Universidad Nacional de Colombia

Facultad de Ingeniería y Arquitectura
Programa de Ingeniería Electrónica – Sede Manizales

Sistema de Control de Sala (Room Control)

Basado en Microcontroladores ARM Cortex-M4 STM32L4

Proyecto en C bare-metal sobre la placa NUCLEO-L476RG. El sistema combina interrupciones, UART y control PWM en una arquitectura modular basada en ISR y aplicación.

Elaborado por:
Juan Pablo Vargas Córdoba

Asignatura: Estructuras de Computación
Docente: Samuel Andrés Cifuentes

Manizales, Colombia
Octubre de 2025

1. Introducción

El **Sistema de Control de Sala (Room Control)** es un proyecto diseñado para simular el funcionamiento de un controlador de iluminación en una habitación. Su propósito es demostrar el uso de microcontroladores como unidades de control embebido capaces de regular la intensidad de una luz mediante técnicas de modulación por ancho de pulso (*Pulse Width Modulation, PWM*). En este caso, el sistema utiliza un LED para representar una bombilla cuyo brillo puede variar de manera gradual o por comandos específicos.

El sistema se implementa sobre la placa de desarrollo **NUCLEO-L476RG**, basada en el microcontrolador **ARM Cortex-M4** de la familia STM32L4. Esta plataforma permite trabajar a nivel *bare-metal*, es decir, directamente sobre el hardware sin un sistema operativo intermedio, lo cual ofrece un control total sobre los periféricos y las rutinas de interrupción. Mediante la comunicación serial **UART**, el usuario puede enviar comandos desde un monitor de puerto serie (como PuTTY o el monitor serial de VS Code) para ajustar el nivel de brillo del LED o cambiar el estado general del sistema.

El proyecto está estructurado en módulos que separan claramente las funciones de hardware y de aplicación. Los controladores de bajo nivel (*drivers*) manejan los periféricos esenciales: temporizador SysTick, interrupciones NVIC, comunicación UART y temporizador TIM3 para la generación de la señal PWM. Sobre esta base se ejecuta una capa de aplicación que implementa la lógica de control, incluyendo la lectura del botón físico, el manejo de interrupciones y la interpretación de comandos recibidos por UART.

En conjunto, el sistema ofrece una representación sencilla pero completa de un **controlador de iluminación inteligente**, integrando interacción por hardware (botón y LED) y software (máquina de estados y parser de comandos). De esta forma, el proyecto combina conceptos de programación embebida, arquitectura modular y diseño de sistemas de control, acercando al estudiante a la práctica real del desarrollo con microcontroladores.

2. Hardware Utilizado

El sistema fue desarrollado sobre la placa de evaluación **NUCLEO-L476RG**, perteneciente a la familia de microcontroladores **STM32L4** de STMicroelectronics. Esta plataforma integra un microcontrolador **ARM Cortex-M4 de 32 bits**, diseñado para aplicaciones de bajo consumo energético y alta eficiencia en tareas de control embebido. Su arquitectura permite ejecutar instrucciones de punto flotante y manejar interrupciones de manera rápida y eficiente, lo que la convierte en una herramienta ideal para proyectos académicos y de prototipado.

Placa NUCLEO-L476RG

La placa cuenta con una amplia gama de periféricos integrados, entre ellos:

- Un **temporizador SysTick** que genera interrupciones periódicas de 1 ms, empleado para medir tiempos y generar retardos en el sistema.

- Un **controlador NVIC (Nested Vectored Interrupt Controller)**, encargado de gestionar las interrupciones del microcontrolador, permitiendo la atención rápida a eventos como la pulsación del botón o la recepción de datos por UART.
- Un **módulo USART2** configurado para la comunicación serial con el computador, facilitando el envío y recepción de comandos desde un monitor serial a 115200 baudios.
- Un **temporizador TIM3** utilizado en modo PWM para controlar la intensidad luminosa de un LED que simula una bombilla principal.

