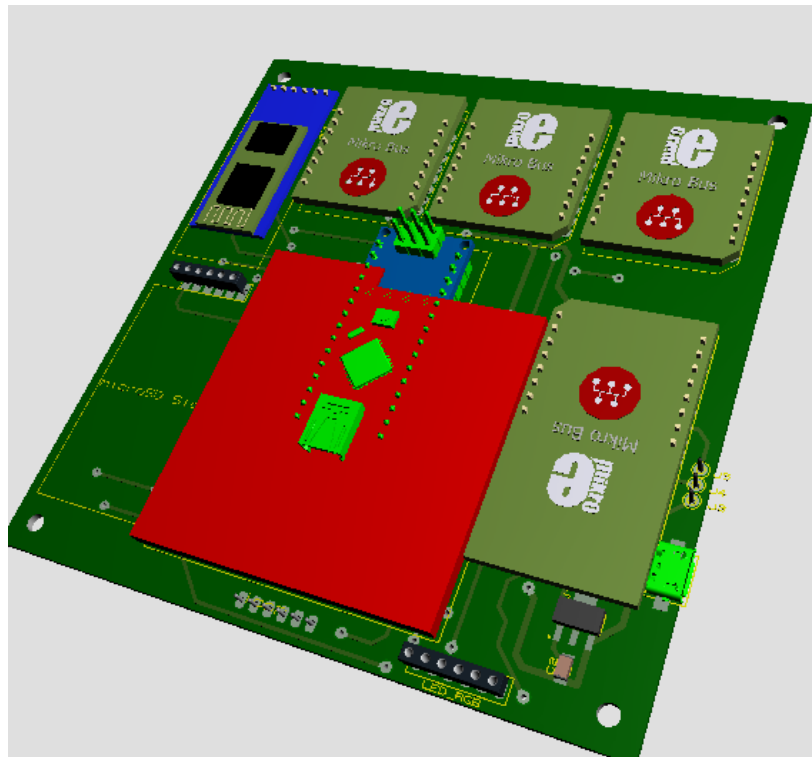


Ilustración 28 Modelo 3D de la placa de circuito impreso desarrollada

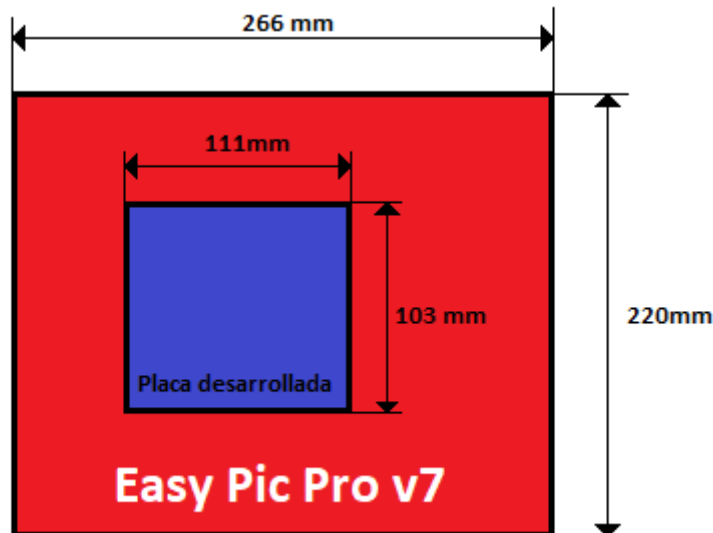


Ilustración 29 Comparativa de tamaño entre la antigua placa y la nueva

Con ello vemos que se ha conseguido reducir el tamaño de la placa original a un quinto del tamaño original, obteniendo una reducción de tamaño de aproximadamente el 80% consiguiendo así una tarjeta compacta y específica para el desarrollo de dicha función.

3.3 Implementación con DALI y leds RGB genéricos

Gracias a la disponibilidad de un módulo DALI click, se ha implementado la placa con compatibilidad para emplear este protocolo. Para ello, basta con conectar el bus DALI al conector del módulo DALI click destinado para ello y el sistema se encargará de realizar el envío de comandos a través de este módulo.

Para garantizar la compatibilidad de este sistema con DALI, es necesario tener en cuenta algunos requisitos a la hora de realizar el diseño de la PCB y sus conexiones.

En primer lugar, es necesario disponer de un pin que soporte interrupciones hardware conectado al pin "INT" del módulo DALI click. Por este pin, el módulo comunica al microcontrolador la respuesta de los dispositivos conectados al bus. Este pin es fundamental en las tareas de direccionamiento.

Por otro lado, es necesario conectar un pin de salida digital del microcontrolador al pin TX del módulo. A través de él se realizará el envío de comandos DALI.

Para el microcontrolador Arduino se ha optado por utilizar el pin D2 y para el microcontrolador PIC el pin RB1, como se ha comentado en el apartado anterior.

Con estas conexiones, es posible enviar las escenas y configuraciones de iluminación a luminarias compatibles con este protocolo.

En la Ilustración 32 y la Ilustración 33 se muestran algunas de las escenas implementadas en una luminaria DALI disponible en el laboratorio de electrónica.

Además de implementar esta compatibilidad con DALI, el sistema está preparado para funcionar con tiras RGB de forma directa. Para realizar las pruebas de funcionamiento con este tipo de sistemas se ha empleado el led RGB disponible en el módulo Color click, de forma que

se puede comparar el color que adquiere la luminaria DALI con el color de dicho led. En la Ilustración 33 se puede ver cómo la tonalidad de dicho led es similar a la mostrada por la luminaria DALI.

La conexión de un led RGB se realiza a los pines 5, 6 y 10 del microcontrolador Arduino, ya que todos ellos soportan función PWM, lo que nos permite ajustar el color resultante.

La utilización de esta funcionalidad anula el pin DALI_PHY del módulo DALI click por necesidad de emplear el pin D6 de Arduino. A pesar de ello, el impacto general es mínimo ya que en la mayoría de casos si se dispone del módulo DALI lo recomendable sería emplear el control a través del bus DALI. En caso de no disponer un bus DALI, se puede conectar una luminaria RGB tradicional y realizar el control a través del conector “LED_RGB” de la placa, por lo que no se perdería funcionalidad. La selección del modo de operación se realiza a través de un jumper uniendo el pin J4 con J5 para operar con la línea DALI_PHY o J4 con J3 para el control a través del conector LED_RGB. En ausencia de conexión el sistema podría trabajar con luminarias DALI sin necesidad de emplear la línea DALI_PHY y podría controlar luminarias LED de hasta dos colores, ya que la línea que entra en conflicto con DALI_PHY es el color verde la luminaria. También se podrían controlar dos luminarias blancas con los canales Rojo y Azul del conector LED_RGB.

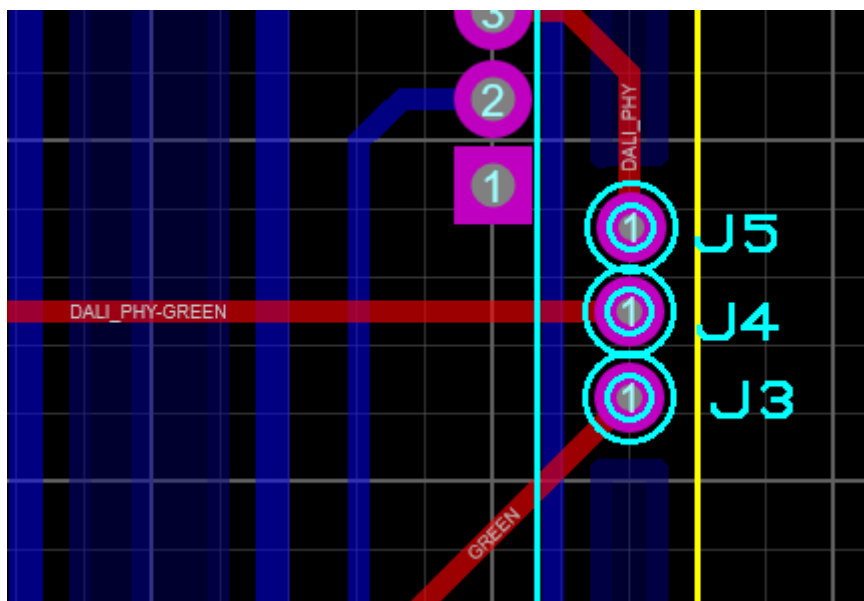


Ilustración 30 Jumper de selección para el modo DALI o LED_RGB

El conector para emplear LEDs RGB es un conector de seis pines como el que se muestra en la Ilustración 31. El pinout de dicho conector es tal como se muestra en la siguiente tabla:

Número de pin	Función
1	GND
2	GND
3	GND
4	Azul
5	Verde
6	Rojo

Tabla 2 Pinout del conector para LEDs RGB

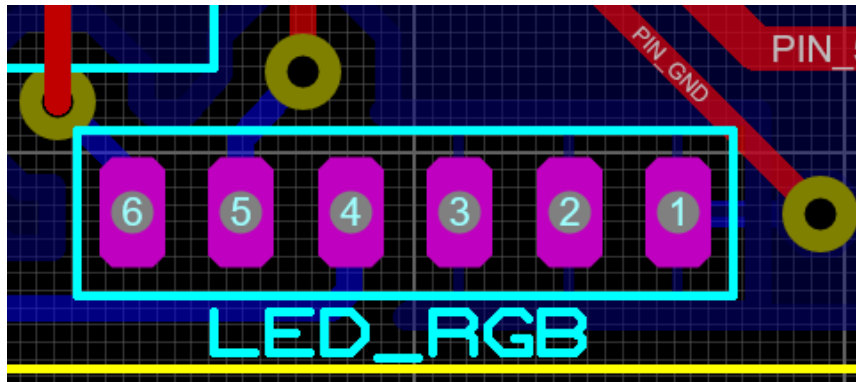


Ilustración 31 Conector para LEDs RGB

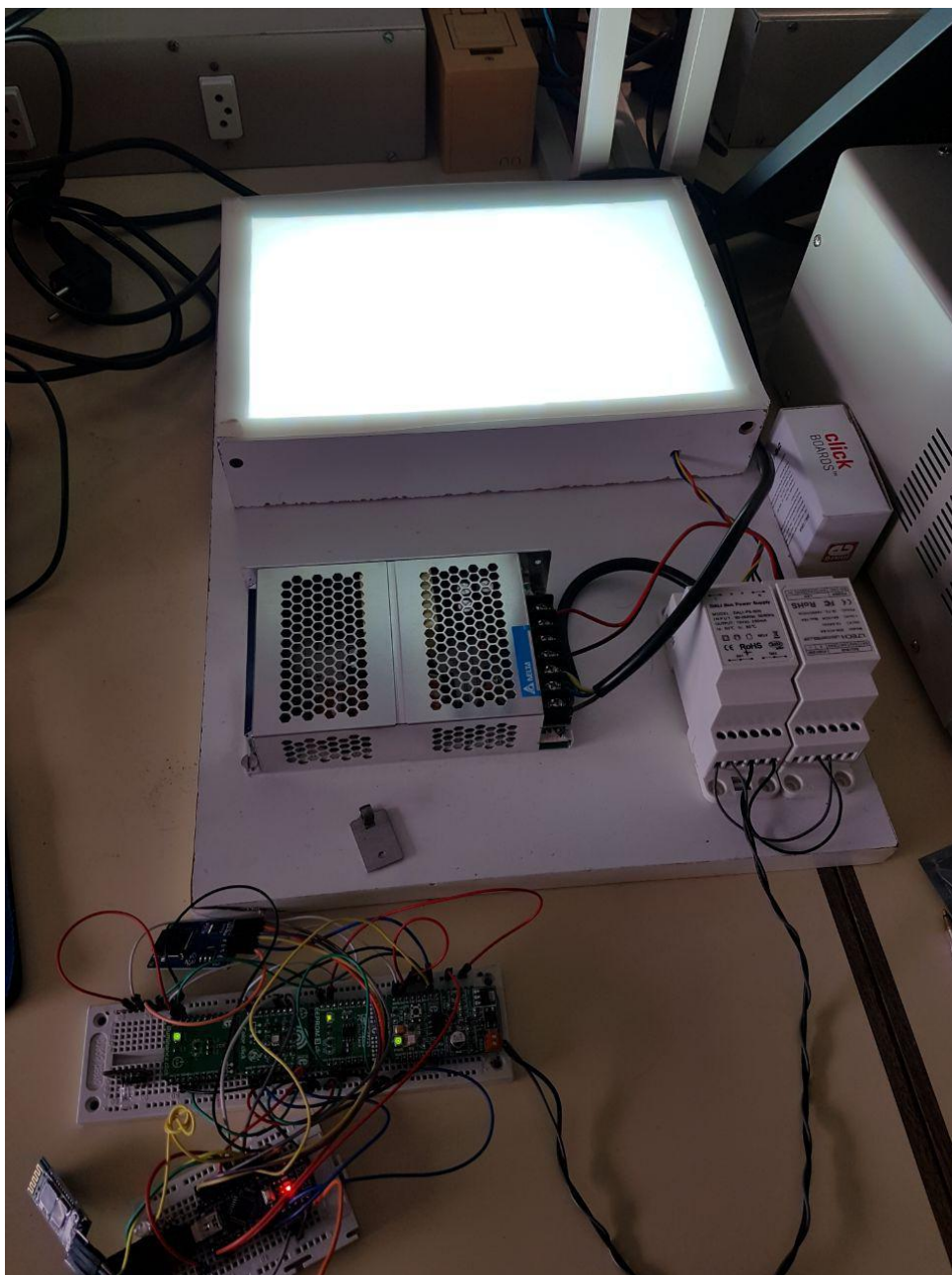


Ilustración 32 Escena de luz blanca 4000K en la luminaria DALI

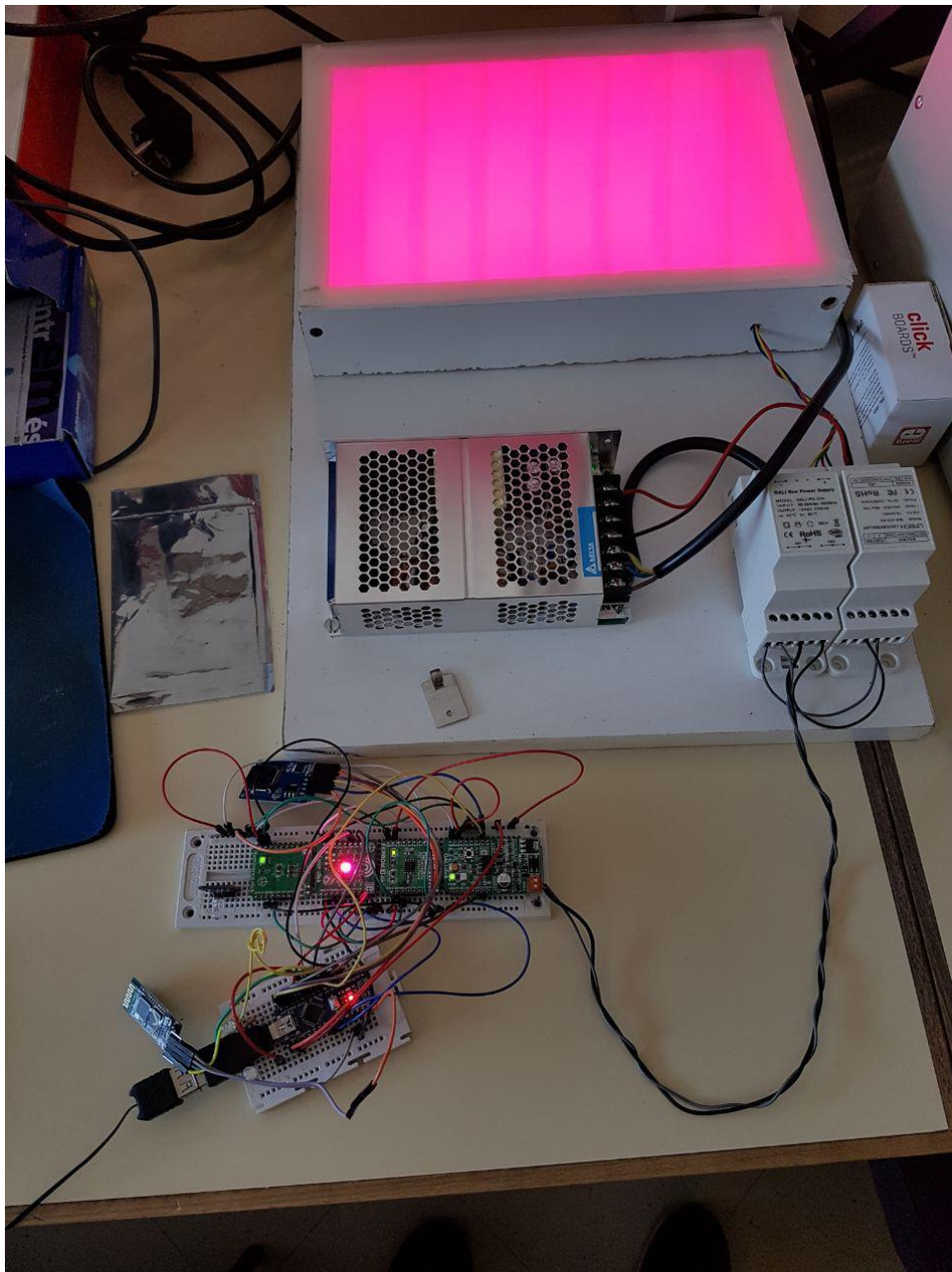


Ilustración 33 Escena dinámica Rojo-Azul en luminaria DALI y led RGB

3.4 Diseño de la envolvente

Por seguridad y para proteger el sistema, se ha decidido diseñar una carcasa para introducir el sistema desarrollado. La caja es de un diseño sencillo favoreciendo su fabricación con una impresora 3D.

La envolvente constará de dos partes principales que se encajan una sobre otra para cerrar el sistema además de contar con algunas trampillas que se abren y se cierran para dejar accesibles ciertos puertos de la placa diseñada.

En la parte interior se dispone la ranura para poder insertar y extraer la tarjeta micro SD y dos huecos que serán tapados por las trampillas posteriormente.

En esta parte también se encuentran unos pequeños soportes en la base que sirven para apoyar y atornillar la placa de circuito impreso con el fin de lograr una buena sujeción.

En la ranura de la tarjeta microSD se ha diseñado una hendidura en la carcasa que facilite la inserción y extracción de la tarjeta, ya que una vez introducida queda enrasada con la placa donde se ubica el lector y sería complicado extraerla, ya que su extracción se hace pulsando la tarjeta.

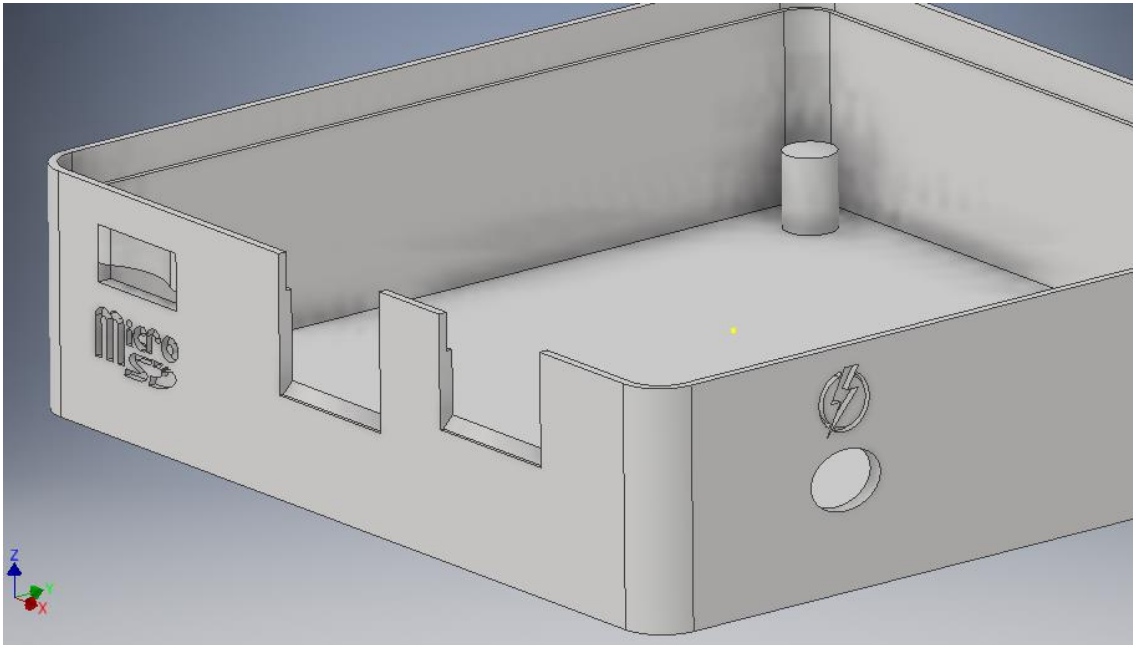


Ilustración 34 Parte inferior de la envoltente diseñada

Por otro lado, en el lateral derecho de la carcasa se encuentra un orificio para conectar un micro-USB y alimentar la placa en caso de emplear un microcontrolador PIC. Junto al orificio se ha añadido una serigrafía con forma de rayo que permite identificar de forma más clara la función de ese conector.

El último detalle por considerar en esta parte de la envoltente es el bisel generado en la parte superior que permitirá encajar a presión ambas partes de la carcasa.

La parte superior de la envoltente es bastante más baja ya que servirá únicamente como tapadera. En ella se encuentran los dos huecos para colocar las trampillas y el mecanismo de inserción para las mismas por la parte inferior y una ventana en la parte superior que permite que la luz del exterior incida sobre los módulos de MikroBus que disponen de los sensores de iluminancia y color.

Esta parte cuenta con el mismo bisel que la parte inferior y que permitirá encajarlas posteriormente.

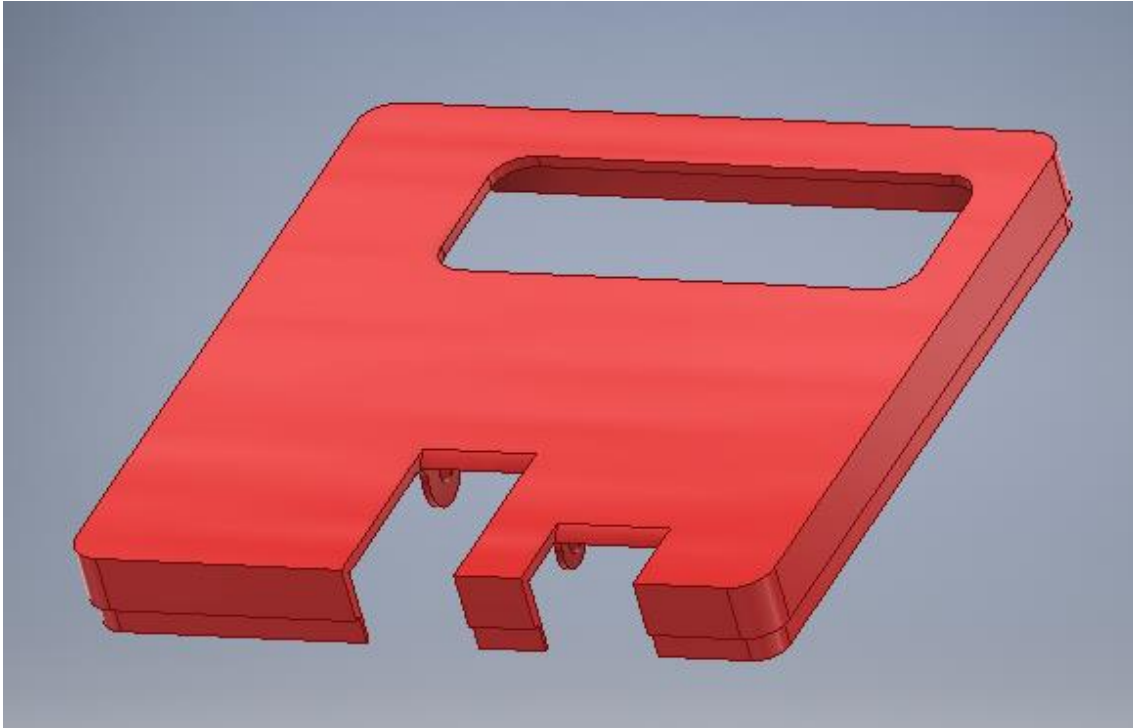


Ilustración 35 Parte superior de la envoltente

Las trampillas que se han mencionado anteriormente servirán para tapar el conector para LEDs RGB, la regleta que permite la conexión al bus DALI y el puerto ICSP de programación para el PIC y el USB de la placa Arduino Nano cuando sea éste el microcontrolador conectado.