Elementos de interfaz

El proyecto hace uso de los periféricos integrados en la placa y algunos elementos simulados mediante LEDs:

- **LED PA5 (Heartbeat):** Representa el estado general del sistema. En el modo de reposo (IDLE) parpadea cada 0.5 s, simulando que el sistema está activo pero sin actividad. Cuando el botón o un comando UART activan el modo ocupado (OCCUPIED), el LED se mantiene encendido de manera continua durante tres segundos.
- **LED PA6 (PWM):** Simula la *bombilla principal* controlada por modulación de ancho de pulso (PWM). Su brillo varía según los comandos recibidos por UART, permitiendo ajustar la intensidad desde 0 % hasta 100 %. **Nota de conexión:** LED *externo* en *PA6 (D12)* $\rightarrow$ *resistencia 220–330* Ω $\rightarrow$ *LED* $\rightarrow$ *GND*. El pin **PA6** opera en **función alternativa AF2 (TIM3_CH1)** para generar el PWM.
- **Botón de usuario (PC13):** Actúa como entrada física para cambiar el estado del sistema. Cada pulsación válida (con antirrebote por software) alterna entre los estados de reposo y ocupado, además de reiniciar el temporizador de 3 s.
- **Interfaz UART:** La comunicación serial se establece entre el microcontrolador y el computador a través del puerto USART2 (pines PA2 y PA3). Esto permite que el usuario envíe comandos en texto, tales como `PWM 50`, `B7`, o `H`, para modificar el brillo del LED o cambiar el estado del sistema.

Resumen funcional

En conjunto, el hardware proporciona una estructura completa para la implementación del sistema de control:

- **Entrada:** Botón de usuario (evento físico) y comunicación UART (evento lógico).
- **Procesamiento:** Microcontrolador STM32L476RG con temporizadores e interrupciones configuradas.
- **Salida:** LEDs PA5 y PA6, que representan el estado general y el nivel de iluminación de la “sala”.

Esta configuración permite reproducir el comportamiento de un sistema de iluminación inteligente a pequeña escala, donde tanto la interacción física como los comandos digitales influyen en el estado del entorno simulado.

3. Funcionalidades

El sistema **Room Control** integra múltiples funcionalidades orientadas a simular un controlador de iluminación moderno. Estas permiten tanto la interacción física mediante el botón de usuario, como el control digital por comandos UART. A continuación, se describen las principales funciones implementadas en el sistema.

3.1. Heartbeat LED

El **LED de heartbeat** (ubicado en el pin PA5) tiene como propósito indicar que el sistema está operativo, aun cuando no exista interacción del usuario. Cuando el sistema se encuentra en el estado de **reposo (IDLE)**, este LED parpadea con una frecuencia de 0.5 segundos, lo que simboliza que el microcontrolador se encuentra en funcionamiento y atendiendo el bucle principal de eventos. Este parpadeo es controlado por el temporizador **SysTick**, que genera interrupciones cada milisegundo. La función `SysTick_Handler()` utiliza un contador interno (`ms_counter`) para alternar el estado del LED cada 500 ms.

Además de servir como indicador de actividad, el heartbeat cumple una función de diagnóstico: si el LED deja de parpadear, es señal de que el sistema ha entrado en un estado de falla o está detenido por depuración.

3.2. Control de Bombilla por Botón

El **botón de usuario (PC13)** permite alternar manualmente el estado del sistema entre los modos **IDLE** y **OCCUPIED**. Cada vez que el botón es presionado, se genera una interrupción externa (EXTI13) que es capturada por el controlador NVIC. La rutina `EXTI15_10_IRQHandler()` valida la pulsación con una rutina de *debounce* de 50 ms para evitar lecturas falsas por rebote mecánico.

Cuando la interrupción es válida, el sistema realiza las siguientes acciones:

- Cambia de estado: si estaba en IDLE pasa a OCCUPIED, y viceversa.
- Enciende el **LED PA5** durante 3 segundos, simulando la bombilla principal de la sala.
- Envía un mensaje por UART notificando el cambio de estado, por ejemplo:
  - `ROOM → OCCUPIED (button)`
  - `ROOM → IDLE (button)`

El tiempo de encendido del LED se controla mediante el temporizador SysTick. Si durante esos 3 segundos ocurre una nueva interacción (botón o comando UART), el temporizador se reinicia, manteniendo el LED encendido mientras haya actividad.

3.3. Comunicación UART

La comunicación serial se implementa a través del módulo **USART2** (pines PA2 y PA3) a una velocidad de **115200 baudios** y con fin de línea tipo **CRLF**. Esta interfaz permite al usuario interactuar con el sistema mediante comandos escritos desde un monitor serial, como el incluido en VS Code o PuTTY.

El sistema incluye un mecanismo de *eco de caracteres*, lo que significa que cada carácter enviado por el usuario es devuelto por la UART para confirmación visual. Además, el parser implementado en `room_control.c` reconoce distintos comandos para modificar el comportamiento del sistema:

- `H` o `h`: Enciende la bombilla principal (PWM = 100 %).
- `L` o `l`: Apaga la bombilla (PWM = 0 %).
- `B + dígito (0{9)`: Ajusta el brillo en pasos de 10 %, por ejemplo `B5` = 50 %.
- `PWM n`: Permite establecer el valor exacto del ciclo de trabajo, donde `n` puede variar entre 0 y 100.

El sistema responde a cada comando con mensajes informativos, como `OK PWM` o `ERR rango (0..100)`, indicando si la instrucción fue válida. Cualquier comando recibido reinicia el temporizador de inactividad de 3 s, impidiendo que el sistema vuelva al modo IDLE mientras haya actividad reciente.

3.4. Control PWM de Bombilla

El control del brillo del LED (simulando una bombilla de sala) se logra mediante la técnica de **Modulación por Ancho de Pulso (PWM)**. Este método consiste en variar el tiempo durante el cual la señal permanece en nivel alto dentro de un periodo constante, lo que cambia la cantidad de energía promedio que recibe el LED, afectando su luminosidad percibida.

El periférico **TIM3_CH1** del STM32L476RG se configuró en modo PWM con una **frecuencia de 1 kHz**, suficiente para evitar parpadeos perceptibles por el ojo humano. El parámetro clave del PWM es el **ciclo de trabajo (duty cycle)**, que puede variar entre 0 % (LED apagado) y 100 % (LED encendido al máximo). Los valores intermedios permiten generar distintos niveles de brillo controlados digitalmente desde la UART.

El control dinámico del duty cycle se realiza mediante la función:

```
tim3_ch1_pwm_set_duty_cycle(uint8_t valor);
```

Esta función actualiza el registro de comparación del temporizador (`CCR1`) para ajustar el ancho del pulso en tiempo real. *Se utiliza el modo PWM1 con 'preload' habilitado, lo que asegura cambios suaves y precisos en el brillo al actualizar el comparador.* Así, el usuario puede manipular la intensidad luminosa del LED desde comandos UART o de manera automática al cambiar de estado entre IDLE y OCCUPIED.

4. Arquitectura del Sistema

El **Sistema de Control de Sala (Room Control)** fue diseñado siguiendo una arquitectura modular y jerárquica que separa las responsabilidades del software en diferentes capas. Esta organización permite que cada componente tenga una función específica, mejorando la mantenibilidad, la claridad y la posibilidad de ampliación del sistema. A nivel general, el sistema se estructura en tres capas principales: *drivers*, *interrupciones (ISR)* y *aplicación*.