Cada una de las tapaderas tiene una serigrafía en su frontal que permite identificar las partes que cubre cada una.

En la Ilustración 36 se puede ver el diseño de la trampilla que cubre los conectores del bus DALI y los LED RGB. La serigrafía del frontal es un LED con un rayo en su interior para facilitar su identificación y asimilación. Se ha decidido poner esta serigrafía porque un LED es un símbolo muy conocido por una amplia parte de la sociedad y es asimilado rápidamente con la iluminación mientras que el logotipo de DALI es un símbolo más específico que sólo entenderán e interpretarán adecuadamente personas entendidas o que trabajen en ello. Además, como existe la posibilidad de conectar elementos que no sigan el protocolo DALI, como son las tiras de LED RGB no sería apropiado poner ese logotipo en el frontal.

En la Ilustración 37 se muestra la trampilla que oculta el puerto USB del Arduino y el puerto de programación ICSP. El diseño es similar a la anterior y la serigrafía elegida en este caso han sido las letras ICSP en la parte inferior del frontal y un icono con el símbolo del USB y una batería que representa la posibilidad de alimentar el sistema a través de ese puerto USB cuando se utilice el microcontrolador Arduino, empleando por ejemplo una batería externa.

Ambas trampillas cuentan en su parte inferior con un pequeño resalto que facilita su apertura y la terminación del canto inferior de ambas trampillas está cortado en ángulo para proporcionar un cierre adecuado y evitar desniveles entre las paredes de la carcasa y las tapaderas una vez cerradas.

Para el sistema de apertura se ha optado por unas bisagras circulares que permiten una apertura suficiente de las trampillas y una vez cerradas quedan ocultas en el interior de la carcasa. De esta forma se respeta el aspecto exterior de toda la envolvente.

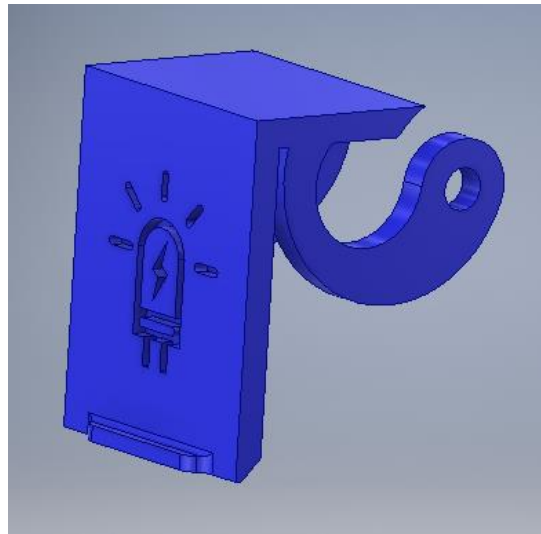


Ilustración 36 Trampilla conectores LED y bus DALI

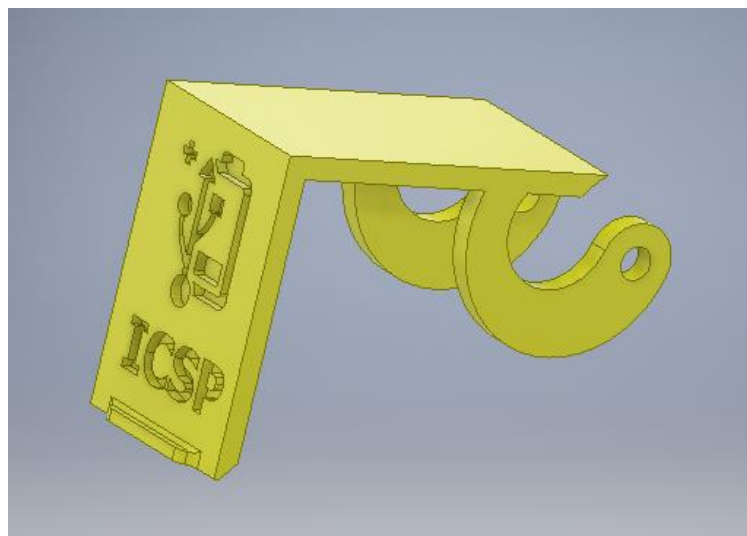


Ilustración 37 Trampilla conector USB Arduino y puerto ICSP del PIC

En la Ilustración 38 se muestra una vista en corte de la bisagra diseñada. El diseño redondeado de los bordes tanto de la trampilla como del hueco en la parte superior de la carcasa permite una apertura sin roces de ambas partes y la unión semicircular evita colisiones al abrir la trampilla.

La unión entre la trampilla y la carcasa se hace introduciendo el orificio en las patillas de la trampilla en un saliente diseñado especialmente para ello en la zona superior de la carcasa.

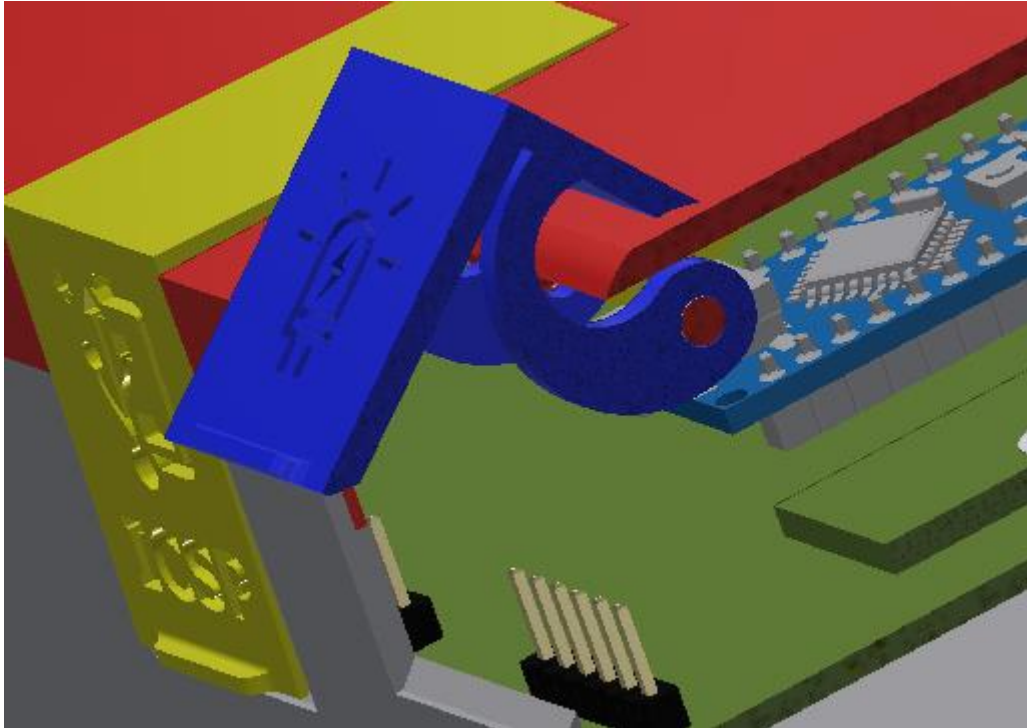


Ilustración 38 Funcionamiento de la bisagra diseñada

En la Ilustración 39 se muestra el mecanismo con la trampilla cerrada, donde se ve el ajuste entre la trampilla y ambas partes de la carcasa (redondeo de la parte superior y chaflán de la parte inferior).

Este sistema permite extraer los cables de forma discreta y no produce colisiones en el interior de la carcasa.

Para probar el correcto dimensionamiento de todos los elementos diseñados se creó un modelo 3D muy básico de la PCB desarrollada y los componentes empleados que tuviese las principales características de la PCB y que permitiese comprobar que toda la envolvente estaba correctamente diseñada.

Para ello el modelo 3D de la PCB realizado cuenta con los siguientes elementos:

- Microcontrolador Arduino: Permite comprobar que la trampilla que deja al descubierto el puerto mini-USB de la placa está correctamente posicionada.
- Tira de pines ICSP y LEDs: Permiten comprobar que ambas trampillas, tanto la que cubre el ICSP como la que cubre el puerto de LEDs RGB están correctamente posicionadas y permiten acceder a los puertos una vez abiertas.
- Módulos Mikro Bus: Permite comprobar que la ventana está correctamente diseñada y deja al descubierto todos los módulos. La ventana deja al descubierto todos ellos ya que no existe ningún tipo de restricción a la hora de conectarlos y por tanto los sensores de iluminancia y color pueden estar ubicados en cualquier posición. Esta ventana permite además identificar rápidamente los módulos Mikro Bus conectados.



Ilustración 39 Vista en corte de la trampilla cerrada y el mecanismo oculto

- Conector micro-USB: Permite comprobar que el orificio lateral para la conexión del cable de alimentación (en caso de emplear un microcontrolador PIC) es correcto tanto en tamaño como en posición.

No se ha diseñado la tarjeta microSD porque al ir anclada al frontal se unirá a la PCB a través de cables lo que permite una mayor flexibilidad.

El resultado final es el que se puede ver en la Ilustración 40.

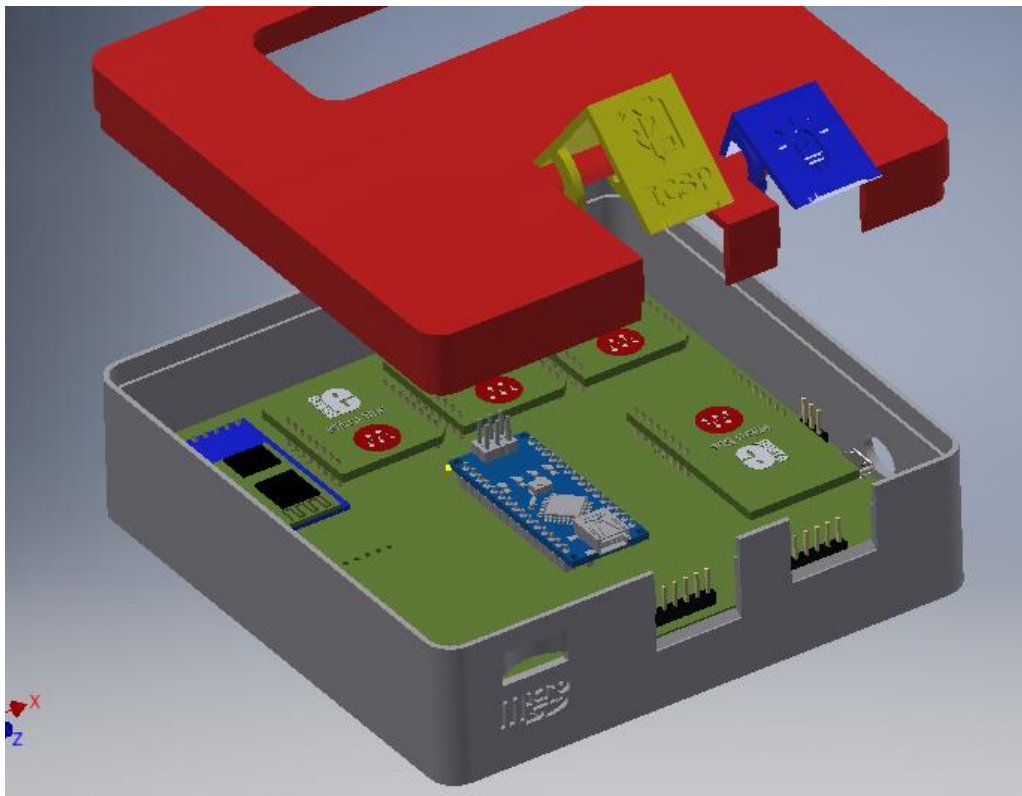
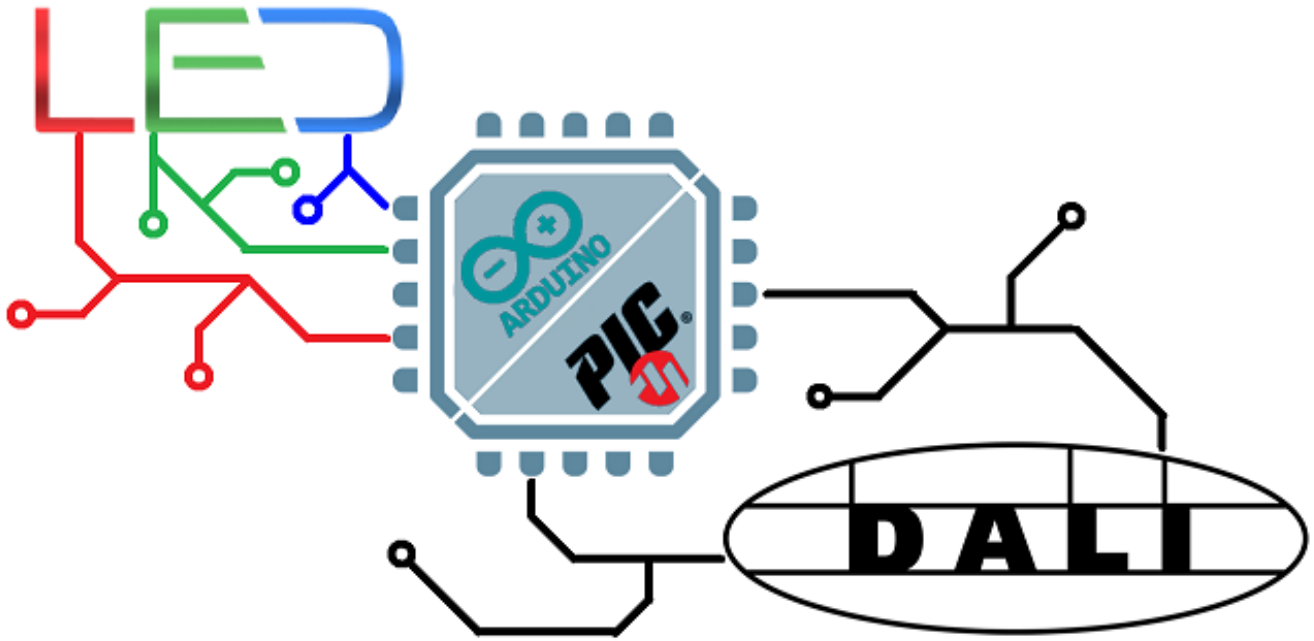


Ilustración 40 Resultado final de la envolvente

Todas las piezas se han diseñado pensando en su fabricación con impresora 3D, logrando que tanto la parte superior como la parte inferior de la envolvente se puedan fabricar sin necesidad de utilizar estructuras de soporte durante el proceso de impresión. En el caso de las trampillas, será necesario material de soporte únicamente en los brazos de unión, pero el acabado exterior no se ve afectado por ello ya que la parte impresa sobre los soportes queda oculta una vez instalada la trampilla.



4. Desarrollo software

Además del desarrollo de la placa de circuito impreso, se ha querido complementar el trabajo fin de máster con otros desarrollos complementarios que agreguen valor al sistema y que pongan en valor lo aprendido en las diferentes asignaturas impartidas en este Máster. Entre el resto de desarrollos realizados se encuentran:

- Desarrollo de una aplicación sencilla para dispositivos Android que permita el control de las luminarias via bluetooth.
- Diseño de una envolvente para contener el sistema electrónico desarrollado
- Migración del software desarrollado para microcontroladores PIC a Arduino.

4.1 Aplicación de configuración para Android

La aplicación de configuración para Android desarrollada busca permitir la aplicación y edición de escenas de iluminación de una forma versátil. Por ello, la aplicación permitirá:

- Guardar hasta un máximo de 4 escenas, estáticas o dinámicas, en la memoria EEPROM externa.
- Aplicar de forma rápida cualquiera de las 4 escenas almacenadas en la memoria EEPROM.
- Generar escenas personalizadas.
- Modificar las escenas almacenadas en la EEPROM.
- Establecer de forma rápida una iluminación con las temperaturas de color más habituales.

Todas estas acciones permiten utilizar el sistema de una forma ágil, cómoda y versátil.

Como se ha mencionado anteriormente, una de las funcionalidades que se ha querido implementar en la aplicación es la posibilidad de cambiar la temperatura del color de la luminaria. Para ello se han programado unos botones específicos con las temperaturas de color más habituales, de forma que se pueda alternar entre ellas de forma rápida y sencilla.

En la Ilustración 41 se pueden ver los valores RGB que hay que aplicar a cada LED para obtener cada temperatura de color. De todas ellas, las más comunes son luz cálida (en torno a 2700K), luz fría (en torno a 4000K) y luz muy fría (en torno a 6500K), por lo que estos tres colores aparecerán ya programados en la aplicación de forma fija, para poder alternar entre ellos de forma rápida.







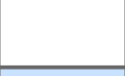
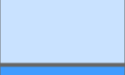

Light Source	Kelvin temperature	R G B Values	Color
Candle	1900	255, 147, 41	
40W Tungsten	2600	255, 197, 143	
100W Tungsten	2850	255, 214, 170	
Halogen	3200	255, 241, 224	
Carbon Arc	5200	255, 250, 244	
High Noon Sun	5400	255, 255, 251	
Direct Sunlight	6000	255, 255, 255	
Overcast Sky	7000	201, 226, 255	
Clear Blue Sky	20000	64, 156, 255	

Ilustración 41 Relación entre los valores RGB y la temperatura de color obtenida

Para los valores que no aparecen en la tabla de forma directa, se realizará una interpolación de los valores RGB, con el fin de aproximarse lo más posible al valor deseado de temperatura. Así los valores para las temperaturas definidas anteriormente para luz cálida, luz fría y luz muy fría se muestran en la Ilustración 41.

Descripción	Temperatura Kelvin	Valores RGB
Luz cálida	2700K	255, 204, 154
Luz fría	4000K	255,245,232
Luz muy fría	6500K	228, 240, 255

Tabla 3 Valores RGB de las temperaturas de color más comunes

Para el desarrollo de la aplicación, se han tenido en cuenta los conocimientos adquiridos durante este Máster con relación a la usabilidad de sistemas con interfaz gráfica con el usuario (GUI), lo que ha permitido realizar una aplicación sencilla y visual.

Algunas de las pautas de diseño que se han seguido durante el desarrollo de la aplicación han sido:

- Establecer pantallas sencillas, sin sobrecarga gráfica
- Aplicar acciones precisas a cada botón para minimizar la navegación del usuario.
- Establecer relación entre los colores de algunos elementos de la aplicación y los colores que tomarán realmente los LED de las luminarias.
- Tener en cuenta las diferentes resoluciones de pantalla que podemos encontrarnos en los dispositivos y colocar todo el contenido de forma coherente y que pueda ser visualizado sin problema en varias resoluciones (probada en HD, Full HD y QHD)
- Utilizar colores lógicos y reconocibles para las acciones de enviar datos o cancelar una acción, usando el verde para enviar y el rojo para cancelar.

- Utilizar, en la medida de lo posible iconos estándar y fácilmente reconocibles por el usuario para las diferentes acciones, como conectar el bluetooth, realizar ajustes o guardar valores en la memoria EEPROM.

Siguiendo estas pautas se ha conseguido desarrollar una aplicación funcional e intuitiva, cuya curva de aprendizaje es rápida y en la que el usuario es capaz de realizar cualquier acción en menos de 6 pulsaciones (incluyendo las necesarias para conectar con el dispositivo por bluetooth) y pasar por menos de 3 pantallas diferentes.

El diagrama de funcionamiento de la aplicación se puede ver en la Ilustración 42.

A continuación, se hará una breve descripción de la aplicación, explicando las principales pantallas y explicando aquellos aspectos que se consideren relevantes.

Para abrir la aplicación basta con pulsar sobre el icono de la misma, mostrado en la Ilustración 43. Se ha intentado que el icono sea bastante representativo de la funcionalidad de la aplicación, por lo que se ha puesto especial cuidado en este detalle. El icono elegido reúne varios elementos característicos y que nos ponen en adelanto cuál va a ser la función de la aplicación antes de abrirla. En este caso, la bombilla con la palabra LED en su interior hace referencia al tipo de luminaria empleada y el círculo cromático alrededor de la misma indica la posibilidad de cambiar el color de la iluminación. Al pulsar el icono se abrirá la pantalla del título. En ella podremos encontrar el logotipo de DALI y el botón “Comenzar” que cambia de color en función de la posición del móvil. Se ha intentado que este detalle haga de preámbulo antes de abrir la aplicación y permita intuir que se trata de una aplicación para cambiar de color las luminarias compatibles con DALI.

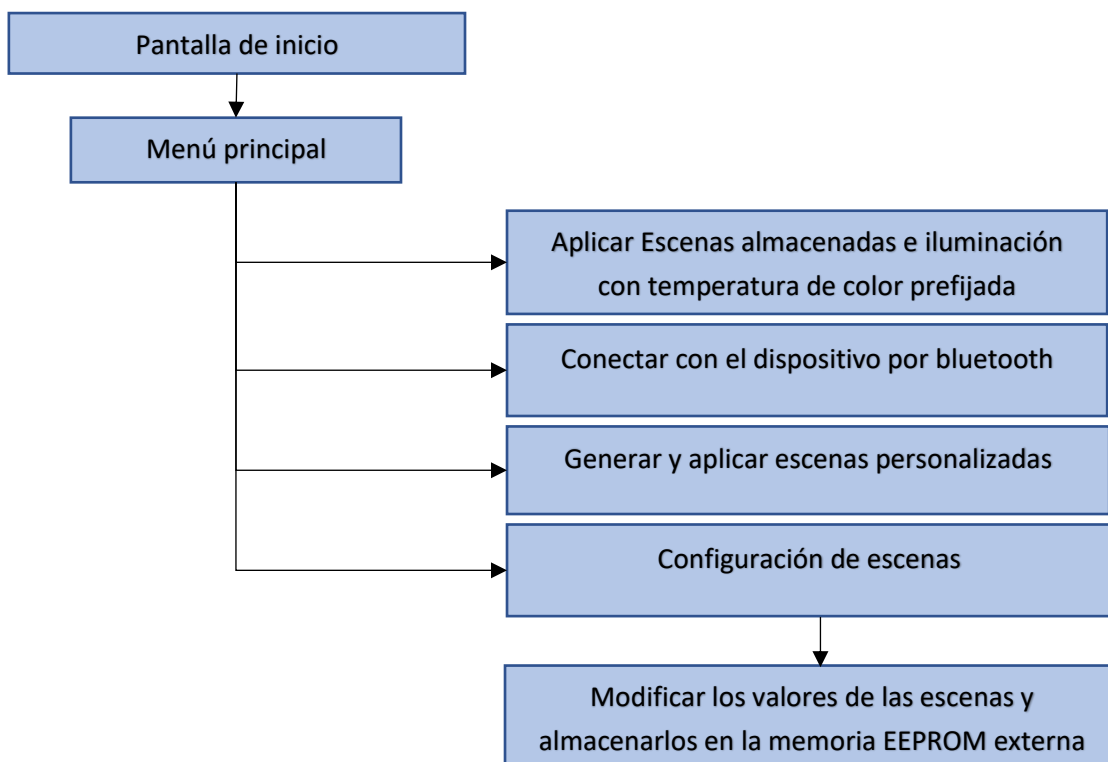


Ilustración 42 Diagrama de funcionamiento de la Aplicación para Android



Ilustración 43 icono de la aplicación desarrollada

La pantalla principal de la aplicación se puede ver en la Ilustración 44. En ella se destacan las principales acciones que se pueden llevar a cabo. En la parte superior nos permite aplicar cualquiera de las cuatro escenas grabadas en la EEPROM pulsando sobre el botón correspondiente a la escena que queremos aplicar. El icono de estos botones se ha establecido como un microchip, que hace referencia a la memoria EEPROM donde se almacenan las escenas.