4.1. Módulos

Cada módulo del sistema cumple una tarea concreta dentro del proceso de control. A continuación, se describen brevemente los más relevantes:

- **rcc.c / rcc.h:** Inicializa y habilita los relojes del sistema (*Reset and Clock Control*), permitiendo el funcionamiento de los periféricos principales (GPIO, UART, TIM, SYSCFG).
- **gpio.c / gpio.h:** Configura los pines del microcontrolador como entradas, salidas o funciones alternativas. Se utiliza para controlar el LED de estado (PA5), el LED PWM (PA6) y leer el botón de usuario (PC13).
- **systick.c / systick.h:** Implementa un temporizador de referencia de 1 ms mediante interrupciones periódicas. Este módulo permite llevar el conteo del tiempo, realizar retardos y gestionar eventos temporizados (por ejemplo, el apagado automático del LED).
- **nvic.c / nvic.h:** Configura y habilita las interrupciones externas, en especial la línea EXTI13 para el botón de usuario y la interrupción USART2 para la comunicación serial.
- **uart.c / uart.h:** Controla la transmisión y recepción de datos por el puerto serial. Incluye funciones de envío de cadenas de texto y de eco de caracteres, utilizadas para interactuar con el usuario desde el monitor serial.
- **tim.c / tim.h:** Configura el temporizador TIM3 en modo *Pulse Width Modulation (PWM)*, responsable del control de brillo del LED principal (PA6) en función de los comandos recibidos.
- **isr.c:** Contiene las rutinas de servicio a interrupciones (*Interrupt Service Routines, ISR*), encargadas de responder a eventos asincrónicos como pulsaciones de botón o recepción de datos por UART.
- **room_control.c / room_control.h:** Implementa la **lógica de aplicación** del sistema, incluyendo la máquina de estados (*IDLE* y *OCCUPIED*), el parser de comandos UART y las acciones de control del brillo mediante PWM.
- **main.c:** Actúa como punto de entrada del programa. Inicializa los periféricos, ejecuta el bucle principal de eventos y coordina la comunicación entre las capas de hardware e interfaz de usuario.

Cada módulo se comunica únicamente con los componentes necesarios, evitando dependencias cruzadas innecesarias. Esta organización facilita el mantenimiento del código y promueve la reutilización en futuros proyectos embebidos.

4.2. Flujo de Ejecución

El flujo de ejecución del sistema sigue una estructura basada en eventos, en la que las interrupciones activan las acciones correspondientes sin necesidad de bloquear el programa principal. A continuación, se describe el proceso general:

1. **Inicialización del sistema:** Al iniciar el programa, la función `main()` ejecuta las rutinas de configuración de relojes (`rcc_init()`), GPIO, SysTick, UART, NVIC y PWM. Una vez que todos los periféricos están listos, se inicializa la aplicación con `room_control_app_init()` y se envía el mensaje `\Sistema Inicializado"` por UART.

2. **Bucle principal (*main loop*):** El programa entra en un bucle infinito que ejecuta de forma periódica la función `room_control_update()`. Esta función evalúa el estado actual del sistema, procesa los eventos generados (botón o UART) y actualiza la salida PWM o los LEDs según corresponda. En esta etapa no hay retardos activos; el flujo depende completamente de las interrupciones.

3. **Gestión de interrupciones:** Las interrupciones son el núcleo del control del sistema:

   - **SysTick:** Incrementa el contador global `ms_counter` cada milisegundo y gestiona el parpadeo del LED de heartbeat, así como los temporizadores de apagado automático.
   - **EXTI13:** Detecta pulsaciones del botón, aplica *debounce* y cambia el estado entre IDLE y OCCUPIED.
   - **USART2:** Recibe los caracteres enviados desde el monitor serial, los entrega al parser de comandos y genera respuestas inmediatas al usuario.

4. **Actualización de salidas:** Según el estado del sistema y los comandos recibidos, se actualiza el valor del ciclo de trabajo del PWM (TIM3_CH1) para controlar el brillo del LED PA6. De igual forma, el LED PA5 refleja el estado global del sistema (parpadeando o encendido fijo).

5. **Timeout de inactividad:** Si no se detecta ninguna interacción durante 3 segundos, el sistema vuelve automáticamente al estado IDLE, apaga el LED principal y retoma el parpadeo del heartbeat.