Debajo de esta zona se encuentra el botón para crear escenas personalizadas. A este botón se le ha puesto un icono de una paleta de colores para dar a entender que el usuario puede aplicar el color que prefiera en esa opción.

Un poco más abajo encontramos las escenas prefijadas con las temperaturas de color más habituales, 2700K, 4000K y 6500K. Pulsando sobre cada uno de los botones se aplica directamente dicha escena.

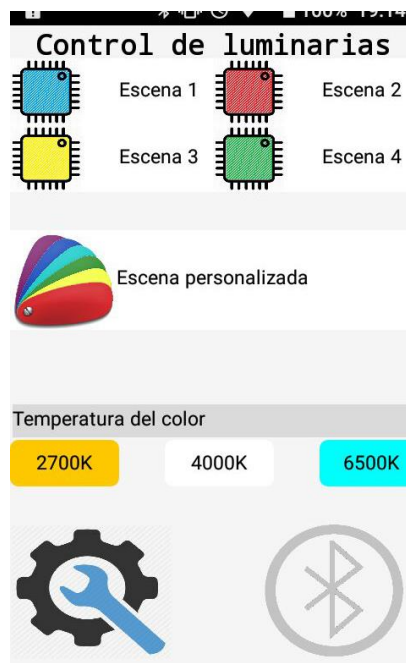


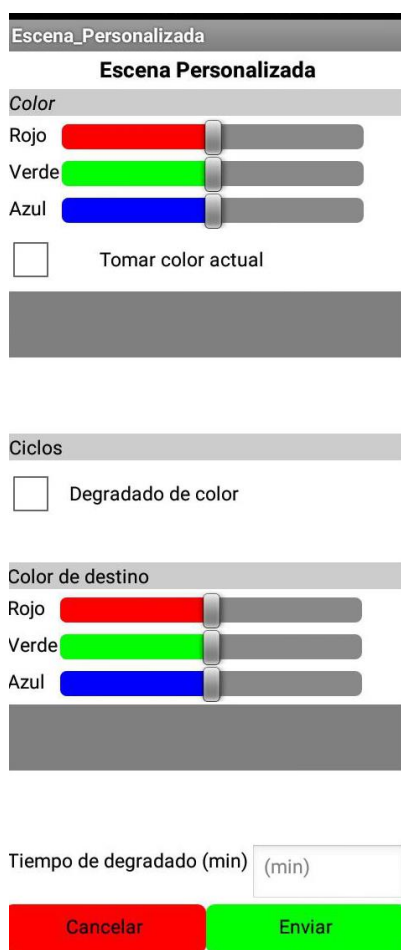
Ilustración 44 Pantalla principal de la aplicación

Para finalizar, en la parte inferior se disponen dos grandes botones redondos, a la izquierda el botón para configurar las escenas y a la derecha el botón de conexión por bluetooth, que cambiará a color azul cuando se haya establecido dicha comunicación.

Se ha querido destacar la agrupación de los elementos en la pantalla principal y para ello se ha establecido un fondo no blanco y sobre él las agrupaciones de botones en conjuntos blancos, de forma que a simple vista se diferencien las distintas posibilidades existentes en esta pantalla sin

necesidad de introducir muchos títulos. Se ha optado por introducir un título en la parte de las escenas predefinidas de temperatura de color para facilitar la comprensión del usuario de esos botones, ya que el texto de cada uno de ellos puede no resultar claro y explicativo para las personas que accedan por primera vez a la aplicación.

Si se accede a edición de escenas, ya sea por el botón de crear escena personalizada o bien por la configuración de cada una de las escenas que se almacenan en la EEPROM nos encontraremos con una como pantalla como la mostrada en la Ilustración 45. En ella se permite seleccionar el color de la luminaria a través de tres deslizadores que representan la cantidad de cada color presente en la luz resultante. Un valor más alto indica que el LED del color correspondiente brillará con una mayor intensidad mientras que el valor mínimo hace que el LED del color correspondiente se apague.



Escena_Personalizada

Escena Personalizada

Color

Rojo

Verde

Azul

Tomar color actual

Ciclos

Degradado de color

Color de destino

Rojo

Verde

Azul

Tiempo de degradado (min) (min)

Cancelar Enviar

Ilustración 45 Pantalla para configurar una escena

Debajo de los deslizadores hay una paleta de colores, que muestra el color aproximado ³que debe salir en la luminaria. También existe una casilla que permite mantener el color actual en la luminaria en caso de que se encuentre en una escena dinámica.

³ El color representado en la paleta de colores puede diferir en el mostrado realmente por la luminaria debido a la diferente representación de los colores en la pantalla del dispositivo y en la luminaria, ya que el dispositivo es capaz de mezclar el color negro para oscurecer tonos de color, algo que no es posible en la luminaria. Así todos los colores que tengan un contenido alto en negro se representarán en la paleta de colores del dispositivo de una manera más oscura a la realidad

La casilla de “Degradado de color” permite crear escenas dinámicas. En este tipo de escenas la iluminación variará desde el color superior hasta el color inferior en el tiempo especificado en el cuadro de texto llamado “Tiempo de degradado (min)”. Una vez alcanzado el color final, se realiza el recorrido inverso, volviendo del color final al inicial en el mismo tiempo, generando un ciclo infinito de cambio de color. Se puede detener el ciclo en cualquier momento marcando la casilla “Tomar color actual”.

Esta pantalla será la misma para guardar las escenas en la memoria EEPROM externa.

Finalmente, la última pantalla relevante es la de configuración, que nos permite almacenar escenas en la memoria no volátil para poder ser utilizadas más tarde y mantenerlas cuando se apague el dispositivo.

La pantalla que se verá al pulsar sobre el botón de configuración ubicado en la pantalla principal es la que aparece en la Ilustración 46. En ella se ha elegido un icono representativo para cada escena de forma que el usuario asocie con mayor facilidad que se va a guardar la escena que indique a continuación. Para cada escena se mantendrá el color que aparecía en la pantalla principal para facilitar que el usuario asocie cada escena con un color y lo recuerde de manera más sencilla.

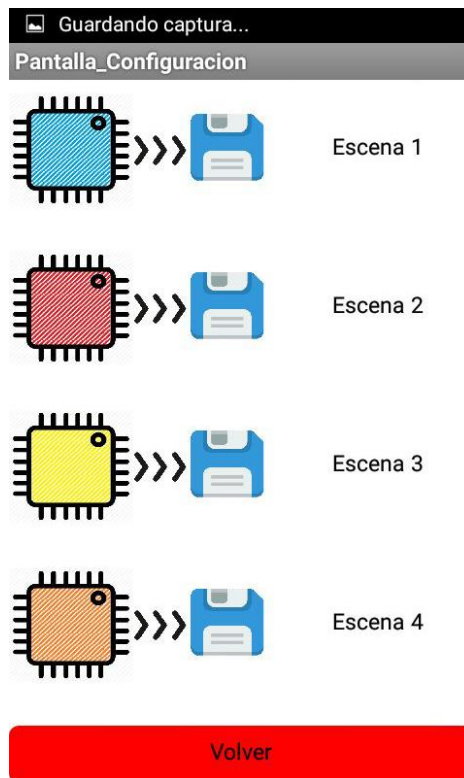


Ilustración 46 Pantalla de configuración de escenas

4.2 Funcionalidades del software de prueba desarrollado

El software de pruebas desarrollado e implementado en el microcontrolador Arduino, en este caso, permite realizar las siguientes funciones:

- Establecer escenas estáticas y dinámicas, como se ha visto en la explicación de la aplicación para Android.
- Establecer comunicación por bluetooth con un dispositivo móvil e interpretar adecuadamente la trama recibida.

- Realizar lecturas periódicas de los sensores conectados por I2C
- Almacenar en la memoria EEPROM los datos de las escenas indicadas.
- Generar un registro en la tarjeta microSD del funcionamiento del sistema que permita determinar posibles fallos y registre el estado de los sensores en cada momento, así como la escena aplicada.
- El sistema debe seguir funcionando con la última escena aplicada en caso de que se cierre la aplicación del móvil o se desconecte el bluetooth.
- Realizar la comunicación con el bus DALI para el control de las luminarias.

Todas estas funciones se pueden resumir en un diagrama de funcionamiento, que represente de forma clara y concisa las etapas por las que pasa el microcontrolador para realizar la ejecución de todas estas funciones.

La Ilustración 47 muestra un diagrama de las principales etapas por las que pasa el microcontrolador a lo largo de la ejecución del programa y que permiten implementar las funciones indicadas anteriormente.

Se tienen las siguientes etapas:

- Inicialización: En esta etapa se configuran los elementos del microcontrolador que vamos a utilizar (inicialización de puertos de comunicaciones y de entrada/salida, los canales del convertidor digital-analógico...)
- Espera de recepción de datos Bluetooth: En esta etapa se entra en el programa principal, que se ejecutará de forma cíclica hasta que el microcontrolador se apague. La primera tarea que realiza es comprobar si existen datos del bluetooth disponibles para leer. Los datos llegarán por el puerto serie correspondiente.
- Analiza e interpreta la trama: En caso de que se hayan recibido datos del módulo bluetooth, se almacenan en memoria y se interpreta la trama con el fin de determinar la acción que se quiere realizar.
- Realiza la acción recibida: En esta etapa, ya se tiene identificada la acción que hay que llevar a cabo y se realiza. Se distinguen varias acciones:
 - Cargar escenas predeterminadas
 - Guardar escenas
 - Aplicar escenas personalizadas

Cada una de estas acciones lleva implícita la generación de registros con el fin de realizar un seguimiento del funcionamiento del dispositivo.⁷

- Anota logs de funcionamiento: En esta etapa se graban los registros de funcionamiento en la tarjeta microSD en función de la acción ejecutada.
- Comprobar temporizadores: Realiza una consulta para comprobar si ha transcurrido el tiempo necesario para realizar una lectura de los sensores
- Tomar lectura de los sensores: En caso de haber transcurrido el tiempo indicado se realiza la lectura y se actualizan los valores de los sensores.
- Anotar valores en los logs: Se guarda en el fichero de registro de la tarjeta microSD los nuevos valores de los sensores.

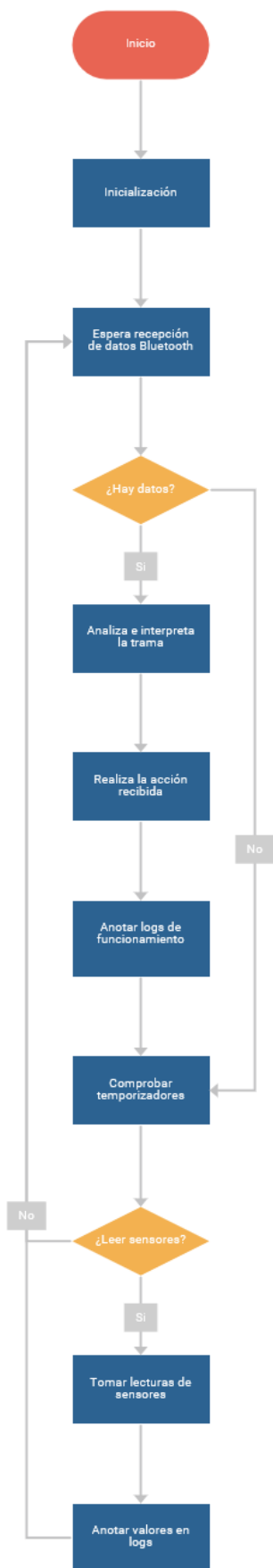


Ilustración 47 Diagrama básico de funcionamiento del programa

Este proceso se repetirá cíclicamente mientras el microcontrolador esté encendido.

A continuación, se describirá cada una de las funcionalidades indicadas anteriormente con un poco más de detalle, explicando cuál es el funcionamiento de cada una de ellas.

Establecimiento de escenas estáticas y dinámicas

Como ya se ha explicado brevemente en el apartado de descripción de la aplicación para Android, el sistema será capaz de generar escenas estáticas y dinámicas con un máximo de dos colores determinados.

Para el caso de las escenas estáticas, se establece un color fijo, que no varía con el tiempo hasta que el usuario aplique otra escena diferente.

Para el caso de las escenas dinámicas, el usuario establece dos colores, uno inicial y otro final, de forma que el color de la luminaria variará de forma uniforme y gradual desde el color inicial hasta el color final y viceversa, generando un ciclo continuo de cambio de color. El usuario podrá detener el color en cualquier momento o aplicar otra escena, tanto estática como dinámica. El tiempo que tarda en pasar del color inicial al color final lo establece el usuario, inicialmente en un rango de 1-9 minutos.

La aplicación de otra escena cuando se encuentra activa una escena dinámica anula la escena anterior y el color de la luminaria pasa inmediatamente al color inicial de la última escena aplicada, ya sea estática o dinámica.

Comunicación por bluetooth

El sistema dispone de un módulo Bluetooth HC-05 que permite conectarse con otros dispositivos. Para poder establecer una comunicación fiable ha sido necesario definir la trama del mensaje que van a intercambiar cliente y servidor, con el fin de poder transmitir e interpretar toda la información.

El sistema embebido funcionará como sistema maestro y a él se conectarán los dispositivos que quieran modificar el estado de las luminarias (esclavos).

A la hora de definir los campos que deberá tener la trama es necesario conocer qué parámetros se van a enviar desde el esclavo. Se definen así los siguientes datos para transmitir:

- El color de la escena que se aplica. Este color será el inicial en las escenas dinámicas.
- La información relativa a la casilla de "Tomar color actual" de la aplicación que permite detener el color de las escenas dinámicas.
- La información relativa a la casilla "Degradado" de la aplicación que permite establecer escenas dinámicas.
- El color final de las escenas dinámicas.
- El tiempo de degradado para las escenas dinámicas.

Junto a estos datos, será necesario establecer unos elementos de control, que permitan identificar lo siguiente:

- El tipo de escena que se está aplicando (personalizada, escena 1, escena2, escena3 o escena4)
- Si la escena se aplica o se guarda en la memoria EEPROM externa
- Carácter que indique que la trama ha finalizado.

En base a esto, se determina que son necesarios al menos 12 bytes para transmitir toda esta información y se establece la trama de los mensajes de la siguiente forma:

Cab. Control		DATOS									FIN
B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0

Donde:

B11: Es el primer byte de la trama. Indica la escena que se aplica y puede tomar los siguientes valores:

- 'P': Indica que es una escena personalizada y por ello no se guarda en la EEPROM.
- '1': Indica que la información siguiente corresponde a la escena 1
- '2': Indica que la información siguiente corresponde a la escena 2
- '3': Indica que la información siguiente corresponde a la escena 3
- '4': Indica que la información siguiente corresponde a la escena 4

B10: Es el segundo byte de la trama. Indica si la trama se aplica o se guarda en la memoria EEPROM. Puede tomar los siguientes valores:

- 'X': Indica que la trama no se guarda en la memoria EEPROM, sólo se aplica.
- 'E': Indica que la trama se guarda en la memoria EEPROM, en la posición reservada para escena correspondiente según el campo anterior.

Si la trama corresponde a una escena personalizada no se tendrá en cuenta el byte B10, ya que las escenas personalizadas no pueden almacenarse en la memoria EEPROM.

B9: Primer campo de datos de la trama. Este byte almacena información relativa a la casilla "Tomar color actual" de la aplicación. Indica si se detiene el color en las escenas dinámicas. Puede tomar los valores 0 o 1.

B8: Segundo campo de datos de la trama. Este byte almacena el valor del color rojo inicial. Puede tomar valores de 0 a 255.

B7: Tercer campo de datos de la trama. Este byte almacena el valor del color verde inicial. Puede tomar valores de 0 a 255.

B6: Cuarto campo de datos de la trama. Este byte almacena el valor del color azul inicial. Puede tomar valores de 0 a 255.

B5: Quinto campo de datos de la trama. Este byte almacena información relativa a la casilla "Degradado" de la aplicación. Indica si se va a aplicar una escena dinámica (1) o estática (0). Puede tomar los valores 0 o 1.

B4: Sexto campo de datos de la trama. Este byte almacena el valor del color rojo final para escenas dinámicas. Puede tomar valores de 0 a 255.

B3: Séptimo campo de datos de la trama. Este byte almacena el valor del color verde final para escenas dinámicas. Puede tomar valores de 0 a 255.

B2: Octavo campo de datos de la trama. Este byte almacena el valor del color azul final para escenas dinámicas. Puede tomar valores de 0 a 255.

Notar que los bytes B4-B2 sólo se tendrán en cuenta si B5 tiene el valor 1. En caso contrario se ignoran estos valores.

B1: Noveno campo de datos de la trama. Este byte almacena información relativa a la casilla “Tiempo de degradado” de la aplicación. Indica el número de minutos que tarda una escena dinámica en pasar del color inicial al final. En escenas estáticas se ignora este valor.

B0: Carácter especial que indica el final de la trama. Siempre toma el valor ‘Z’.

Lecturas periódicas de los sensores

El sistema realiza de manera periódica lecturas de los sensores que tiene conectados y almacena esta información en un fichero dentro de la tarjeta microSD.

El periodo de muestreo para cada sensor puede ser diferente y configurable por software.

En base a las lecturas de los sensores se podrán generar alarmas cuando, por ejemplo, se han dejado las luces encendidas y existe suficiente luz ambiental. También se pueden generar alarmas cuando existe un exceso de radiación infrarroja gracias al sensor de luz.

Almacenar en la memoria EEPROM las escenas configurables.

El sistema es capaz de almacenar en la memoria EEPROM hasta cuatro escenas configurables a través de la aplicación móvil. El sistema reserva un fragmento de memoria para cada una de las escenas según el mapa de memoria que se muestra en la Tabla 4.

Dirección de memoria	Contenido
0x00 00	B11 de la escena 1
0x00 01	B10 de la escena 1
...	...
0x00 0B	B0 de la escena 1
...	...
0x01 00	B11 de la escena 2
0x01 01	B10 de la escena 2
...	...
0x01 0B	B0 de la escena 2
...	...
0x02 00	B11 de la escena 3
0x02 01	B10 de la escena 3
...	...
0x02 0B	B0 de la escena 3
...	...
0x03 00	B11 de la escena 4
0x03 01	B10 de la escena 4
...	...
0x03 0B	B0 de la escena 4
...	...

Tabla 4 Mapa de memoria de la EEPROM

Se ha decidido dejar un espacio entre las posiciones de dos escenas consecutivas para permitir ampliar la trama en un futuro sin necesidad de realizar una reestructuración del contenido de la memoria EEPROM y permitir, dado el caso la retrocompatibilidad con el sistema ideado

actualmente, de forma que para añadir nuevos elementos a la trama de información bastaría con añadirlos detrás del carácter de finalización 'Z' y establecer al final de la nueva trama un nuevo carácter de finalización. De esta forma se pueden identificar fácilmente los datos antiguos y los añadidos posteriormente.

Registros en la tarjeta SD

El sistema dispone de una ranura para tarjetas microSD, lo que permite almacenar información de manera permanente, de forma similar a la memoria EEPROM pero además se puede acceder fácilmente a la información almacenada, ya que se encontrará en un fichero de texto dentro de la tarjeta.

Este medio de almacenamiento se empleará para generar registros donde se va añadiendo información acerca del funcionamiento del sistema como las acciones que se van realizando por parte del usuario, si ha cambiado la escena actual, si ha guardado una nueva configuración para alguna de las escenas etc.

Con el fin de disponer la información de manera más ordenada se generarán varios ficheros de texto en los que se irá almacenando periódicamente esta información. Así podemos encontrar los siguientes ficheros:

- Func.log
- Ambiente.log

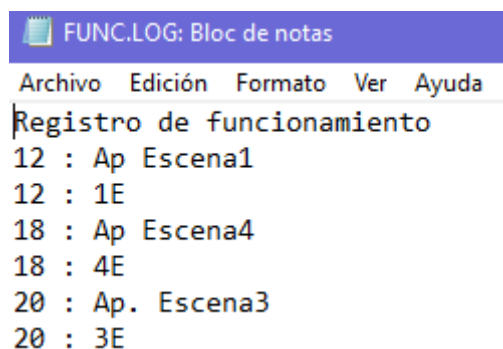
En el archivo "Func.log" se irán almacenando de manera periódica los siguientes datos:

Marca de tiempo	Acción realizada
-----------------	------------------

Tabla 5 Orden de los datos para el archivo Funcionamiento.log

La marca de tiempo servirá para determinar cuándo se realiza cada acción. En el caso de que el sistema no disponga de un reloj en tiempo real (RTC), se indicará el número de segundos transcurridos desde que se encendió la placa. De esta forma, además podemos conocer si se han producido reinicios inesperados ya que en ese caso la marca de tiempo se reiniciará.

A continuación de la marca de tiempo se indicará la acción realizada. Aquí se especifica el tipo de acción que se ha llevado a cabo, si se ha modificado alguna escena, se ha aplicado alguna escena, etc.



```
FUNC.LOG: Bloc de notas
Archivo Edición Formato Ver Ayuda
Registro de funcionamiento
12 : Ap Escena1
12 : 1E
18 : Ap Escena4
18 : 4E
20 : Ap. Escena3
20 : 3E
```

Ilustración 48 muestra del registro de funcionamiento

En el archivo "Ambiente.log" se almacena información recogida de los sensores. Los datos en este archivo se escriben de forma periódica, según lo configurado en el software y permitirá extraer conclusiones posteriormente. El formato de los datos se ha realizado de forma que

resulte sencillo importarlos posteriormente en un programa de cálculo para obtener, por ejemplo, representaciones gráficas de la evolución de la iluminancia, tanto infrarroja como visible a lo largo del tiempo o el color de la iluminación en cada instante.

El orden de los datos es como se muestra en la

Marca de tiempo	Iluminancia de luz visible e infrarroja	Iluminancia de luz infrarroja	Componente roja de la luz ambiente	Componente verde de la luz ambiente	Componente azul de la luz ambiente
-----------------	-----------------------------------------	-------------------------------	------------------------------------	-------------------------------------	------------------------------------

Tabla 6 Orden de los datos para el archivo Ambiente.log

Se ha realizado una prueba de adquisición y procesado de los datos, dejando el sistema funcionando durante 25 minutos aproximadamente y tomando medidas cada segundo, para posteriormente recoger la información generada en la tarjeta microSD y estudiar los datos.

Las condiciones de la prueba son las siguientes:

La prueba se ha realizado el día 21 de febrero en torno a las 17:00h en una habitación sin luz artificial y una ventana con cortina en el lado izquierdo del prototipo. Toda la iluminación que recibe el sistema proviene de esa fuente de luz natural y se provocó una obstrucción a la luz durante un breve periodo de tiempo al inicio de la prueba para luego verificar el comportamiento del sistema.

Tras realizar la prueba, el resultado obtenido en el archivo de texto “Ambiente.log” se muestra en la Ilustración 49. Aparece un breve encabezado que nos da información acerca de los datos para poder procesarlos de una forma más cómoda.

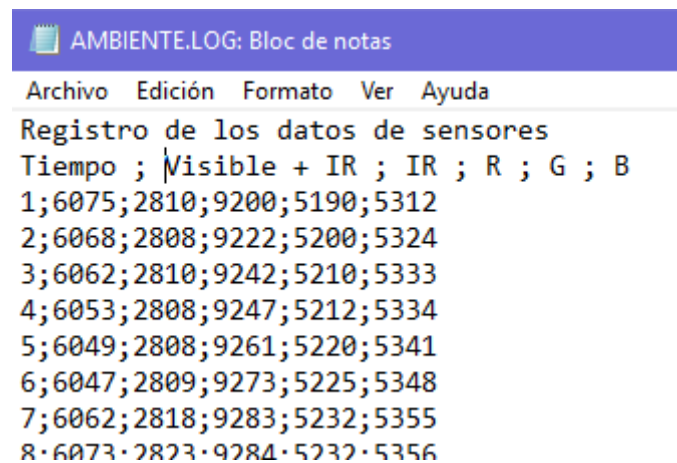


Ilustración 49 Aspecto de los datos en bruto generados por el sistema en el fichero AMBIENTE.LOG

Se ha utilizado el programa de cálculo Microsoft Excel para importar dichos datos y generar un gráfico que nos permita visualizar de forma más cómoda la información y extraer conclusiones.

Valores de iluminancia y color de la luz

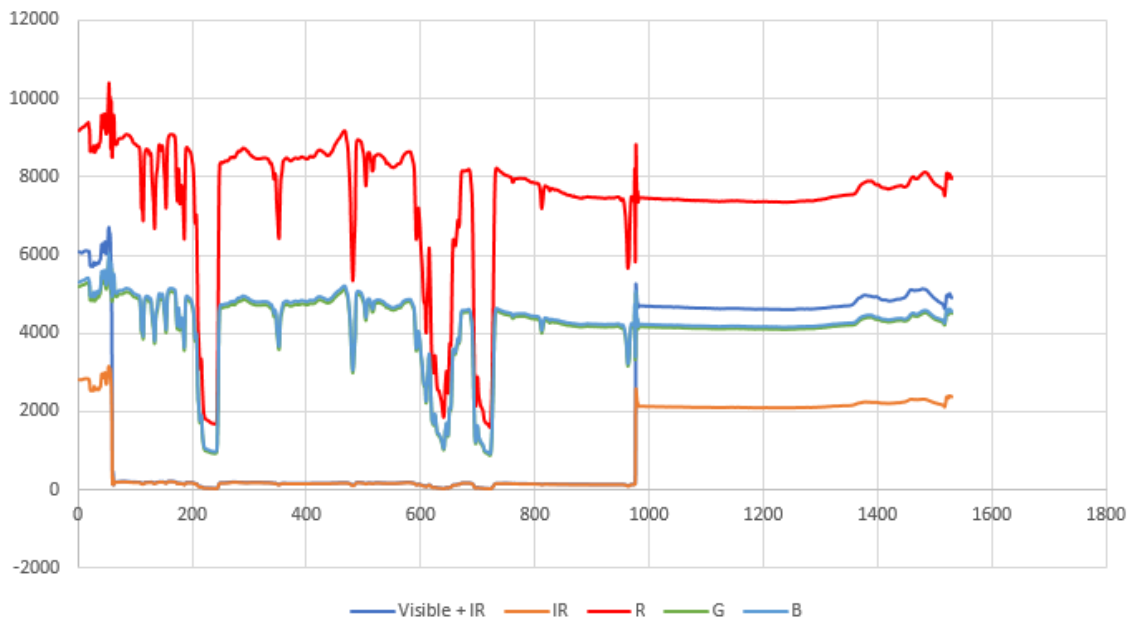


Ilustración 50 Gráfico que representa la iluminancia y el color de la luz en cada momento.

Del dicho gráfico (Ilustración 50) vemos que se produce una caída súbita en los valores correspondiente a iluminancia visible e infrarroja y únicamente infrarroja y posteriormente una recuperación también súbita a valores nominales, en los que luego parece que se estabiliza. Este periodo se corresponde al momento en el que se tapó el sensor de luz, obstruyendo así el paso de la luz. Como se puede apreciar, los valores de contenido rojo, verde y azul de la luz permanecen casi inalterados durante ese periodo, ya que son captados por otro receptor sobre el cual no se aplicó ninguna obstrucción. Los picos de bajada que tienen los valores correspondientes al color de la luz se pueden deber a momentos en los que se nubló el cielo, ya que han tenido un impacto en la iluminancia visible e infrarroja, aunque en menor medida por estar obstruida.

Del análisis de la cantidad de color en la luz ambiente se puede determinar que la luz era cálida, ya que hay un gran contenido de luz roja en el ambiente, algo lógico ya que a la hora de realizar la prueba nos encontrábamos en el atardecer, momento en el que la luz ambiente es cada vez más cálida.

Autonomía del sistema

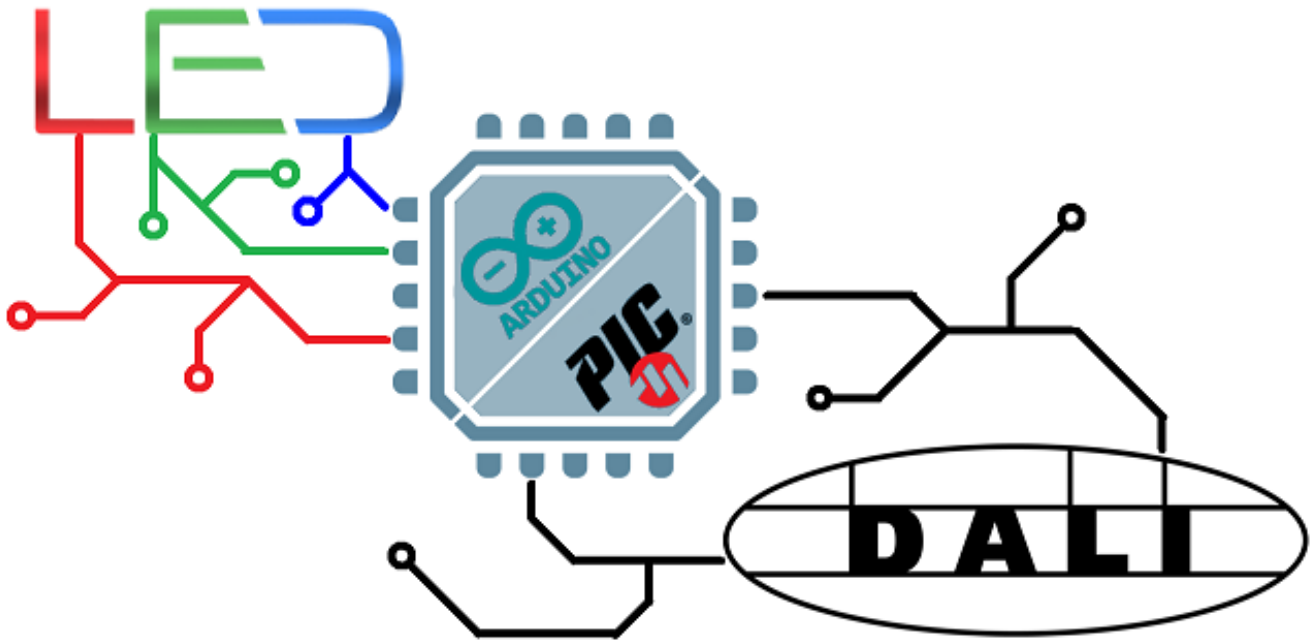
El sistema es capaz de seguir funcionando cuando se pierda la comunicación con el dispositivo bluetooth. Esto permite que se pueda desvincular el teléfono del sistema una vez que se han enviado las órdenes deseadas para ahorrar batería. Esto se puede conseguir gracias al almacenamiento en la memoria EEPROM de las escenas, de forma que el dispositivo remoto no necesita enviar de forma continua los colores por los que debe ir pasando en las escenas dinámicas.

Comunicación con el bus DALI

El sistema es capaz de comunicarse con dispositivos conectados a un bus DALI. Esto permite controlar de forma independiente hasta 64 luminarias modificando la intensidad o color de las mismas. Con el fin de dotar al sistema de una mayor versatilidad, la implementación de las

funciones que realizan este control también son compatibles con dispositivos LED RGB que cuenten con cuatro pines. De esta forma, se amplía el funcionamiento no sólo a luminarias DALI sino también a otro tipo de luminarias más genéricas.

Esto ha sido posible ya que el número de bits que definen la intensidad de un color en DALI es el mismo que el utilizado por el generador PWM de Arduino, que ha sido el microcontrolador utilizado para el prototipo. Aprovechando esta característica, se han implementado las funciones compatibles con DALI como son las funciones “step_up” y “step_down” que permiten aumentar o disminuir la intensidad de un determinado color y “establecer_color” que permite fijar un color determinado en la luminaria DALI empleando los valores RGB en un rango de 0 a 254, el mismo que se emplea en Arduino para iluminar un LED RGB por señal PWM. La única característica que se ha tenido que tener en cuenta para garantizar la compatibilidad ha sido la de limitar el valor máximo de iluminación empleando PWM a 254, eliminando el valor 255 porque en el protocolo DALI dicho valor se emplea para hacer un broadcast de apagado, por lo que el máximo valor que se alcanzará con señal PWM será de 254 en lugar de 255.

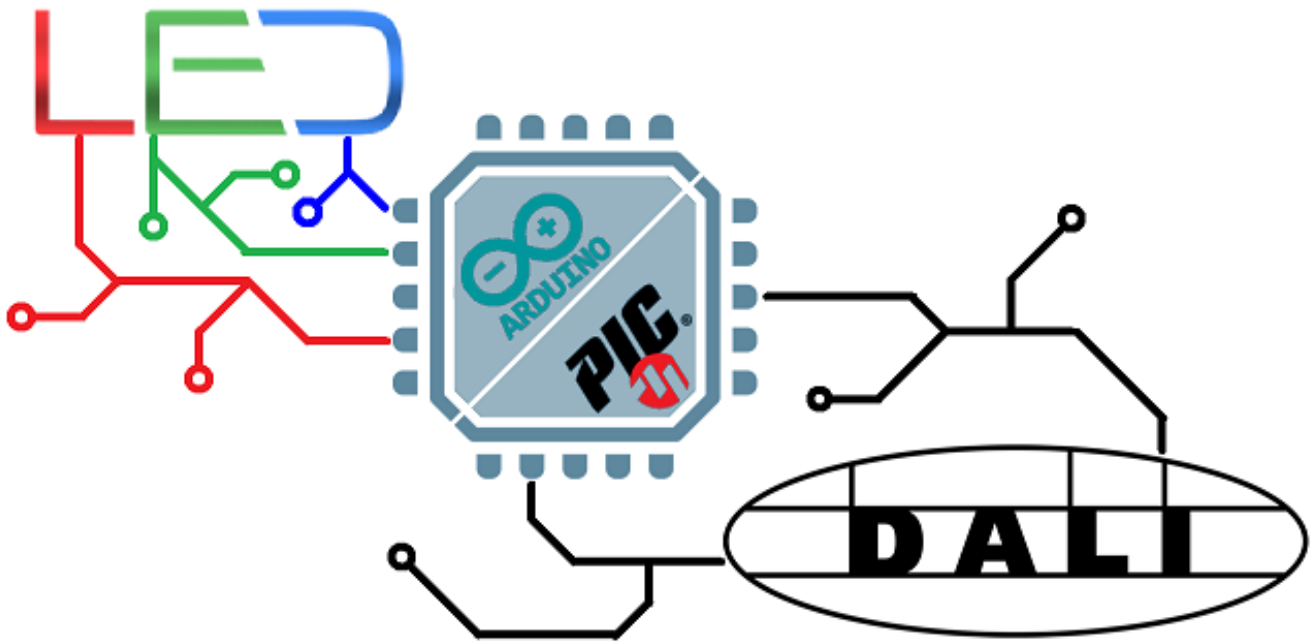


5. Conclusiones

Finalmente, evaluando el resultado, se puede concluir que el trabajo ha cumplido los objetivos planteados inicialmente ya que se ha finalizado el desarrollo de una placa de circuito impreso modular, que permite no sólo la interconexión de diferentes módulos que sigan el estándar MikroBus, también permite intercambiar el microcontrolador entre una serie de ellos compatibles entre los que podemos encontrar el Arduino Nano o el PIC18F87K22 empleado en el proyecto que sirve de antecedente a este. La placa desarrollada cuenta además con un zócalo MikroBus L en el que se puede conectar el módulo DALI click, que permite la conexión del sistema a un bus DALI lo cual era otro de los objetivos planteados inicialmente en el desarrollo del presente trabajo.

Se ha conseguido también agregar valor al producto y añadiendo algunas características interesantes con un coste bastante bajo. Algunas de las características que se han desarrollado han sido las siguientes:

- Se ha establecido una comunicación con sensores en módulos MikroBus a través de un bus I2C lo que permite conectar múltiples sensores en un único bus de comunicaciones.
- Se ha conseguido recoger y almacenar datos de los sensores para poder ser exportados de forma fácil a través de la tarjeta micro-SD y poder ser procesados en hojas de cálculo de manera sencilla gracias al formato empleado.
- Se ha establecido una comunicación bluetooth con un dispositivo Android para que se pueda modificar el color o intensidad de la luz y aplicar escenas de forma inalámbrica.
- Se ha conseguido que el sistema sea compatible con algunos de los microcontroladores más extendidos actualmente como son los PIC y los ATmega con la placa de desarrollo Arduino Nano.
- Se ha añadido la posibilidad controlar luminarias LED RGB tradicionales sin necesidad de implementar un bus DALI.
- Se ha conseguido desarrollar una aplicación sencilla para dispositivos Android que permita el control de las luminarias vía bluetooth.
- Se ha diseñado una envolvente para contener el sistema electrónico desarrollado atendiendo a pautas de diseño para poder ser fabricada con impresión 3D.
- Se ha conseguido mirar gran parte del software desarrollado en el trabajo antecedente para microcontroladores PIC a la plataforma Arduino y se han realizado los correspondientes test de funcionamiento dando éstos un resultado satisfactorio.

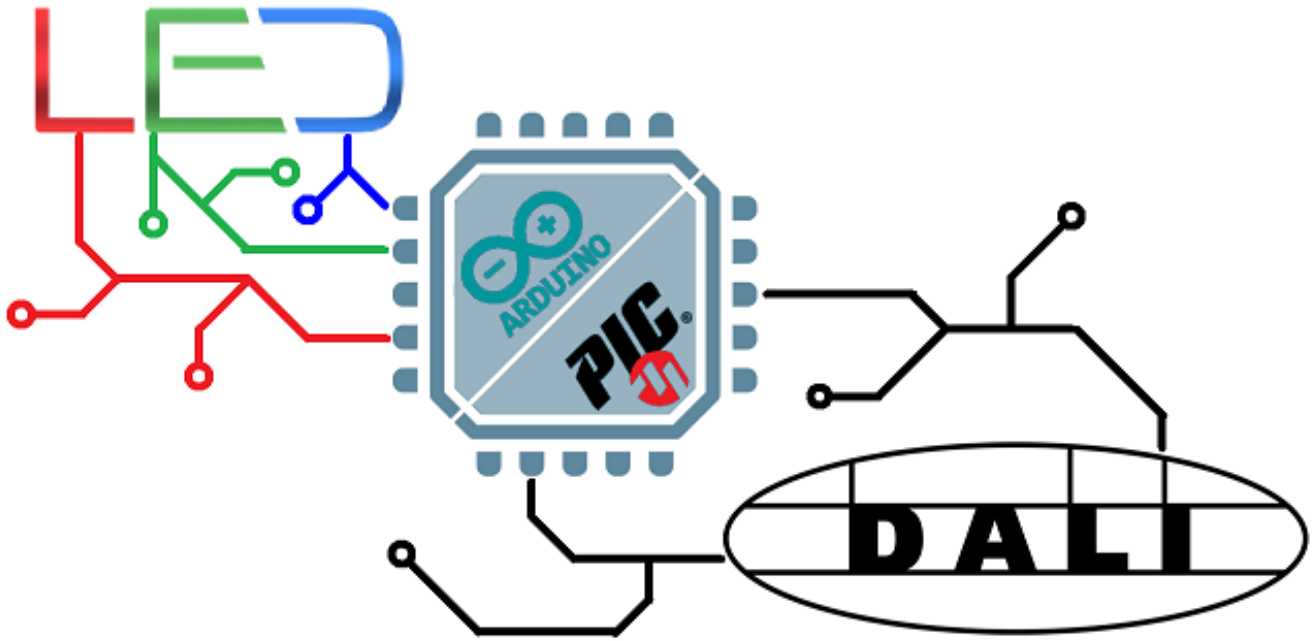


6. líneas futuras

Finalmente se comentarán algunas líneas futuras de mejora que se pueden implementar en este sistema para completarlo y añadirle mejoras y funcionalidades. Algunas de estas mejoras son las siguientes:

- Aumentar el número de colores por lo que pasa un ciclo. Actualmente se han implementado los ciclos de dos colores, uno inicial y otro final. Podría ser interesante implementar varios colores por ciclos, de forma que se tenga un mayor control en el degradado de los mismos y los colores por los que transcurre el ciclo. De esta forma se pueden establecer ciclos que vayan pasando por una serie de colores muy concretos, que no es posible realizarlo actualmente al contar sólo con dos puntos de control.
- Añadir un puerto de expansión I2C a la placa para poder conectar más dispositivos. Actualmente la placa desarrollada cuenta con varios sensores conectados a un bus I2C, pero tiene la limitación de poder conectar únicamente tres módulos. Si bien se pueden crear módulos personalizados que incluyan varios sensores dentro de cada uno y poder aumentar el número de sensores conectados al bus, no elimina la limitación de estar físicamente en el mismo lugar que el controlador, por lo que sería interesante añadir un puerto de expansión I2C que permita conectar sensores externos a este bus. De esta forma, se podrían colocar en una ubicación diferente al controlador, para realizar medidas en otras estancias, por ejemplo.
- Añadir un campo de Checksum a la trama de Bluetooth para incrementar la fiabilidad de los mensajes. Actualmente la comunicación bluetooth no utiliza mecanismo de control de integridad del mensaje. Esto hace que puedan llegar mensajes corruptos que el microcontrolador tratará de analizar. Para evitar esto, se pueden añadir mecanismos de control de integridad de la trama que permitan determinar, antes de analizar todo el mensaje si éste es correcto o se ha dañado durante la transmisión. La implementación de unos bits de “checksum” podrían ser un primer mecanismo sencillo.
- Añadir cifrado a las tramas bluetooth para incrementar la seguridad. En la implementación actual no se tiene ninguna capa de seguridad que pueda impedir el acceso no autorizado al sistema, pudiendo enviar mensajes todo el que sepa el protocolo utilizado. A esto se añade la inexistencia de algún tipo de cifrado de la información transmitida, lo que haría relativamente sencillo que alguien con suficientes conocimientos en el tema pueda “escuchar” los mensajes enviados y deducir los campos de la trama para posteriormente acceder al sistema y enviar mensajes.
- Añadir un control en lazo cerrado para iluminación y color. Gracias a que se dispone de un sensor de color y de iluminancia, sería posible realizar un control en lazo cerrado de la iluminación. Una vez establecidos el color e intensidad de la luz deseada en el ambiente, agregar una función que de forma inteligente adapte el color e intensidad de la luminaria para compensar posibles perturbaciones provenientes de otras fuentes (luz natural, luz de edificios colindantes, etc) para lograr mantener en el interior las especificaciones dadas inicialmente.
- Incluir un reloj en tiempo real para tener marca temporal en los logs y poder establecer alarmas en función de la hora del día. Como no se disponía de un reloj en tiempo real (RTC) en el diseño implementado, se emplea como marca temporal para los registros el número de segundos que lleva encendida la placa. Esto tiene varios inconvenientes como son los siguientes:
 - No se tiene control sobre la hora del día en el que se están almacenando los registros.
 - No se sabe el día de la semana al que corresponden los datos guardados.

- Se producen desbordamientos periódicos que dan lugar a discontinuidades temporales en los datos, lo que dificulta el tratamiento masivo de los datos. Si se conecta un reloj en tiempo real al sistema se podrían sustituir las marcas temporales por el día y la hora a las que se han realizado los registros evitando así los problemas indicados anteriormente.
- Permitir la programación de encendido/apagado de las luces, necesario un RTC para saber la hora del día. En relación con el punto anterior, si se dispone de un RTC sería posible realizar la programación del sistema en base a días, horas e incluso estaciones. También sería posible determinar los días festivos y poder realizar programaciones discriminando días laborales de días festivos.



7. Bibliografía

Sanz, E. (2017). 6 efectos de la luz azul que deberías conocer. Muy Interesante. Retrieved 27 February 2018, from <https://www.muyinteresante.es/salud/articulo/6-efectos-de-la-luz-azul-que-deberias-conocer>

Cortés Parra, C., Figueroa Torres, J., Montoya Lavrde, D., & Piedrahita Calderón, L. (2013). Efectos psicológicos de la luz. Último acceso: 27 de Febrero.

Miočinović, D., Rešetar, I., & Katona, M. (2011). Development of modern embedded electronics (PCB) for a small-volume production - IEEE Conference Publication. IEEE. Retrieved 27 February 2018, from <http://ieeexplore.ieee.org/document/5733896/?part=1>

PIC MCU Cards - 80-pin Microcontroller Cards for BIGPIC Development System. Mikroee.com. Retrieved 27 February 2018, from <https://www.mikroe.com/bigpic5-pic18f8520>

EasyPIC PRO v7 Empty MCUCard 80pin TQFP. Retrieved from <https://www.mikroe.com/easypic-pro-v7-80-pin-tqfp-empty>

Abdul Amsyar Bin Hj, M., & Scott, R. DALI – THE DIGITAL ADDRESSABLE LIGHTING INTERFACE. University of Glasgow.

Digitally Addressable Designer Reference Manual Lighting Interface (DALI) Unit Using the MC68HC908KX8. (2002). Nxp. Retrieved 27 February 2018, from <https://www.nxp.com/docs/en/reference-manual/DRM004.pdf>

J. Ribarich, T., & Contenti, C. (2002). Analog and Digital Fluorescent Lighting Dimming Systems. International Rectifier Technical Paper. Retrieved 27 February 2018, from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.5888&rep=rep1&type=pdf>

Diario de León. (2005). La luz influye directamente en la salud, los biorritmos y el sueño. Retrieved from http://www.diariodeleon.es/noticias/leon/luz-influye-directamente-salud-biorritmos-sueno_198265.html

Digital Addressable Lighting Interface (DALI) Implementation Using MSP430 Value Line Microcontrollers. (2009). Texas Instrument Application Report. Retrieved 27 February 2018, from <http://www.ti.com/lit/an/slaa422a/slaa422a.pdf>

Dali manual. (2013). Tridonic. Retrieved 27 February 2018, from http://www.tridonic.se/it/download/technical/DALI-manual_en.pdf

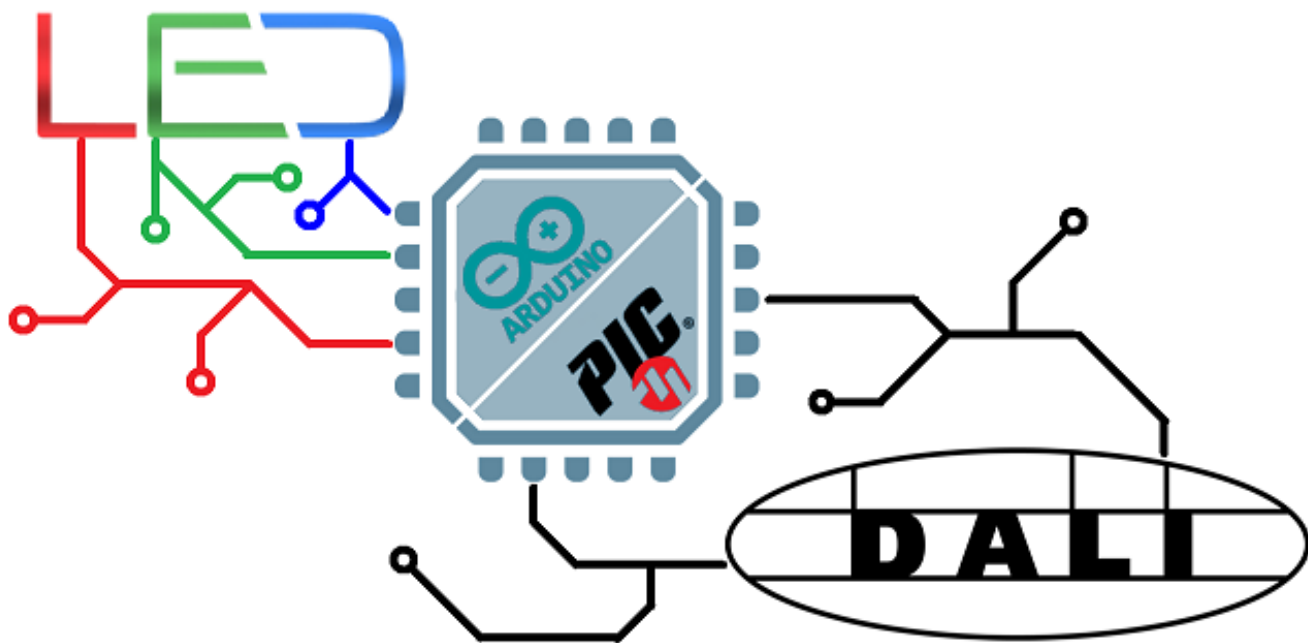
Dali. (2001). DALI manual.