Esta secuencia garantiza que el sistema permanezca siempre activo, responda en tiempo real a los eventos externos y mantenga una operación fluida sin necesidad de un sistema operativo. La arquitectura basada en interrupciones y el bucle de eventos hacen que el sistema sea eficiente, determinista y fácilmente escalable a funcionalidades más avanzadas.

5. Uso del Sistema

El presente sistema fue diseñado para ser de fácil uso y comprensión, permitiendo al usuario interactuar tanto de forma física mediante un botón, como de forma digital a través de comandos UART. A continuación, se describen los pasos necesarios para su correcta conexión, puesta en marcha e interacción.

5.1. Conexión

Para ejecutar el sistema, se requiere la placa **NUCLEO-L476RG** conectada a un computador mediante un cable **USB tipo A–Micro B**. El entorno de conexión y comunicación debe configurarse de la siguiente forma:

- **Puerto Serie:** La conexión UART se realiza a través del convertidor USB integrado en la placa (ST-Link). En el computador, esta interfaz aparece como un puerto COM virtual.
- **Velocidad de comunicación:** 115200 baudios.
- **Formato de datos:** 8 bits de datos, sin paridad, 1 bit de parada (`8N1`).
- **Fin de línea:** Carácter de retorno de carro y salto de línea (`CR+LF`).

Para visualizar la comunicación, puede emplearse un programa de monitoreo serial, como:

- El *Serial Monitor* integrado en VS Code.
- **PuTTY**, configurado con el puerto COM correspondiente.
- **Tera Term** o cualquier otro cliente compatible con UART.

Para observar el PWM en hardware, conecte un LED *externo* en *PA6 (D12)* con una resistencia serie de 220–330 Ω hacia GND.

Una vez conectada la placa, el microcontrolador se alimenta directamente desde el puerto USB (5 V), por lo que no se requiere fuente de alimentación adicional. El LED integrado en la placa (LD2 – PA5) debe comenzar a parpadear, indicando que el sistema está en funcionamiento.

5.2. Inicio del Sistema

Al encender o reiniciar la placa, el microcontrolador ejecuta las rutinas de inicialización de los periféricos esenciales: relojes, GPIO, temporizadores, UART y controlador de interrupciones (NVIC). Durante esta fase, se envía el mensaje inicial por el puerto serial:

```
Sistema Inicializado!
```

Inmediatamente después, el sistema entra en el estado de **reposo (IDLE)**, en el cual:

- El LED **PA5 (heartbeat)** parpadea cada 0.5 segundos, señalando que el sistema está activo.
- El LED **PA6 (PWM)** permanece apagado.
- El sistema está preparado para recibir pulsaciones del botón o comandos UART.

Esta fase confirma que la comunicación serial, las interrupciones y los periféricos han sido configurados correctamente.

5.3. Interacción con el Sistema

El usuario puede interactuar con el sistema mediante dos vías principales:

1. Control físico — Botón de usuario (PC13)

El botón integrado en la placa funciona como un interruptor de modo:

- Una pulsación válida cambia el estado del sistema.
- Si el sistema está en IDLE, pasa a **OCCUPIED**: el LED PA5 se enciende de forma continua durante 3 segundos y el LED PWM se ajusta al 100 %.
- Si el sistema está en OCCUPIED, vuelve a IDLE: el LED PA5 retoma su parpadeo regular y el PWM se reduce a 0 %.

Cada cambio de estado se notifica mediante mensajes enviados por UART, como:

`ROOM → OCCUPIED (button)` `ROOM → IDLE (button)`

2. Control digital — Comunicación UART

El sistema también puede controlarse desde el computador mediante el envío de comandos en texto. Estos comandos permiten manipular directamente el brillo del LED PWM (PA6) o cambiar el estado general del sistema. A continuación, se listan los comandos disponibles:

- `H / h`: Fija el brillo al 100 %.
- `L / l`: Apaga el LED (0 %).
- `B + dígito (0{9)`: Ajusta el brillo en pasos de 10 %.
- `PWM n`: Establece manualmente el valor del duty cycle de 0 a 100.