AN10760 USB - DALI master using the LPC2141. (2008). Nxp Application Note. Retrieved 27 February 2018, from <https://www.nxp.com/docs/en/application-note/AN10760.pdf>

CorelDRAW X6 Help. (2012). CorelDRAW X6 Help. Retrieved 27 February 2018, from http://product.corel.com/help/CorelDRAW/540240626/Main/ES/Doc/wwhelp/wwhimpl/common/html/wwhelp.htm?context=CorelDRAW_Help&file=CorelDRAW-Understanding-color-models.html

Easy PIC Pro v7. (2013). Mikroelektronika. Retrieved 27 February 2018, from <https://download.mikroe.com/documents/full-featured-boards/easy/easypic-pro-v7/easypic-pro-v7-manual-v101.pdf>

Generic Standard on Printed Board Design IPC2221B(2012). IPC. Disponible en:
<http://www.ipc.org/TOC/IPC-2221B.pdf>



Anexos

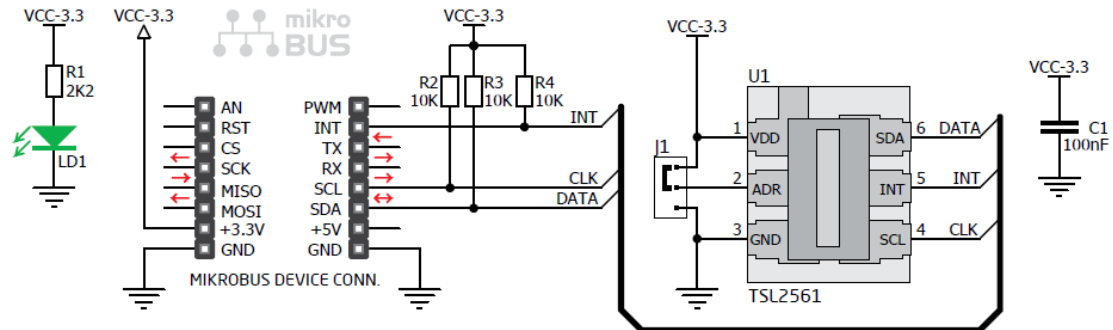
ANEXO I. Tabla de comandos DALI

Number	Command Code	Repeat < 100 ms	Answer Slave	Command Name
-	YAAA AAA0 XXXX XXXX	no	no	DIRECT ARC POWER CONTROL
0	YAAA AAA1 0000 0000	no	no	OFF
1	YAAA AAA1 0000 0001	no	no	UP
2	YAAA AAA1 0000 0010	no	no	DOWN
3	YAAA AAA1 0000 0011	no	no	STEP UP
4	YAAA AAA1 0000 0100	no	no	STEP DOWN
5	YAAA AAA1 0000 0101	no	no	RECALL MAX LEVEL
6	YAAA AAA1 0000 0110	no	no	RECALL MIN LEVEL
7	YAAA AAA1 0000 0111	no	no	STEP DOWN AND OFF
8	YAAA AAA1 0000 1000	no	no	ON AND STEP UP
9-15	YAAA AAA1 0000 1XXX			RESERVED
16 - 31	YAAA AAA1 0001 XXXX	no	no	GO TO SCENE
32	YAAA AAA1 0010 0000	yes	no	RESET
33	YAAA AAA1 0010 0001	yes	no	STORE ACTUAL LEVEL IN THE DTR
34 - 41	YAAA AAA1 0010 XXXX			RESERVED
42	YAAA AAA1 0010 1010	yes	no	STORE THE DTR AS MAX LEVEL
43	YAAA AAA1 0010 1011	yes	no	STORE THE DTR AS MIN LEVEL
44	YAAA AAA1 0010 1100	yes	no	STORE THE DTR AS SYSTEM FAILURE LEVEL
45	YAAA AAA1 0010 1101	yes	no	STORE THE DTR AS POWER ON LEVEL
46	YAAA AAA1 0010 1110	yes	no	STORE THE DTR AS FADE TIME
47	YAAA AAA1 0010 1111	yes	no	STORE THE DTR AS FADE RATE
48 - 63	YAAA AAA1 0011 XXXX			RESERVED
64 - 79	YAAA AAA1 0100 XXXX	yes	no	STORE THE DTR AS SCENE
80 - 95	YAAA AAA1 0101 XXXX	yes	no	REMOVE FROM SCENE
96 - 111	YAAA AAA1 0110 XXXX	yes	no	ADD TO GROUP
112 -127	YAAA AAA1 0111 XXXX	yes	no	REMOVE FROM GROUP
128	YAAA AAA1 1000 0000	yes	no	STORE DTR AS SHORT ADDRESS
129 -143	YAAA AAA1 1000 XXXX			RESERVED
144	YAAA AAA1 1001 0000	no	yes	QUERY STATUS
145	YAAA AAA1 1001 0001	no	yes	QUERY BALLAST
146	YAAA AAA1 1001 0010	no	yes	QUERY LAMP FAILURE
147	YAAA AAA1 1001 0011	no	yes	QUERY LAMP POWER ON
148	YAAA AAA1 1001 0100	no	yes	QUERY LIMIT ERROR
149	YAAA AAA1 1001 0101	no	yes	QUERY RESET STATE
150	YAAA AAA1 1001 0110	no	yes	QUERY MISSING SHORT ADDRESS
151	YAAA AAA1 1001 0111	no	yes	QUERY VERSION NUMBER
152	YAAA AAA1 1001 1000	no	yes	QUERY CONTENT DTR
153	YAAA AAA1 1001 1001	no	yes	QUERY DEVICE TYPE
154	YAAA AAA1 1001 1010	no	yes	QUERY PHYSICAL MINIMUM LEVEL
155	YAAA AAA1 1001 1011	no	yes	QUERY POWER FAILURE
156 - 159	YAAA AAA1 1001 11XX			RESERVED
160	YAAA AAA1 1010 0000	no	yes	QUERY ACTUAL LEVEL
161	YAAA AAA1 1010 0001	no	yes	QUERY MAX LEVEL
162	YAAA AAA1 1010 0010	no	yes	QUERY MIN LEVEL

163	YAAA AAA1 1010 0011	no	yes	QUERY POWER ON LEVEL
164	YAAA AAA1 1010 0100	no	yes	QUERY SYSTEM FAILURE LEVEL
165	YAAA AAA1 1010 0101	no	yes	QUERY FADE TIME / FADE RATE
166 - 175	YAAA AAA1 1010 XXXX			RESERVED
176 - 191	YAAA AAA1 1011 XXXX	no	yes	QUERY SCENE LEVEL (SCENES 0-15)
192	YAAA AAA1 1100 0000	no	yes	QUERY GROUPS 0-7
193	YAAA AAA1 1100 0001	no	yes	QUERY GROUPS 8-15
194	YAAA AAA1 1100 0010	no	yes	QUERY RANDOM ADDRESS (H)
195	YAAA AAA1 1100 0011	no	yes	QUERY RANDOM ADDRESS (M)
196	YAAA AAA1 1100 0100	no	yes	QUERY RANDOM ADDRESS (L)
197 - 223	YAAA AAA1 110X XXXX			RESERVED
224 - 255	YAAA AAA1 11XX XXXX			APPLICATION EXTENDED COMMANDS
256	1010 0001 0000 0000	no	no	TERMINATE
257	1010 0011 XXXX XXXX	no	no	DATA TRANSFER REGISTER (DTR)
258	1010 0101 XXXX XXXX	yes	no	INITIALISE
259	1010 0111 0000 0000	yes	no	RANDOMISE
260	1010 1001 0000 0000	no	yes	COMPARE
261	1010 1011 0000 0000	no	no	WITHDRAW
262	1010 1101 0000 0000			RESERVED
263	1010 1111 0000 0000			RESERVED
264	1011 0001 HHHH HHHH	no	no	SEARCHADDRH
265	1011 0011 MMMM MMMM	no	no	SEARCHADDRM
266	1011 0101 LLLL LLLL	no	no	SEARCHADDRL
267	1011 0111 0AAA AAA1	no	no	PROGRAM SHORT ADDRESS
268	1011 1001 0AAA AAA1	no	yes	VERIFY SHORT ADDRESS
269	1011 1011 0000 0000	no	yes	QUERY SHORT ADDRESS
270	1011 1101 0000 0000	no	no	PHYSICAL SELECTION
271	1011 1111 XXXX XXXX			RESERVED
272	1100 0001 XXXX XXXX	no	no	ENABLE DEVICE TYPE X
273 - 287	110X XXX1 XXXX XXXX			RESERVED

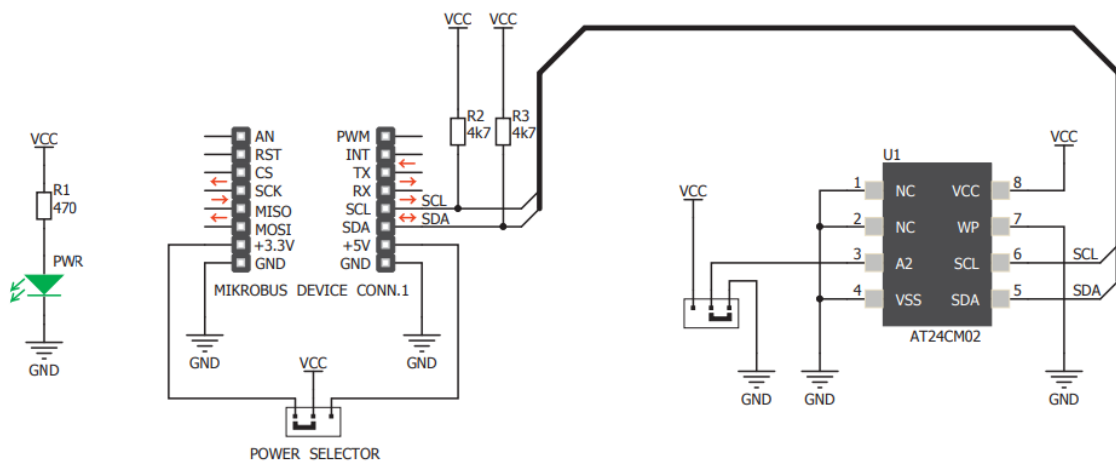
ANEXO II. Hojas de datos de los sensores empleados

Esquema del módulo "illuminance click"



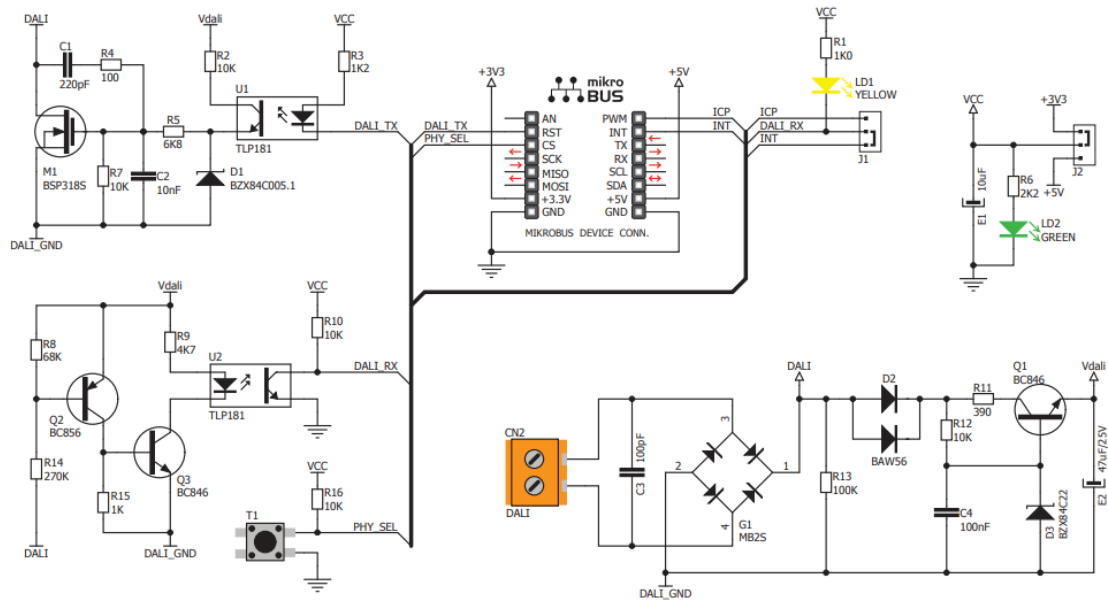
Datasheet del componente TSL2561 adjunto a este documento

Esquema del módulo "EEPROM click"

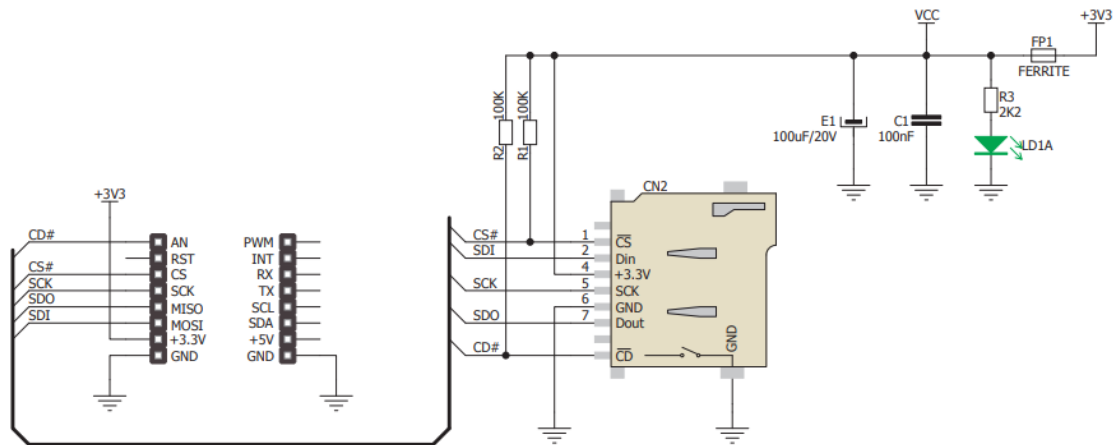


Datasheet del componente AT24CM02 adjunto a este documento

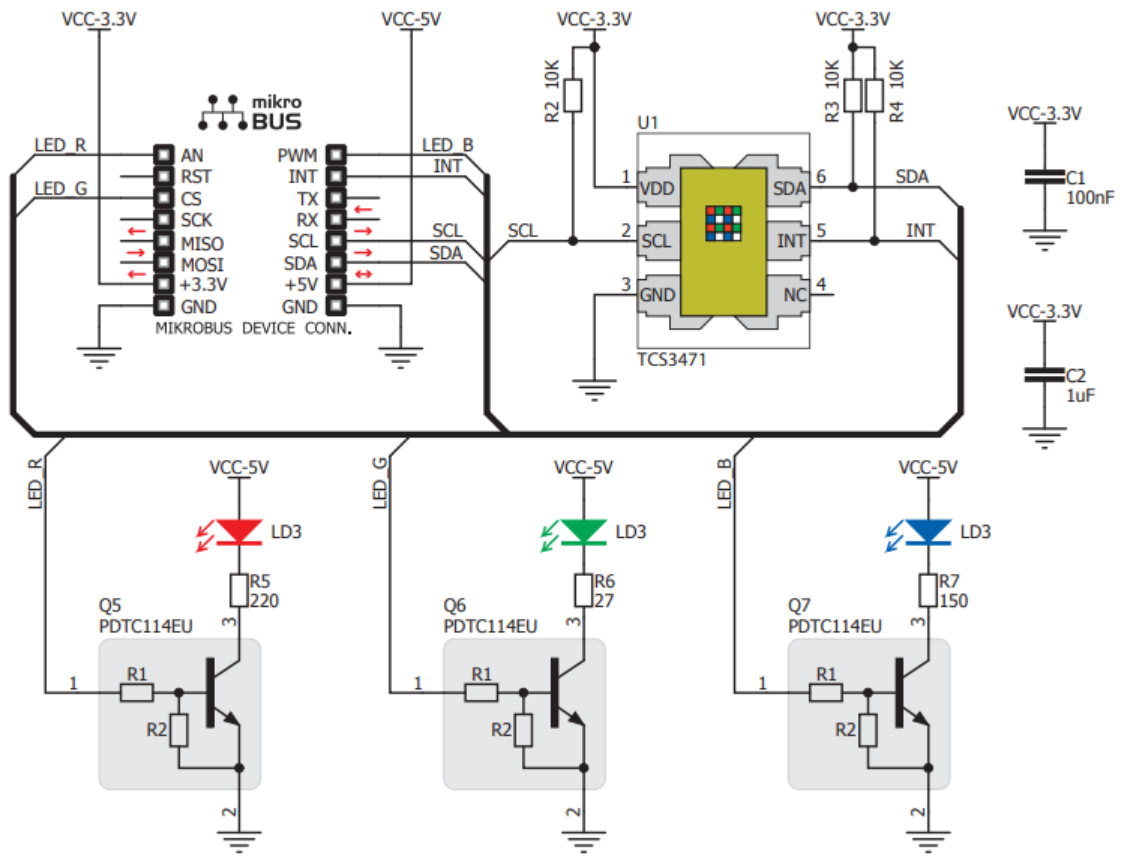
Esquema del módulo "DALI click"



Esquema del módulo "microSD click"

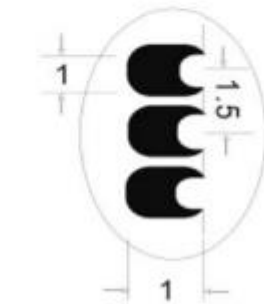
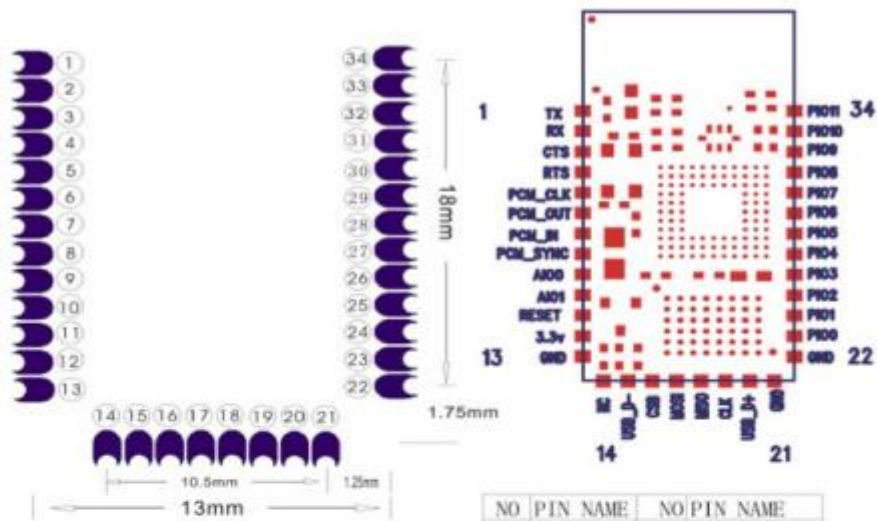


Esquema del módulo "color click"



Datasheet del componente TCS3471 disponible adjunto a este documento.