Por ejemplo:

- `PWM 25` → el LED se ilumina al 25 % de su brillo máximo.
- `B7` → el LED se ilumina al 70 %.

Cada comando recibido se confirma mediante una respuesta por el monitor serial:

`OK PWM` o `ERR rango (0..100)`

El sistema está programado para mantener el estado de iluminación durante 3 segundos después de cada acción. Si no se detecta actividad (botón o comando UART) en ese intervalo, el sistema retorna automáticamente al estado IDLE, apagando la bombilla simulada y reanudando el parpadeo del LED heartbeat.

6. Diagramas

En esta sección se presentan los diagramas principales que describen el comportamiento lógico y la organización interna del **Sistema de Control de Sala**. Ambos diagramas permiten comprender tanto la secuencia de funcionamiento (máquina de estados) como la relación entre los distintos módulos de software y hardware.

6.1. Diagrama de Estados

El diagrama de la Figura 1 representa la **máquina de estados** que gobierna el sistema. El flujo inicia en el estado **REPOSO (IDLE)**, en el cual el LED PA5 parpadea periódicamente indicando que el sistema está activo pero sin actividad reciente. Cuando el usuario presiona el botón o envía un comando UART válido, el sistema cambia al estado **OCUPADO (OCCUPIED)**, en el que el LED PA5 permanece encendido fijo durante 3 segundos y el PWM (PA6) se ajusta según el brillo solicitado. Si no se detecta ninguna interacción durante 3 segundos, el sistema regresa automáticamente a REPOSO.

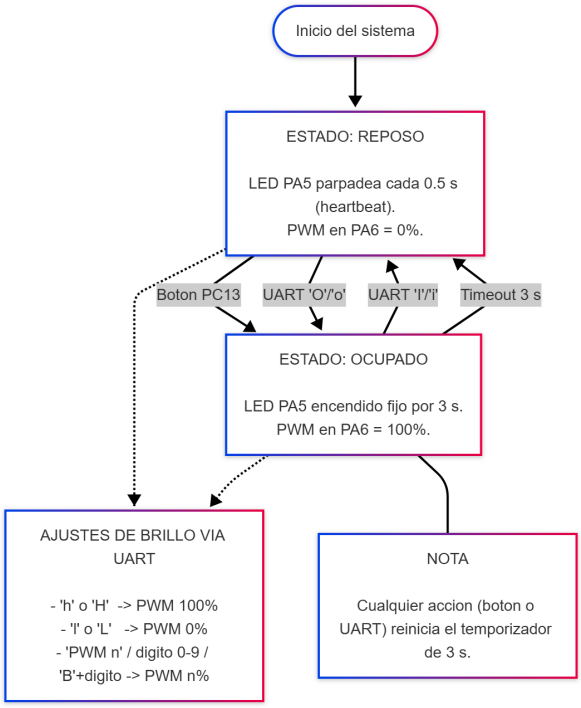


Figura 1: Diagrama de estados del sistema Room Control, mostrando las transiciones entre los modos de operación.

6.2. Diagrama de Componentes

La Figura 2 muestra la **arquitectura del software**, organizada de manera jerárquica desde las entradas hasta las salidas. En la parte superior se ubica el usuario (interfaz UART y botón físico), seguido por las interrupciones que gestionan los eventos asincrónicos. La capa de aplicación, implementada en `main.c` y `room_control.c`, coordina la lógica principal del sistema y comunica los módulos de *drivers* con el hardware. Finalmente, la capa de salida controla los periféricos de iluminación (LED PA5 y PA6) y la gestión del reloj interno.

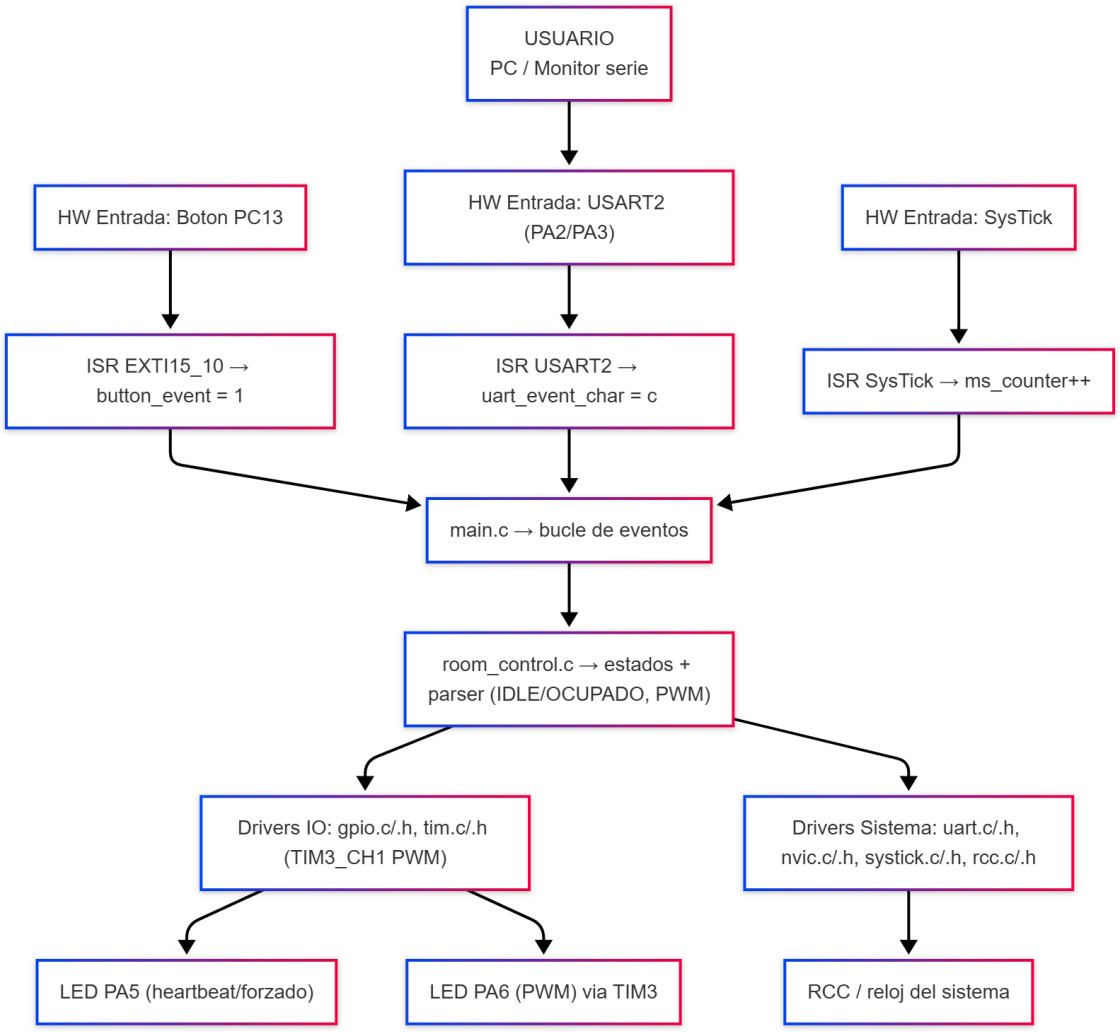
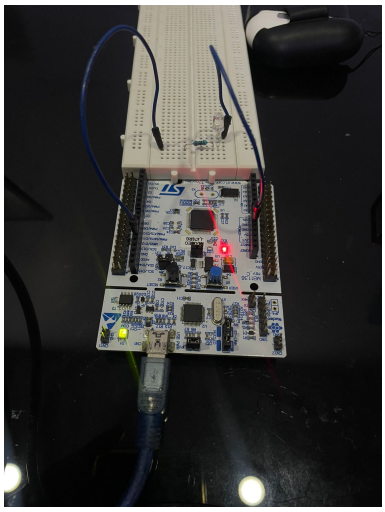