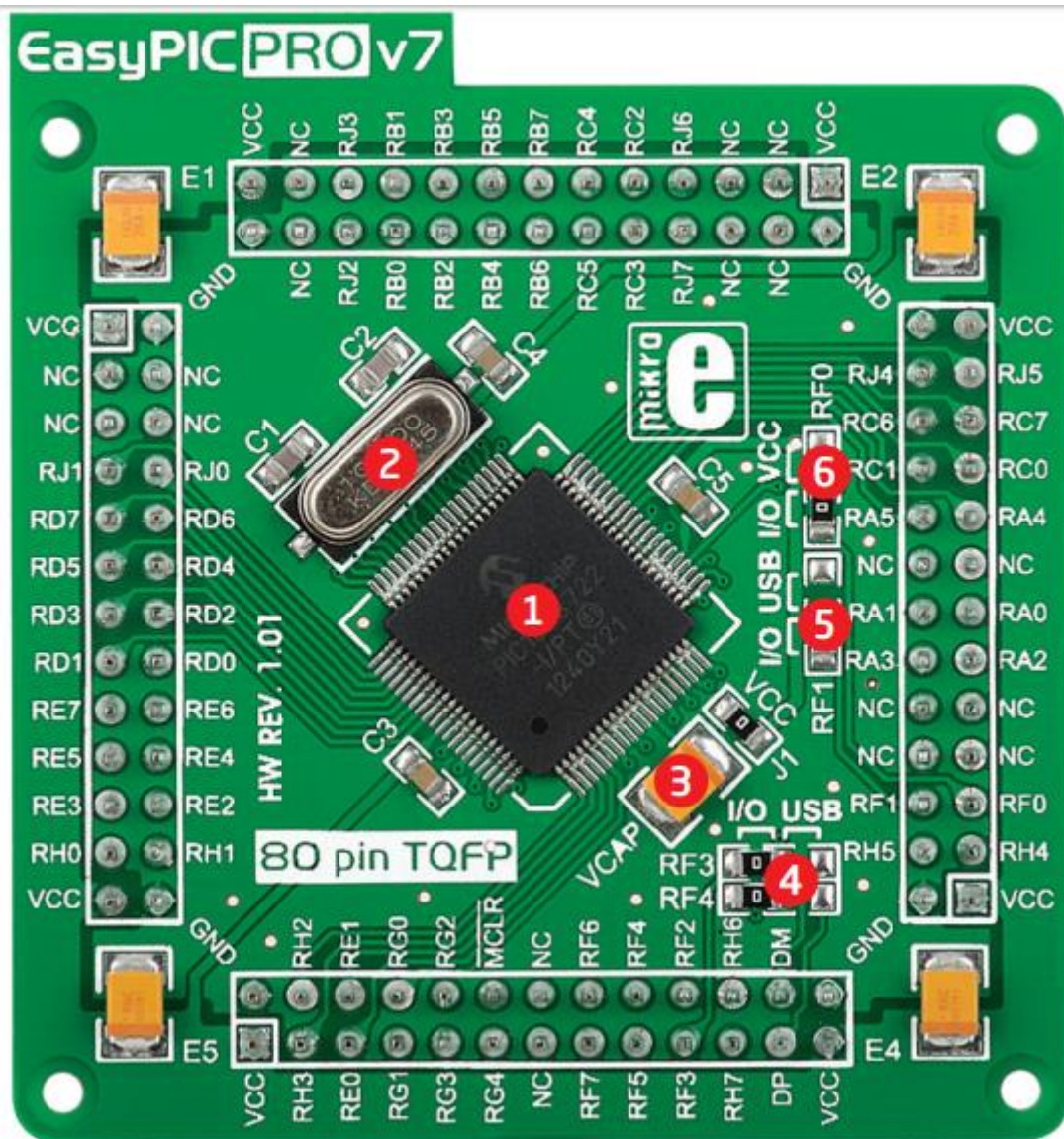
Pinout del módulo HC05



PCB Layout 请参考实物

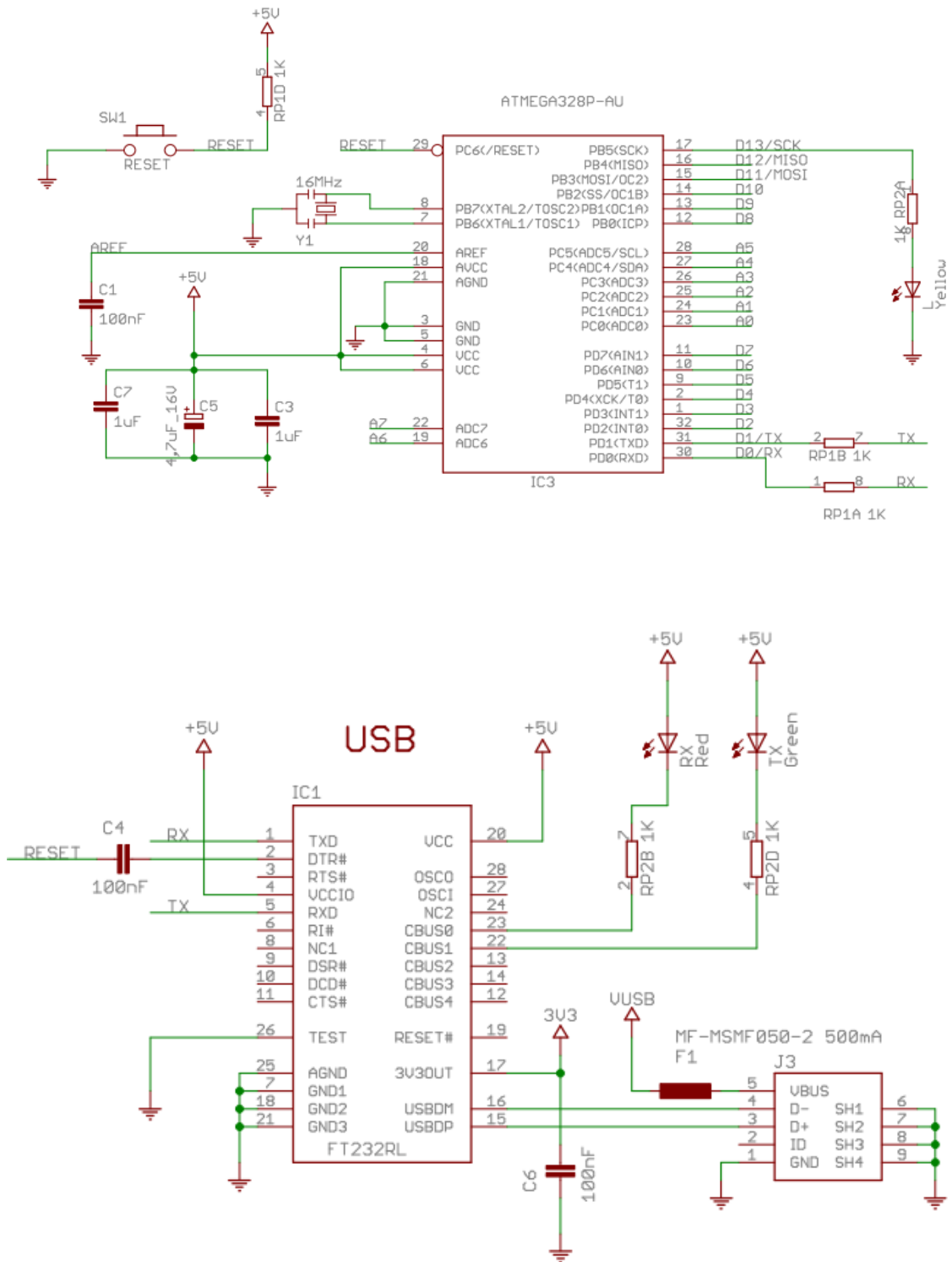
NO	PIN NAME	NO	PIN NAME
1	TX	20	USB D+
2	RX	21	GND
3	CTS	22	GND
4	RTS	23	P100
5	PCM CLK	24	P101
6	PCM OUT	25	P102
7	PCM IN	26	P103
8	PCM SYNC	27	P104
9	AIO0	28	P105
10	AIO1	29	P106
11	RESET	30	P107
12	3.3V	31	P108
13	GND	32	P109
14	NC	33	P1010
15	USB D-	34	P1011
16	CSB		
17	MOSI		
18	MISO		
19	CLK		

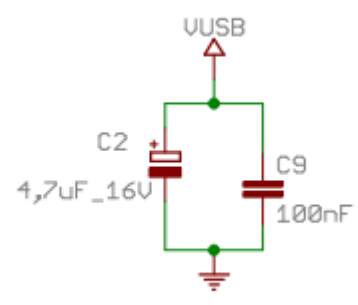
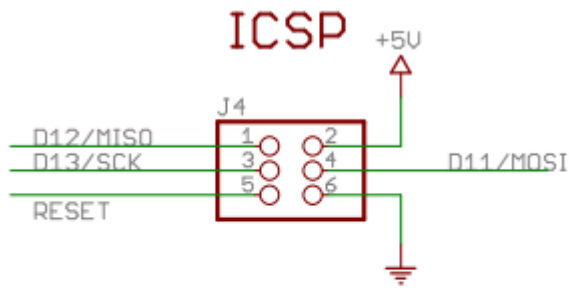
Esquema del módulo "MCU Card Easy Pic PRO v7"



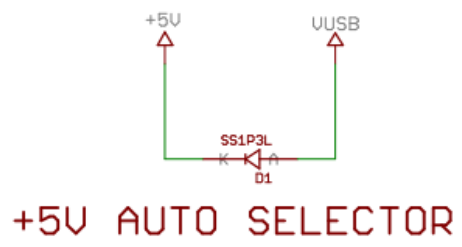
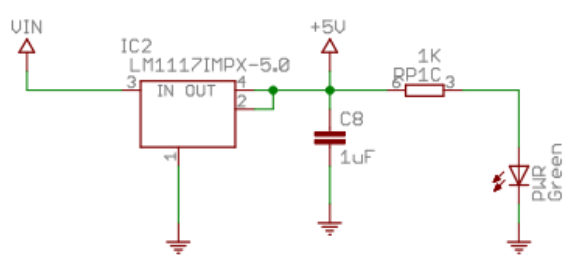
Datasheet del componente PIC18F87K22 disponible adjunto a este documento junto a la documentación de la placa "Standard 80-pin TQFP card with PIC18F87K22 MCU"

Esquemas de Arduino Nano

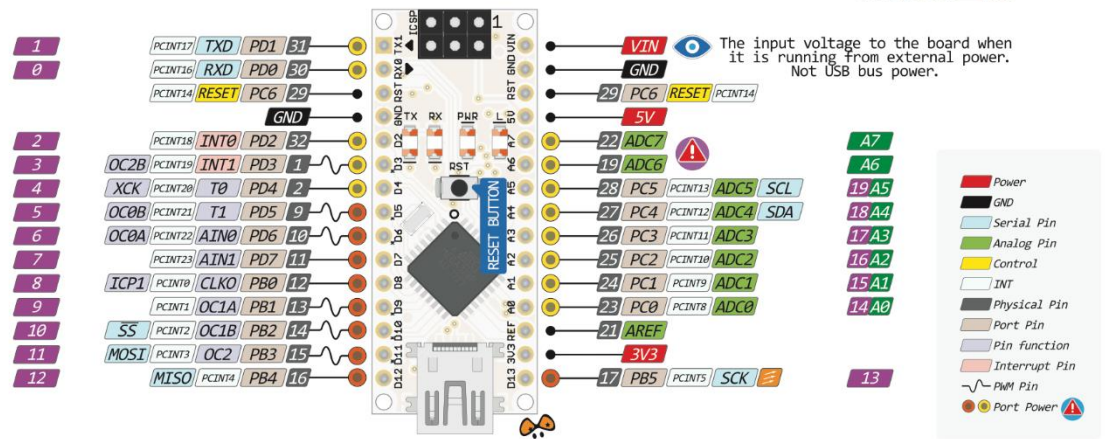
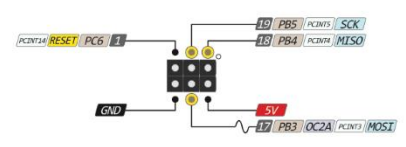




+5V REG



NANO PINOUT



⚠ Absolute MAX per pin 40mA recommended 20mA
 ⚠ Absolute MAX 200mA for entire package

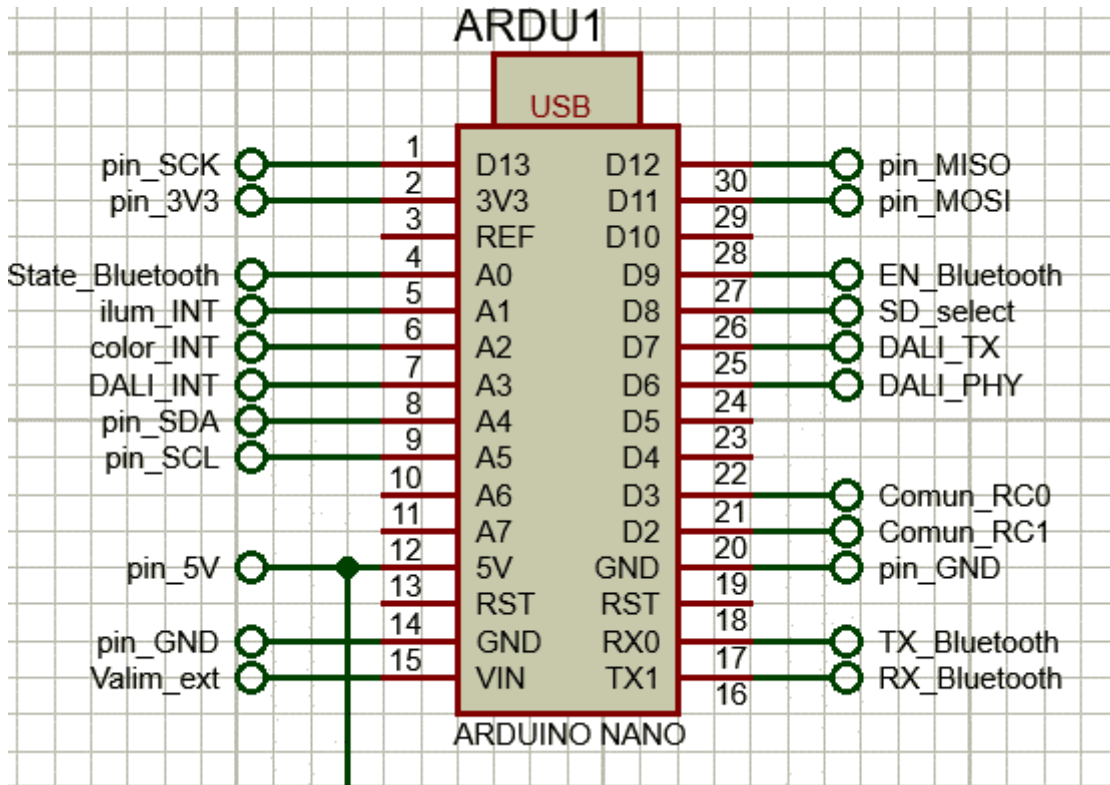


⚠ Analog exclusively Pins

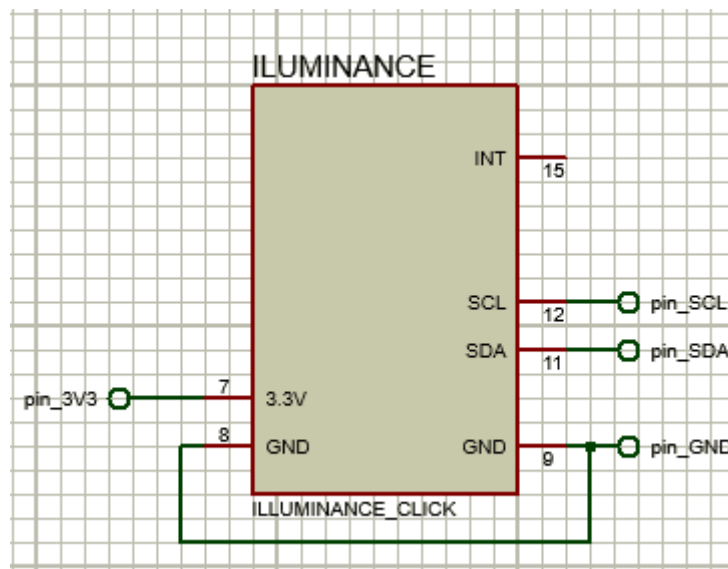
⚠ The power sum for each pin's group should not exceed 100mA



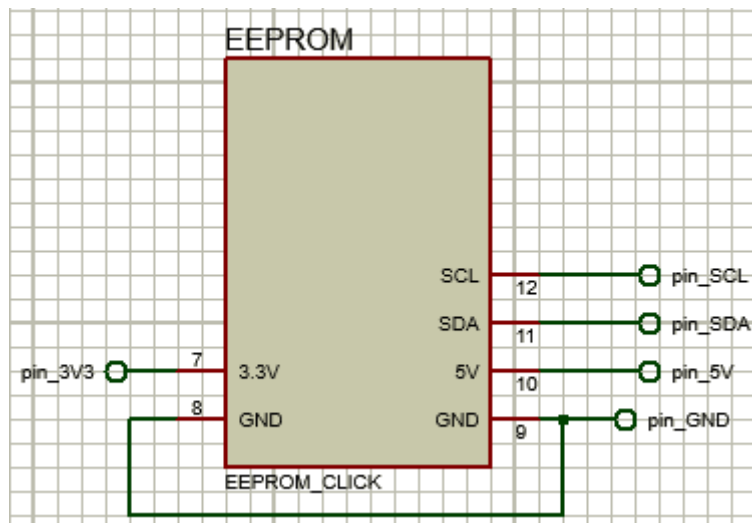
Esquema de conexión del microcontrolador Arduino Nano



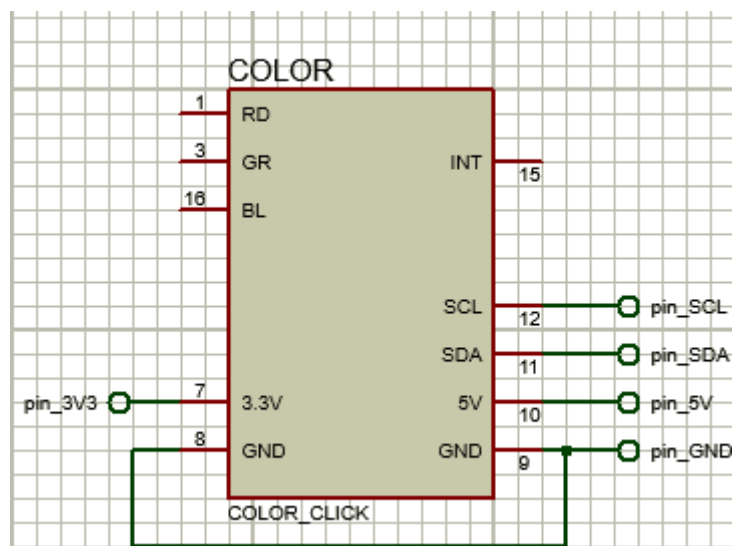
Esquema de conexión del módulo “illuminance click”



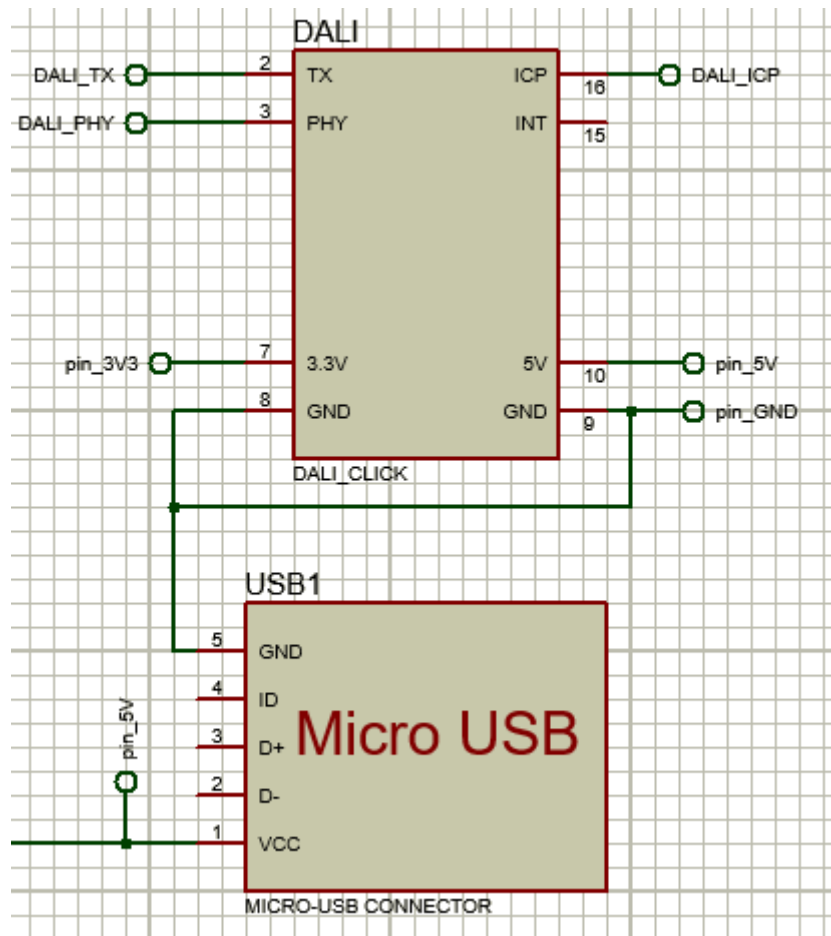
Esquema de conexión del módulo "EEPROM click"



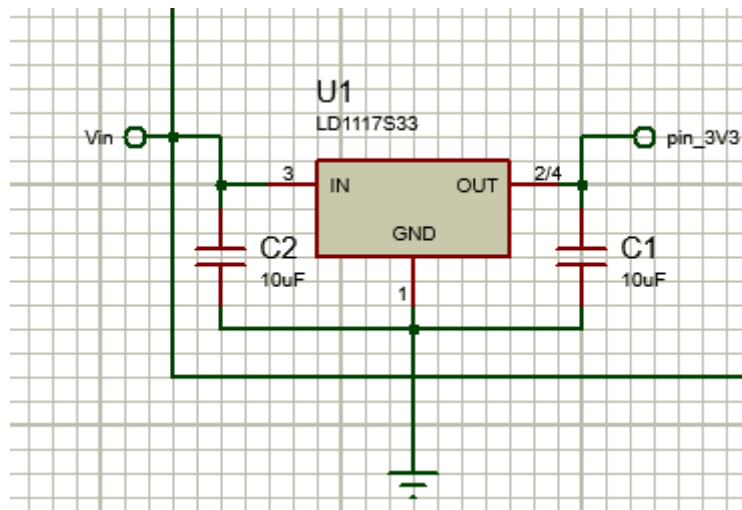
Esquema de conexión del módulo "color click"



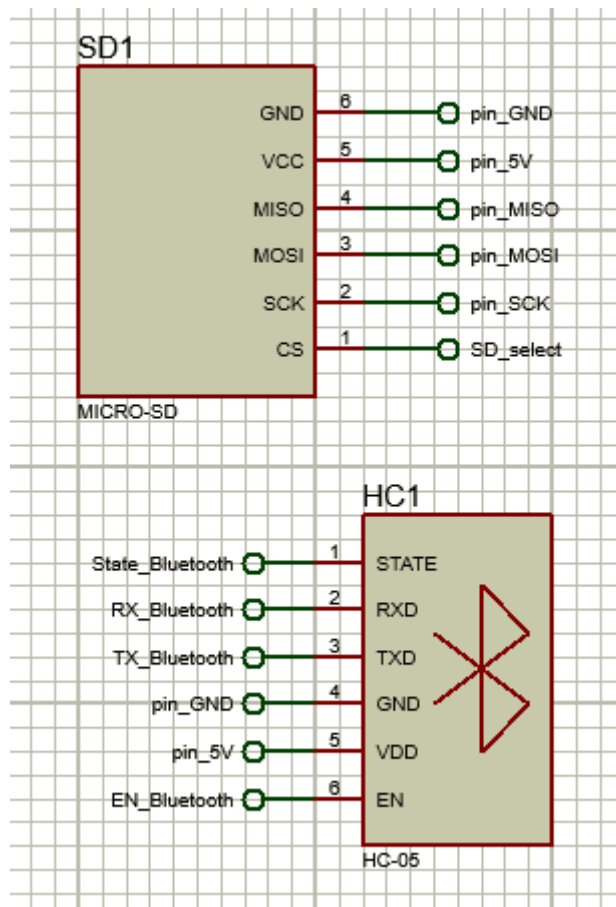
Esquema de conexión del módulo "DALI click" y el conector micro USB



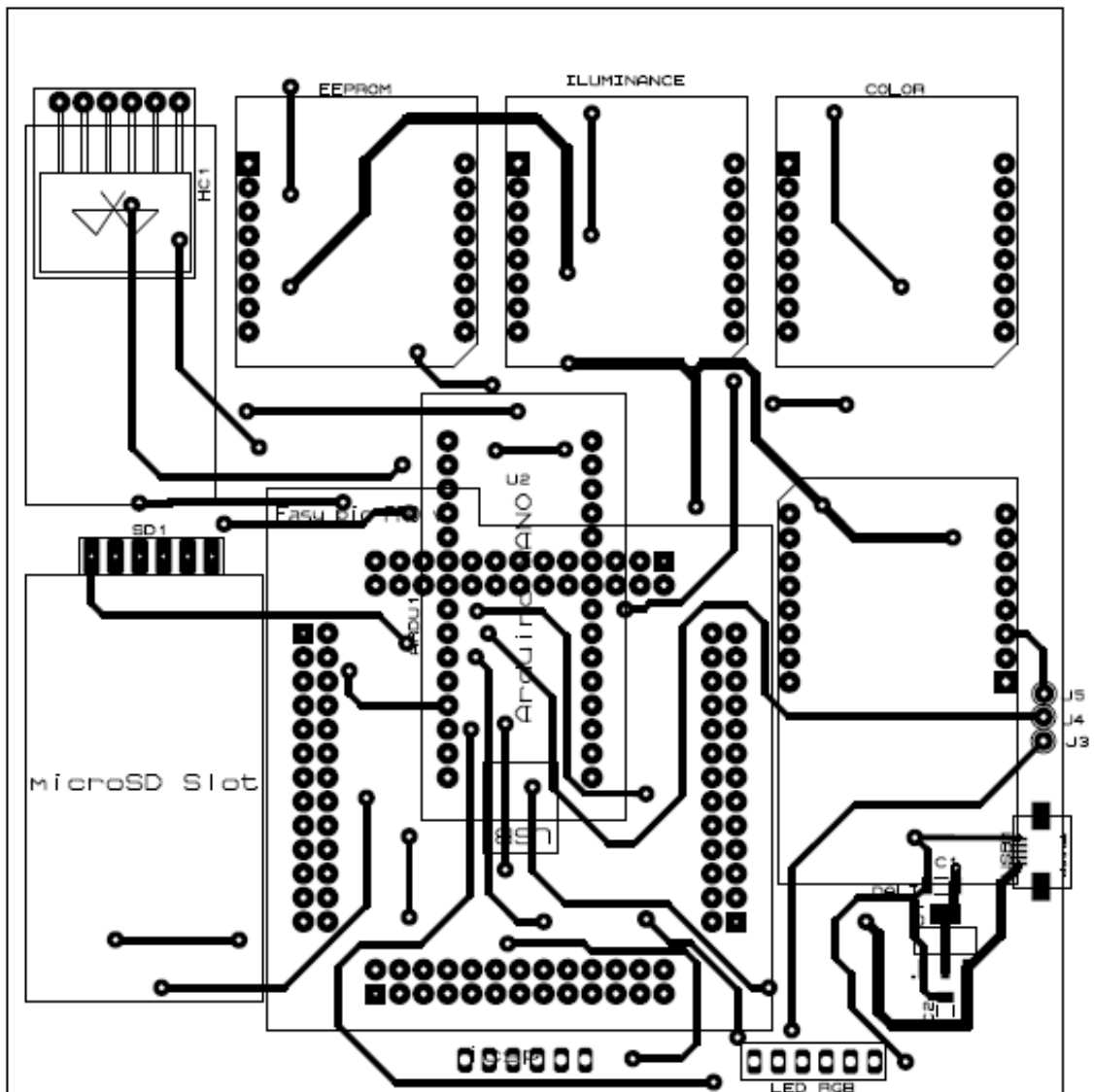
Esquema de conexión del regulador LD1117S33



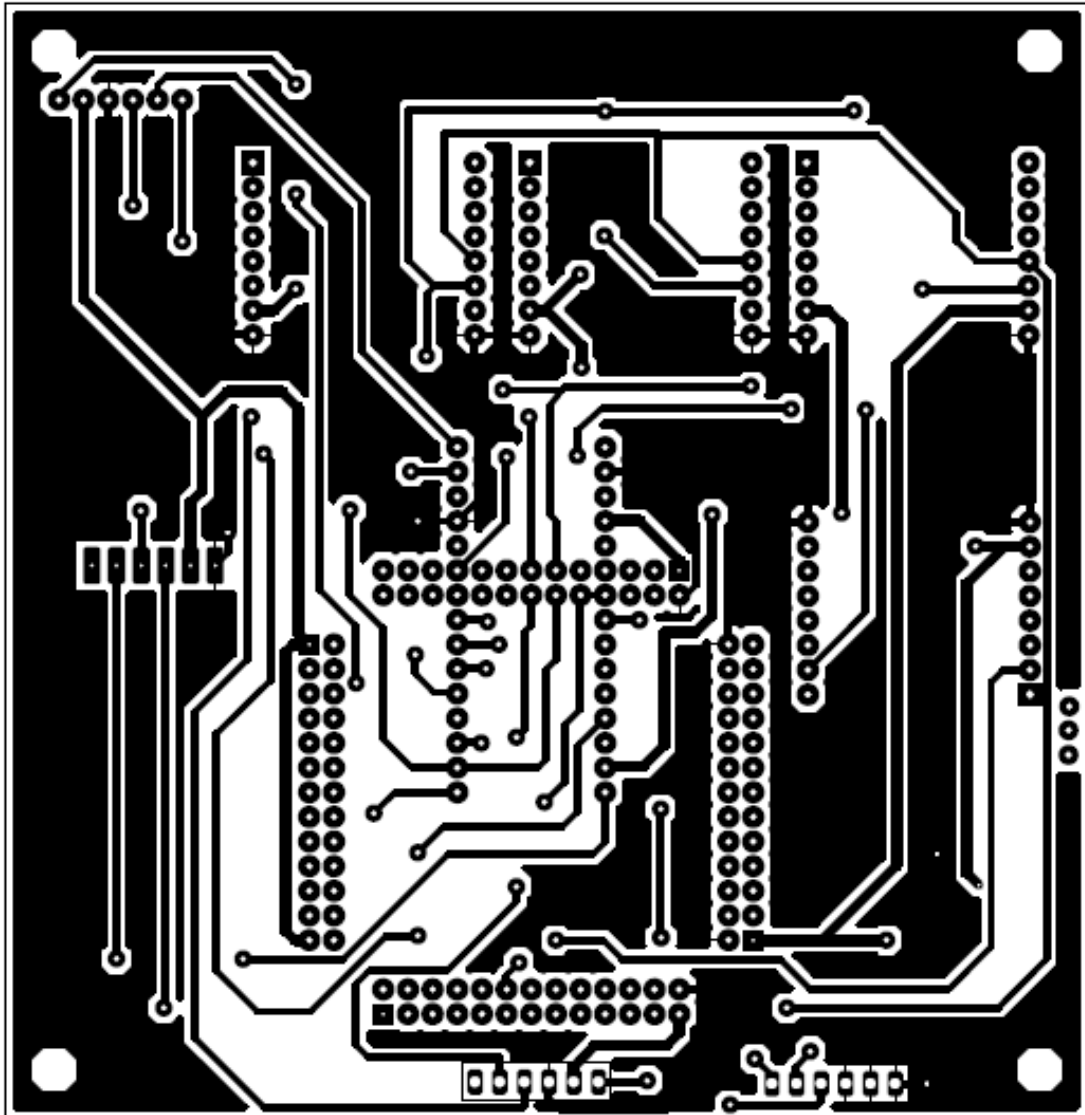
Esquema de conexión del módulo microSD y HC-05 (Bluetooth)



Layout de la placa de circuito impreso (Top)



Layout de la placa de circuito impreso (Bottom)



ANEXO IV. Componentes DALI externos

Balastro DALI

DIN-413-6A

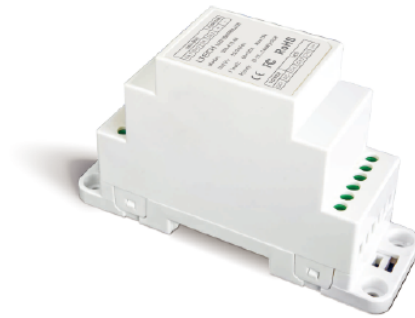
LTECH

DALI

DIN-Rail LED Dimming Driver

0-216...432W 6A x 3CH 12-24V

- Dimming interface: DALI, Push Dim.
- PWM digital dimming, standard DALI logarithmic dimming curve.
- Dimming range: 0-100%, LED start at 0.1% possible.
- Max. current output: 18A.
- Full protective plastic housing.
- DALI bus standard: IEC62386.
- Suitable for indoor environments.

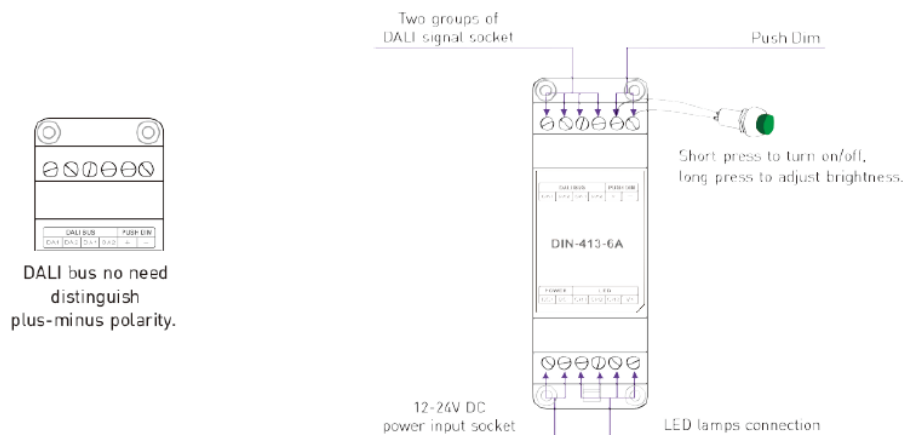


Main Characteristics

Dimming Interface:	DALI (IEC62386), Push Dim
Input Voltage:	12-24V DC
Output Voltage:	12-24V DC
Output Current	6A x 3CH Max. 18A
Max. Output Power:	0-216W...432W
Dimming Range:	0-100%, LED start at 0.1% possible

Mounting:	DIN-Rail or Screw
Working Temperature:	-30°C-55°C
Product Size:	L112xW35xH67(mm)
Package Size:	L114xW37xH70(mm)
Weight(G.W.):	135g

Conjunction Diagram



Bus DALI

DALI-PS-DIN

LTSYS

DALI
Bus Power

DALI Bus Power Supply (DIN Rail)

- Short circuit / Over-temperature protection.
- Natural air cooling.
- Full protective plastic housing.
- Class 2 power supply.
- Compliant with Safety Extra Low Voltage standard
- Suitable for indoor environments.



Main Characteristics

Input Voltage Range:	100-240Vac
Frequency:	50/60Hz
AC Current (typ.):	115Vac/≤0.09A, 230Vac/≤0.05A
Inrush Current (typ.):	Cold start 10A at 230Vac (twidth=75µs measured at 50% Ipeak)
Leakage Current:	<0.5mA/230Vac
Rated Current:	200mA
Operating Voltage:	15V DC
Max. Rated Power:	3W
Ripple & Noise (max.):	0.5Vp-p (working voltage at full load)
No Load Output Voltage (max.):	≤17V DC
Setup Time:	115Vac<2500ms, 230Vac<1000ms at full load
Working Temp.:	-30°C ~ 55°C
Working Humidity:	20 - 95%RH, non-condensing

Storage Temp., Humidity:	-40 ~ 80°C, 10-95%RH
Temp. Coefficient:	±0.03%/°C(0-50°C)
Vibration:	10-500Hz, 2G 12min./1cycle, period for 72min. each along X, Y, Z axes

Protection

Short Circuit and Over-temperature: Auto recovery

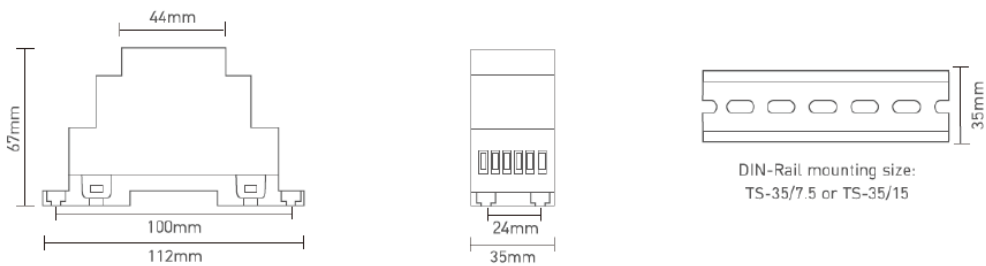
Safety & EMC

Isolation Resistance: I/P-O/P: 100MΩ/500VDC/25°C/70%RH

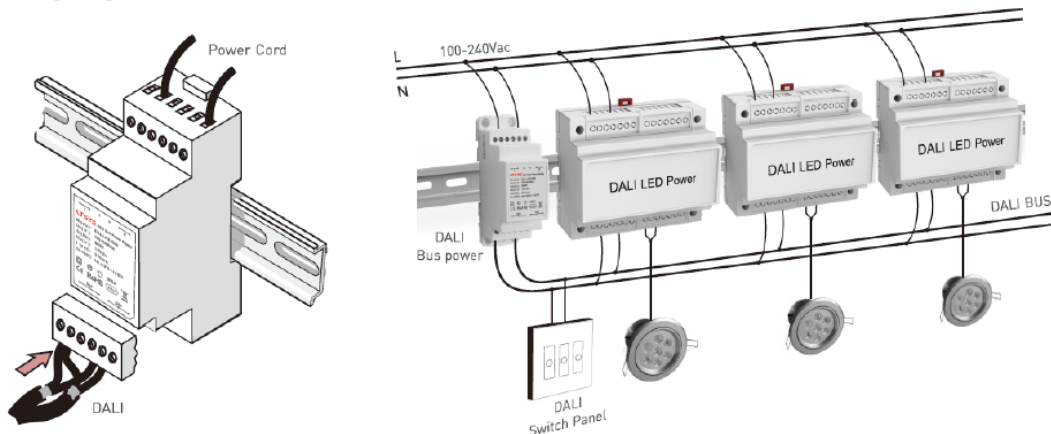
Packing Specification:

Dimension:	112×35×67mm(L×W×H)
Packing:	114×37×70mm(L×W×H)
Weight(G.W.):	115g±10g

Dimensions:



Wiring Diagram:



ANEXO V. Presupuesto

Presupuesto empleando PIC18F8520 como microcontrolador

Componente	Precio unitario	unidades	Precio total
Regulador 3,3V LD1117S33	0,41 €	1	0,41 €
Arduino nano	7,49 €	0	- €
Módulo Big Pic 5 (con PIC18F8520)	9,90 €	1	9,90 €
Módulo HC-05	7,20 €	1	7,20 €
Conector microUSB	0,47 €	1	0,47 €
Módulo DALI click	25,00 €	1	25,00 €
Módulo tarjeta microSD	16,00 €	1	16,00 €
Módulo illuminance click	16,00 €	1	16,00 €
Módulo EEPROM click	14,00 €	1	14,00 €
Módulo color click	15,00 €	1	15,00 €
Condensador 10uF smd 1206	0,12 €	2	0,24 €
Placa FR4 35um de cobre doble cara	5,03 €	1	5,03 €
Impresión fotolito	0,05 €	2	0,10 €
		TOTAL:	109,35 €

Presupuesto empleando Arduino Nano como microcontrolador

Componente	Precio unitario	unidades	Precio total
Regulador 3,3V LD1117S33	0,41 €	1	0,41 €
Arduino nano	7,49 €	1	7,49 €
Módulo Big Pic 5 (con PIC18F8520)	9,90 €	0	- €
Módulo HC-05	7,20 €	1	7,20 €
Conector microUSB	0,47 €	1	0,47 €
Módulo DALI click	25,00 €	1	25,00 €
Módulo tarjeta microSD	16,00 €	1	16,00 €
Módulo illuminance click	16,00 €	1	16,00 €
Módulo EEPROM click	14,00 €	1	14,00 €
Módulo color click	15,00 €	1	15,00 €
Condensador 10uF smd 1206	0,12 €	2	0,24 €
Placa FR4 35um de cobre doble cara	5,03 €	1	5,03 €
Impresión fotolito	0,05 €	2	0,10 €
TOTAL:			106,94 €

ANEXO VI. Código

Programa principal

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

#define SD_CS 8
#define TX 7
#define R 5
#define G 6
#define B 10

#define dir_verde 0x02
#define dir_rojo 0x04
#define dir_azul 0x06

#define pos_Escena 0
#define pos_Guardar 1
#define pos_Cchk 2
#define pos_Ri 3
#define pos_Gi 4
#define pos_Bi 5
#define pos_Dchk 6
#define pos_Rf 7
#define pos_Gf 8
#define pos_Bf 9
#define pos_T 10
#define pos_Fin 11
#define bytes_trama 12

File dataFile;

//Datos de ColorClick
unsigned char enable; //Respuesta del sensor de color
unsigned char atime; //Respuesta del sensor de color
unsigned char wtime; //Respuesta del sensor de color
unsigned char ailtl; //Respuesta del sensor de color
unsigned char ailth; //Respuesta del sensor de color
unsigned char aihtl; //Respuesta del sensor de color
unsigned char aihth; //Respuesta del sensor de color
unsigned char pers; //Respuesta del sensor de color
unsigned char configuracion; //Respuesta del sensor de color
unsigned char control;
unsigned char id; //Respuesta del sensor de color
unsigned char estado; //Respuesta del sensor de color
unsigned char Clear_LOW; //Respuesta del sensor de color
unsigned char Clear_HIGH; //Respuesta del sensor de color
unsigned char Red_LOW; //Respuesta del sensor de color
unsigned char Red_HIGH; //Respuesta del sensor de color
unsigned char Green_LOW; //Respuesta del sensor de color
unsigned char Green_HIGH; //Respuesta del sensor de color
unsigned char Blue_LOW; //Respuesta del sensor de color
unsigned char Blue_HIGH; //Respuesta del sensor de color
unsigned char dir, niv;

//Datos de illuminance
unsigned char illuminance[8];
unsigned char EEPROM_Data[bytes_trama];
bool logs_escritos = false;
unsigned char Rojo_actual, Verde_actual, Azul_actual, Rojo_final,
Verde_final, Azul_final, T_degradado;
```

```

struct EscenaDinamica {
    unsigned char Rojo_inicial;
    unsigned char Verde_inicial;
    unsigned char Azul_inicial;
    unsigned char Rojo_final;
    unsigned char Verde_final;
    unsigned char Azul_final;
    float Step_Rojo;
    float Step_Verde;
    float Step_Azul;
};
float Rojo_tmp, Verde_tmp, Azul_tmp;
//pin CS de la tarjeta microSD

EscenaDinamica EscenaActiva;
bool escena_dinamica = false;
bool genera_cabecera_ambiente = true;
bool genera_cabecera_funcionamiento = true;
unsigned long T_anterior = 0;
unsigned int step_actual, num_steps;
unsigned long ultimo_registro_ambiente = 0;
unsigned long T_registro_ambiente = 1000;
char registro[50];

void setup() {
    // put your setup code here, to run once:

    Wire.begin();
    Serial.begin(9600);
    //Inicialización del sensor de colores
    Wire.beginTransmission(0x29);
    Wire.write(0xA0); //Comando seleccionando ENABLE
    Wire.write(0x03); //Datos a escribir en el registro ENABLE
    Wire.endTransmission();
    delay(3); //Hay que esperar al menos 2.4ms hasta que el modulo se
inicialice
    //Establecer la ganancia
    Wire.beginTransmission(0x29);
    Wire.write(0xA1); //Comando seleccionando el registro RGBCTiming
    Wire.write(0xC0); //Escribe 154ms de tiempo de adq en el registro
RGBC
    Wire.endTransmission();
    //Sensor inicializado

    //Inicializa Illuminance Click
    Wire.beginTransmission(0x49);
    Wire.write(0x80); //Comando seleccionando ENABLE
    Wire.write(0x03); //Activa el sensor
    Wire.endTransmission();
    delay(3); //Hay que esperar al menos 2.4ms hasta que el modulo se
inicialice

```

```

//La EEPROM no hay que inicializarla

//Inicialización de la tarjeta microSD
////Serial.print(F("Iniciando tarjeta microSD..."));
pinMode(SD_CS, OUTPUT);
digitalWrite(SD_CS, HIGH);
if (!SD.begin(SD_CS)) {
    ////Serial.println(F("Fallo al iniciar la tarjeta micro SD"));
    return;
}
////Serial.println(F("Tarjeta micro SD inicializada"));

pinMode(TX, OUTPUT);
pinMode(R, OUTPUT);
pinMode(G, OUTPUT);
pinMode(B, OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    //Leer sensor de color
    int i = 0;
    //Trama de datos recibida por Bluetooth
    unsigned char Data[bytes_trama] = "";
    if (Serial.available()) {
        while (Data[pos_Fin] == 0) {
            if (Serial.available()) {
                Data[i] = Serial.read();
                i++;
                if (i > pos_Fin) {
                    i = 0;
                }
            }
        }
        for (int k = 0; k < bytes_trama; k++) {
            ////Serial.println(Data[k], DEC);
        }
    }
    //Serial.flush();
    //Serial.println("OK");
    /*
    for (int i = 0; i < 12; i++) {
        ////Serial.println(Data[i], DEC);
    }
    */
    //Interpreta la trama recibida
    /*
    Dato [0]
    P -> Aplicar escena personalizada
    1 -> Aplicar escena 1
    2 -> Aplicar escena 2
    3 -> Aplicar escena 3
    4 -> Aplicar escena 4
    A -> Almacenar escena 1
    B -> Almacenar escena 2
    C -> Almacenar escena 3
    D -> Almacenar escena 4
    */
    if (Data[0] == 'P') {
        //Aplicar escena personalizada
        escribe_logs_funcionamiento("Ap EscenaP");
    }
}

```

```

//escribe_logs_funcionamiento(Data);
if (Data[pos_Cchk] == 0) {
    Rojo_actual = Data[pos_Ri];
    Verde_actual = Data[pos_Gi];
    Azul_actual = Data[pos_Bi];
}
else{
    Rojo_actual = Rojo_tmp;
    Verde_actual = Verde_tmp;
    Azul_actual = Azul_tmp;
}
if (Data[pos_Dchk] == 1) {
    Rojo_final = Data[pos_Rf];
    Verde_final = Data[pos_Gf];
    Azul_final = Data[pos_Bf];
    T_degradado = Data[pos_T];
    EscenaActiva.Rojo_inicial = Rojo_actual;
    EscenaActiva.Verde_inicial = Verde_actual;
    EscenaActiva.Azul_inicial = Azul_actual;
    EscenaActiva.Rojo_final = Rojo_final;
    EscenaActiva.Verde_final = Verde_final;
    EscenaActiva.Azul_final = Azul_final;
    EscenaActiva.Step_Rojo = ((float)Rojo_final -
(float)Rojo_actual) / (60.0 * ((float)T_degradado - 48));
    EscenaActiva.Step_Verde = ((float)Verde_final -
(float)Verde_actual) / (60.0 * ((float)T_degradado - 48));
    EscenaActiva.Step_Azul = ((float)Azul_final -
(float)Azul_actual) / (60.0 * ((float)T_degradado - 48));
    num_steps = (T_degradado - 48) * 60;
    step_actual = 0;
    escena_dinamica = true;
}
else {
    escena_dinamica = false;
}
/*
float Rojo_tmp = (float)Rojo_actual;
if (Rojo_tmp > 255) {
    Rojo_tmp = 255;
}
float Verde_tmp = (float)Verde_actual;
if (Verde_tmp > 255) {
    Verde_tmp = 255;
}
float Azul_tmp = (float)Azul_actual;
if (Azul_tmp > 255) {
    Azul_tmp = 255;
}
*/
//Establece el color inicial
analogWrite(R, (unsigned char) Rojo_actual);
analogWrite(G, (unsigned char) Verde_actual);
analogWrite(B, (unsigned char) Azul_actual);
enviar_color((unsigned char) Rojo_actual, (unsigned char)
Verde_actual, (unsigned char) Azul_actual);

}
else {
    if (Data[0] == '1') {
        //Accion referente a escena 1
        if (Data[1] == 'X') {

```

```

//Se aplica la escena 1
escribe_logs_funcionamiento("Ap Escena1");
for (int i = 0; i < bytes_trama; i++) {
    leer_EEPROM(0x00, 0x00, bytes_trama);
    //Resultado en EEPROM_Data[];
}
//Carga el resultado en el vector Data
/////Serial.println(F("Cargamos el resultado de la
EEPROM:"));
for (int i = 0; i < bytes_trama; i++) {
    Data[i] = EEPROM_Data[i];
    ////Serial.println(Data[i]);
}
//escribe_logs_funcionamiento(Data);
if (Data[pos_Cchk] == 0) {
    Rojo_actual = Data[pos_Ri];
    Verde_actual = Data[pos_Gi];
    Azul_actual = Data[pos_Bi];
}
if (Data[pos_Dchk] == 1) {
    Rojo_final = Data[pos_Rf];
    Verde_final = Data[pos_Gf];
    Azul_final = Data[pos_Bf];
    T_degradado = Data[pos_T];
    EscenaActiva.Rojo_inicial = Rojo_actual;
    EscenaActiva.Verde_inicial = Verde_actual;
    EscenaActiva.Azul_inicial = Azul_actual;
    EscenaActiva.Rojo_final = Rojo_final;
    EscenaActiva.Verde_final = Verde_final;
    EscenaActiva.Azul_final = Azul_final;
    EscenaActiva.Step_Rojo = ((float)Rojo_final -
(float)Rojo_actual) / (60.0 * ((float)T_degradado - 48));
    EscenaActiva.Step_Verde = ((float)Verde_final -
(float)Verde_actual) / (60.0 * ((float)T_degradado - 48));
    EscenaActiva.Step_Azul = ((float)Azul_final -
(float)Azul_actual) / (60.0 * ((float)T_degradado - 48));
    num_steps = (T_degradado - 48) * 60;
    step_actual = 0;
    escena_dinamica = true;
}
else {
    escena_dinamica = false;
}
//Establece el color inicial
analogWrite(R, (unsigned char) Rojo_actual);
analogWrite(G, (unsigned char) Verde_actual);
analogWrite(B, (unsigned char) Azul_actual);
enviar_color((unsigned char) Rojo_actual, (unsigned char)
Verde_actual, (unsigned char) Azul_actual);
}
else {
    //Se guarda la escena en la EEPROM
    escribe_logs_funcionamiento("G. Escena1");
    //escribe_logs_funcionamiento(Data);
    for (unsigned char i = 0; i < bytes_trama; i++) {
        escribir_EEPROM(0x00, i, Data[i]);
    }
    ////Serial.println(F("Comprobacion de la memoria EEPROM"));
    leer_EEPROM(0x00, 0x00, bytes_trama);
    for (int i = 0; i < bytes_trama; i++) {
        ////Serial.println(EEPROM_Data[i]);
    }
}

```

```

    }
  }
}
else {
  if (Data[0] == '2') {

    //Accion referente a escena 2
    if (Data[1] == 'X') {
      //Se aplica la escena 2
      escribe_logs_funcionamiento("Ap Escena2");
      for (int i = 0; i < bytes_trama; i++) {
        leer_EEPROM(0x01, 0x00, bytes_trama);
        //Resultado en EEPROM_Data[];
      }
      //Carga el resultado en el vector Data
      ////Serial.println(F("Cargamos el resultado de la
EEPROM:"));
      for (int i = 0; i < bytes_trama; i++) {
        Data[i] = EEPROM_Data[i];
        //Serial.println(Data[i]);
      }
      //escribe_logs_funcionamiento(Data);

      if (Data[pos_Cchk] == 0) {
        Rojo_actual = Data[pos_Ri];
        Verde_actual = Data[pos_Gi];
        Azul_actual = Data[pos_Bi];
      }
      if (Data[pos_Dchk] == 1) {
        Rojo_final = Data[pos_Rf];
        Verde_final = Data[pos_Gf];
        Azul_final = Data[pos_Bf];
        T_degradado = Data[pos_T];
        EscenaActiva.Rojo_inicial = Rojo_actual;
        EscenaActiva.Verde_inicial = Verde_actual;
        EscenaActiva.Azul_inicial = Azul_actual;
        EscenaActiva.Rojo_final = Rojo_final;
        EscenaActiva.Verde_final = Verde_final;
        EscenaActiva.Azul_final = Azul_final;
        EscenaActiva.Step_Rojo = ((float)Rojo_final -
(float)Rojo_actual) / (60.0 * ((float)T_degradado - 48));
        EscenaActiva.Step_Verde = ((float)Verde_final -
(float)Verde_actual) / (60.0 * ((float)T_degradado - 48));
        EscenaActiva.Step_Azul = ((float)Azul_final -
(float)Azul_actual) / (60.0 * ((float)T_degradado - 48));
        num_steps = (T_degradado - 48) * 60;
        step_actual = 0;
        escena_dinamica = true;
      }
      else {
        escena_dinamica = false;
      }
      //Establece el color inicial
      analogWrite(R, (unsigned char) Rojo_actual);
      analogWrite(G, (unsigned char) Verde_actual);
      analogWrite(B, (unsigned char) Azul_actual);
      enviar_color((unsigned char) Rojo_actual, (unsigned char)
Verde_actual, (unsigned char) Azul_actual);
    }
    else {
      //Se guarda la escena en la EEPROM

```

```

    escribe_logs_funcionamiento("G. Escena2");
    //escribe_logs_funcionamiento(Data);
    for (unsigned char i = 0; i < bytes_trama; i++) {
        escribir_EEPROM(0x01, i, Data[i]);
    }
    //Serial.println(F("Comprobacion de la memoria EEPROM"));
    leer_EEPROM(0x01, 0x00, bytes_trama);
    for (int i = 0; i < bytes_trama; i++) {
        //Serial.println(EEPROM_Data[i]);
    }
}
}
else {
    if (Data[0] == '3') {

        //Accion referente a escena 3
        if (Data[1] == 'X') {
            //Se aplica la escena 3
            escribe_logs_funcionamiento("Ap. Escena3");
            for (int i = 0; i < bytes_trama; i++) {
                leer_EEPROM(0x02, 0x00, bytes_trama);
                //Resultado en EEPROM_Data[];
            }
            //Carga el resultado en el vector Data
            //Serial.println(F("Cargamos el resultado de la
EEPROM:"));
            for (int i = 0; i < bytes_trama; i++) {
                Data[i] = EEPROM_Data[i];
                //Serial.println(Data[i]);
            }
            //escribe_logs_funcionamiento(Data);

            if (Data[pos_Cchk] == 0) {
                Rojo_actual = Data[pos_Ri];
                Verde_actual = Data[pos_Gi];
                Azul_actual = Data[pos_Bi];
            }
            if (Data[pos_Dchk] == 1) {
                Rojo_final = Data[pos_Rf];
                Verde_final = Data[pos_Gf];
                Azul_final = Data[pos_Bf];
                T_degradado = Data[pos_T];
                EscenaActiva.Rojo_inicial = Rojo_actual;
                EscenaActiva.Verde_inicial = Verde_actual;
                EscenaActiva.Azul_inicial = Azul_actual;
                EscenaActiva.Rojo_final = Rojo_final;
                EscenaActiva.Verde_final = Verde_final;
                EscenaActiva.Azul_final = Azul_final;
                EscenaActiva.Step_Rojo = ((float)Rojo_final -
(float)Rojo_actual) / (60.0 * ((float)T_degradado - 48));
                EscenaActiva.Step_Verde = ((float)Verde_final -
(float)Verde_actual) / (60.0 * ((float)T_degradado - 48));
                EscenaActiva.Step_Azul = ((float)Azul_final -
(float)Azul_actual) / (60.0 * ((float)T_degradado - 48));
                num_steps = (T_degradado - 48) * 60;
                step_actual = 0;
                escena_dinamica = true;
            }
            else {
                escena_dinamica = false;
            }
        }
    }
}
}

```



```

//Establece el color inicial
analogWrite(R, (unsigned char) Rojo_actual);
analogWrite(G, (unsigned char) Verde_actual);
analogWrite(B, (unsigned char) Azul_actual);
enviar_color((unsigned char) Rojo_actual, (unsigned char)
Verde_actual, (unsigned char) Azul_actual);
}
else {
//Se guarda la escena en la EEPROM
escribe_logs_funcionamiento("G. Escena3");
//escribe_logs_funcionamiento(Data);
for (unsigned char i = 0; i < bytes_trama; i++) {
    escribir_EEPROM(0x02, i, Data[i]);
}
//Serial.println(F("Comprobacion de la memoria EEPROM"));
leer_EEPROM(0x02, 0x00, bytes_trama);
for (int i = 0; i < bytes_trama; i++) {
    //Serial.println(EEPROM_Data[i]);
}
}
}
else {
if (Data[0] == '4') {

//Accion referente a escena 4
if (Data[1] == 'X') {
//Se aplica la escena 4
escribe_logs_funcionamiento("Ap Escena4");
for (int i = 0; i < bytes_trama; i++) {
    leer_EEPROM(0x03, 0x00, bytes_trama);
//Resultado en EEPROM_Data[];
}
//Carga el resultado en el vector Data
////Serial.println(F("Cargamos el resultado de la
EEPROM:"));
for (int i = 0; i < bytes_trama; i++) {
    Data[i] = EEPROM_Data[i];
//Serial.println(Data[i]);
}
//escribe_logs_funcionamiento(Data);

if (Data[pos_Cchk] == 0) {
    Rojo_actual = Data[pos_Ri];
    Verde_actual = Data[pos_Gi];
    Azul_actual = Data[pos_Bi];
}
if (Data[pos_Dchk] == 1) {
    Rojo_final = Data[pos_Rf];
    Verde_final = Data[pos_Gf];
    Azul_final = Data[pos_Bf];
    T_degradado = Data[pos_T];
    EscenaActiva.Rojo_inicial = Rojo_actual;
    EscenaActiva.Verde_inicial = Verde_actual;
    EscenaActiva.Azul_inicial = Azul_actual;
    EscenaActiva.Rojo_final = Rojo_final;
    EscenaActiva.Verde_final = Verde_final;
    EscenaActiva.Azul_final = Azul_final;
    EscenaActiva.Step_Rojo = ((float)Rojo_final -
(float)Rojo_actual) / (60.0 * ((float)T_degradado - 48));
    EscenaActiva.Step_Verde = ((float)Verde_final -
(float)Verde_actual) / (60.0 * ((float)T_degradado - 48));

```

```

        EscenaActiva.Step_Azul = ((float)Azul_final -
(float)Azul_actual) / (60.0 * ((float)T_degradado - 48));
        num_steps = (T_degradado - 48) * 60;
        step_actual = 0;
        escena_dinamica = true;
    }
    else {
        escena_dinamica = false;
    }
    //Establece el color inicial
    analogWrite(R, (unsigned char) Rojo_actual);
    analogWrite(G, (unsigned char) Verde_actual);
    analogWrite(B, (unsigned char) Azul_actual);
    enviar_color((unsigned char) Rojo_actual, (unsigned char)
Verde_actual, (unsigned char) Azul_actual);
}
else {
    //Se guarda la escena en la EEPROM
    escribe_logs_funcionamiento("G. Escena4");
    //escribe_logs_funcionamiento(Data);
    for (unsigned char i = 0; i < bytes_trama; i++) {
        escribir_EEPROM(0x03, i, Data[i]);
    }
    //Serial.println("Comprobacion de la memoria EEPROM");
    leer_EEPROM(0x03, 0x00, bytes_trama);
    for (int i = 0; i < bytes_trama; i++) {
        //Serial.println(EEPROM_Data[i]);
    }
}
}
}
}
}
//Lee los valores de los sensores
leer_colorClick();
leer_illuminanceClick();

//Actualiza el registro de ambiente.log
if (millis()-ultimo_registro_ambiente > T_registro_ambiente) {
    escribe_logs_ambiente();
    ultimo_registro_ambiente = millis();
}
if (escena_dinamica) {
    //escribe_logs_funcionamiento("Escena din.");
    if (millis() - T_anterior > 1000) {
        T_anterior = millis();
        if (step_actual <= num_steps) {

            Rojo_tmp = (float)Rojo_actual + step_actual *
EscenaActiva.Step_Rojo;
            if (Rojo_tmp > 254) {
                Rojo_tmp = 254;
            }
            Verde_tmp = (float)Verde_actual + step_actual *
EscenaActiva.Step_Verde;
            if (Verde_tmp > 254) {
                Verde_tmp = 254;
            }
            Azul_tmp = (float)Azul_actual + step_actual *
EscenaActiva.Step_Azul;

```

```

        if (Azul_tmp > 254) {
            Azul_tmp = 254;
        }
        //Establece el color inicial
        //Serial.println((unsigned char)Rojo_tmp);
        //Serial.println((unsigned char)Verde_tmp);
        //Serial.println((unsigned char)Azul_tmp);
        analogWrite(R, (unsigned char) Rojo_tmp);
        analogWrite(G, (unsigned char) Verde_tmp);
        analogWrite(B, (unsigned char) Azul_tmp);
        enviar_color((unsigned char) Rojo_tmp, (unsigned char)
Verde_tmp, (unsigned char) Azul_tmp);
        step_actual++;
    }
    //Retorno al color inicial
    else {
        if (step_actual <= 2 * num_steps) {
            Rojo_tmp -= EscenaActiva.Step_Rojo;
            if (Rojo_tmp > 254) {
                Rojo_tmp = 254;
            }
            if (Rojo_tmp < 0) {
                Rojo_tmp = 0;
            }
            Verde_tmp -= EscenaActiva.Step_Verde;
            if (Verde_tmp > 254) {
                Verde_tmp = 254;
            }
            if (Verde_tmp < 0) {
                Verde_tmp = 0;
            }
            Azul_tmp -= EscenaActiva.Step_Azul;
            if (Azul_tmp > 254) {
                Azul_tmp = 254;
            }
            if (Azul_tmp < 0) {
                Azul_tmp = 0;
            }
            //Establece el color inicial
            //Serial.println((unsigned char)Rojo_tmp);
            //Serial.println((unsigned char)Verde_tmp);
            //Serial.println((unsigned char)Azul_tmp);
            analogWrite(R, (unsigned char) Rojo_tmp);
            analogWrite(G, (unsigned char) Verde_tmp);
            analogWrite(B, (unsigned char) Azul_tmp);
            enviar_color((unsigned char) Rojo_tmp, (unsigned char)
Verde_tmp, (unsigned char) Azul_tmp);
            step_actual++;
        }
        else {
            step_actual = 0;
        }
    }
}
}
}
}

```

Funciones

```
void leer_colorClick() {
    Wire.beginTransmission(0x29); //Selecciona el dispositivo
    Wire.write(0xA0); //Comando de selección del bit primero de
CLEAR (B4)
    Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
    Wire.requestFrom(0x29, 1);
    while (Wire.available()) {
        enable = Wire.read();
    }

    Wire.beginTransmission(0x29); //Selecciona el dispositivo
    Wire.write(0xA1); //Comando de selección del bit primero de
CLEAR (B4)
    Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
    Wire.requestFrom(0x29, 1);
    while (Wire.available()) {
        atime = Wire.read();
    }

    Wire.beginTransmission(0x29); //Selecciona el dispositivo
    Wire.write(0xA3); //Comando de selección del bit primero de
CLEAR (B4)
    Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
    Wire.requestFrom(0x29, 1);
    while (Wire.available()) {
        wtime = Wire.read();
    }

    Wire.beginTransmission(0x29); //Selecciona el dispositivo
    Wire.write(0xA4); //Comando de selección del bit primero de
CLEAR (B4)
    Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
    Wire.requestFrom(0x29, 1);
    while (Wire.available()) {
        ailtl = Wire.read();
    }

    Wire.beginTransmission(0x29); //Selecciona el dispositivo
    Wire.write(0xA5); //Comando de selección del bit primero de
CLEAR (B4)
    Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
    Wire.requestFrom(0x29, 1);
    while (Wire.available()) {
        ailth = Wire.read();
    }

    Wire.beginTransmission(0x29); //Selecciona el dispositivo
    Wire.write(0xA6); //Comando de selección del bit primero de
CLEAR (B4)
    Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
    Wire.requestFrom(0x29, 1);
    while (Wire.available()) {
        aihtl = Wire.read();
    }

    Wire.beginTransmission(0x29); //Selecciona el dispositivo
    Wire.write(0xA7); //Comando de selección del bit primero de
CLEAR (B4)
    Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
}
```

```

Wire.requestFrom(0x29, 1);
while (Wire.available()) {
  aihth = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0xAC); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
  pers = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0xAD); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
  configuracion = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0xAF); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
  control = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0xB2); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
  id = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0xB3); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
  estado = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0xB4); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
  Clear_LOW = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo

```

```

Wire.write(0xB5); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
    Clear_HIGH = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0xB6); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
    Red_LOW = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0xB7); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
    Red_HIGH = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0xB8); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
    Green_LOW = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0xB9); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
    Green_HIGH = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0x9A); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
    Blue_LOW = Wire.read();
}

Wire.beginTransmission(0x29); //Selecciona el dispositivo
Wire.write(0xBB); //Comando de selección del bit primero de
CLEAR (B4)
Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
Wire.requestFrom(0x29, 1);
while (Wire.available()) {
    Blue_HIGH = Wire.read();
}

```

```

/*
  Serial.println(F("*****"));
  Serial.println(F("*****  DATOS DEL SENSOR  *****"));
  Serial.println(F("*****"));
  Serial.print(F("  ENABLE --> ")); Serial.println(enable);
  Serial.print(F("  ATIME --> ")); Serial.println(atime);
  Serial.print(F("  WTIME --> ")); Serial.println(wtime);
  Serial.print(F("  AILT  --> ")); Serial.println(ailt);
  Serial.print(F("  AILTH --> ")); Serial.println(ailth);
  Serial.print(F("  AIHTL --> ")); Serial.println(aihtl);
  Serial.print(F("  AIHTH --> ")); Serial.println(aihth);
  Serial.print(F("  PERS  --> ")); Serial.println(pers);
  Serial.print(F("  CONFIG --> ")); Serial.println(configuracion);
  Serial.print(F("  CONTROL--> ")); Serial.println(control);
  Serial.print(F("  ID    --> ")); Serial.println(id);
  Serial.print(F("  STATUS --> ")); Serial.println(estado);
  Serial.print(F("  CDATE --> ")); Serial.println(Clear_LOW);
  Serial.print(F("  CDATAH --> ")); Serial.println(Clear_HIGH);
  Serial.print(F("  RDATA  --> ")); Serial.println(Red_LOW);
  Serial.print(F("  RDATAH --> ")); Serial.println(Red_HIGH);
  Serial.print(F("  GDATA  --> ")); Serial.println(Green_LOW);
  Serial.print(F("  GDATAH --> ")); Serial.println(Green_HIGH);
  Serial.print(F("  BDATA  --> ")); Serial.println(Blue_LOW);
  Serial.print(F("  BDATAH --> ")); Serial.println(Blue_HIGH);
  Serial.println(F("*****"));
  Serial.println(F("*****"));
  Serial.print(F("  BLANCO --> ")); Serial.println(Clear_HIGH * 256
+ Clear_LOW);
  Serial.print(F("  ROJO   --> ")); Serial.println(Red_HIGH * 256 +
Red_LOW);
  Serial.print(F("  VERDE  --> ")); Serial.println(Green_HIGH * 256
+ Green_LOW);
  Serial.print(F("  AZUL   --> ")); Serial.println(Blue_HIGH * 256
+ Blue_LOW);
  Serial.println(F("*****"));
  Serial.println(F("*****"));
  delay(3500);
*/
}

void leer_illuminanceClick() {
  int i = 0;
  /*
  Serial.println(F("*****illuminance*****"));
  Serial.println(F("*****"));
  */
  Wire.beginTransmission(0x49); //Selecciona el dispositivo
  Wire.write(0x8C); //Selecciona el registro Visible+IR Low
  Wire.endTransmission(); //Se leen los dos datos y se libera el bus.
  Wire.requestFrom(0x49, 4);
  while (Wire.available()) {
    illuminance[i] = Wire.read();
    //Serial.print(illuminance[i]);
    //Serial.print(F("-"));
    i++;
  }
  /*
  Serial.println(F("*****"));
  Serial.println(F("*****"));
  delay(3500);
  */
}

```

```

}

void leer_EEPROM(unsigned char addressLOW, unsigned char addressHIGH,
unsigned char n_Bytes) {
  //Selecciona la dirección de lectura/escritura
  Wire.beginTransaction(0x54);
  Wire.write(addressLOW); //Comando con la parte alta de la dirección
  Wire.write(addressHIGH); //parte baja de la dirección a escribir
  Wire.endTransmission();

  //Solicita N bytes a partir de la dirección seleccionada
  Wire.requestFrom(0x54, n_Bytes);
  int i = 0;
  while (Wire.available()) {
    EEPROM_Data[i] = Wire.read();
    //Serial.println(EEPROM_Data[i]);
    i++;
  }
  /*
  delay(5000);
  Serial.println("*****");
  */
}

void escribe_logs_funcionamiento(char registro[12]) {
  dataFile = SD.open("Func.log", FILE_WRITE);
  if (dataFile) {
    Serial.println(F("Logs de funcionamiento correctos"));
    if (genera_cabecera_funcionamiento) {
      dataFile.println("Registro de funcionamiento");
      genera_cabecera_funcionamiento = false;
    }
    dataFile.print(millis() / 1000);
    dataFile.print(" : ");
    dataFile.println(registro);
    dataFile.close();
  }
  else {
    Serial.println(F("Error al abrir el log de funcionamiento"));
  }
}

void escribe_logs_ambiente() {
  //Abre el archivo en cuestión
  dataFile = SD.open("Ambiente.log", FILE_WRITE);
  if (dataFile) {
    Serial.println(F("Fichero de logs generado correctamente"));
    if (genera_cabecera_ambiente) {
      dataFile.println(F("Registro de los datos de sensores"));
      dataFile.println(F("Tiempo ; Visible + IR ; IR ; R ; G ; B"));
      genera_cabecera_ambiente = false;
    }
    dataFile.print(millis() / 1000);
    dataFile.print(";");
    dataFile.print(illuminance[0] + illuminance[1] * 256);
    dataFile.print(";");
    dataFile.print(illuminance[2] + illuminance[3] * 256);
    dataFile.print(";");
    dataFile.print(Red_HIGH * 256 + Red_LOW);
    dataFile.print(";");
    dataFile.print(Green_HIGH * 256 + Green_LOW);
    dataFile.print(";");
    dataFile.print(Blue_HIGH * 256 + Blue_LOW);
  }
}

```



```

        dataFile.println("");
        dataFile.close();
    }
    else {
        Serial.println("Error al abrir el los ambiente");
    }
}
/*
void blink_led(unsigned int veces) {
Bluetooth.println("OK");
for (int i = 0; i < veces; i++) {
    digitalWrite(led_pin, HIGH);
    delay(250);
    digitalWrite(led_pin, LOW);
    delay(250);
}
}
*/

void escribir_EEPROM(unsigned char Adress_HIGH, unsigned char
Adress_LOW, unsigned char dato) {
    Wire.beginTransaction(0x54); //Comando para escribir en una
direccion oxoo...
    Wire.write(Adress_HIGH); //Comando con la parte alta de la direccion
    Wire.write(Adress_LOW); //parte baja de la direccion a escribir
    Wire.write(dato); //Datos a escribir en la dirección seleccionada
    Wire.endTransmission();
    delay(5);
}

// ----- FUNCIONES DALI ----- //

void send_one() {
    digitalWrite(TX, LOW);
    delayMicroseconds(416);
    digitalWrite(TX, HIGH);
    delayMicroseconds(416);
}

void send_zero() {
    digitalWrite(TX, HIGH);
    delayMicroseconds(416);
    digitalWrite(TX, LOW);
    delayMicroseconds(416);
}

void send_data(unsigned char value) {
    unsigned char s = 0;
    for (s = 0; s < 8; s++) {
        if ((value & 0x80) != 0) {
            send_one();
        }
        else {
            send_zero();
        }
        value <<= 1;
    }
}

void send_stop() {
    digitalWrite(TX, HIGH);
}

```

```

    delay(15);
}

void blanco_6500K() {
    send_one();
    send_data(dir_rojo);
    send_data(228);
    send_stop();
    send_one();
    send_data(dir_verde);
    send_data(240);
    send_stop();
    send_one();
    send_data(dir_azul);
    send_data(254);
    send_stop();
}

void blanco_4000K() {
    send_one();
    send_data(dir_rojo);
    send_data(254);
    send_stop();
    send_one();
    send_data(dir_verde);
    send_data(245);
    send_stop();
    send_one();
    send_data(dir_azul);
    send_data(200);
    send_stop();
}

void blanco_2700K() {
    send_one();
    send_data(dir_rojo);
    send_data(254);
    send_stop();
    send_one();
    send_data(dir_verde);
    send_data(204);
    send_stop();
    send_one();
    send_data(dir_azul);
    send_data(154);
    send_stop();
}

void fade_up() {
    unsigned char m;
    for (m = 0x00; m < niv; m++) {
        send_one();
        send_data(dir);
        send_data(m);
        send_stop();
    }
}

void fade_down() {
    unsigned char i;
    for (i = niv; i > 0x00; i--) {
        send_one();
        send_data(dir);
    }
}

```

```

        send_data(i);
        send_stop();
    }
    //dali_init();
}

void apagar() {
    dir = dir_verde;
    niv = 0x01;
    fade_down();
    dir = dir_rojo;
    niv = 0x01;
    fade_down();
    dir = dir_azul;
    niv = 0x01;
    fade_down();
}

void step_up(unsigned char dir) {
    send_one();
    send_data(dir + 1);
    send_data(0x03);
    send_stop();
}

void step_down(unsigned char dir) {
    send_one();
    send_data(dir + 1);
    send_data(0x04);
    send_stop();
}

void enviar_color(unsigned char rojo, unsigned char verde, unsigned
char azul) {
    //apagar();
    if (rojo > 254)
        rojo = 254;
    if (azul > 254)
        azul = 254;
    if (verde > 254)
        verde = 254;
    send_one();
    send_data(dir_rojo);
    send_data(rojo);
    send_stop();
    send_one();
    send_data(dir_verde);
    send_data(verde);
    send_stop();
    send_one();
    send_data(dir_azul);
    send_data(azul);
    send_stop();
}

```