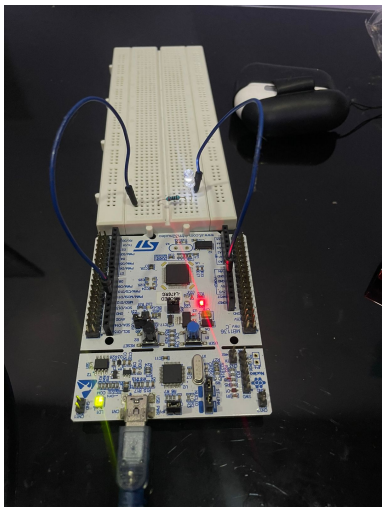
Figura 2: Diagrama de componentes del sistema, mostrando la estructura jerárquica y las relaciones entre módulos.

Anexo: Evidencia visual de niveles PWM

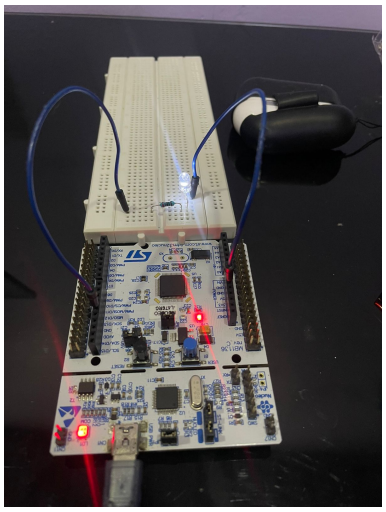
A continuación se muestran fotografías del LED externo conectado a PA6 (D12) con diferentes ciclos de trabajo (duty). Todas las fotos se tomaron con la misma exposición y condiciones de iluminación para evidenciar el cambio de brillo.



0 % (`L` / `PWM 0`)



10 % (`B1` / `PWM 10`)



100 % (`H` / `PWM 100`)

Figura 3: Evidencia fotográfica de tres niveles de PWM en el LED externo (PA6–D12).

7. Conclusión

El desarrollo del **Sistema de Control de Sala (Room Control)** permitió integrar de manera práctica los conceptos fundamentales de la programación embebida sobre microcontroladores ARM Cortex-M4. A través de la implementación en **C bare-metal**, fue posible comprender el funcionamiento interno de los periféricos del STM32L476RG, especialmente el temporizador SysTick, el controlador NVIC, la comunicación UART y la generación de señales PWM mediante el módulo TIM3.

El proyecto no solo demostró el control de un LED como elemento de salida, sino también la importancia de estructurar el software en capas —drivers, interrupciones y aplicación— para lograr un sistema modular, mantenible y fácilmente ampliable. Gracias a la comunicación UART y a la máquina de estados propuesta, el sistema respondió en tiempo real tanto a las pulsaciones del botón físico como a los comandos enviados desde el computador, mostrando un comportamiento estable y determinista.

En términos de aprendizaje, este proyecto consolidó la comprensión del manejo directo de registros, la configuración de periféricos y la relación entre hardware y software dentro de un sistema embebido. Además, evidenció la relevancia de los manuales técnicos, especialmente el *Reference Manual RM0351*, para interpretar correctamente la arquitectura del microcontrolador, así como del texto de *Yifeng Zhu*, que orienta el diseño estructurado de programas sobre la familia STM32.

Referencias

- STMicroelectronics. **RM0351: STM32L4x5 and STM32L4x6 advanced Arm®-based 32-bit MCUs Reference Manual**. Rev 9, July 2023.
- Zhu, Yifeng. **Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C: Third Edition**. E-Man Press LLC, 2017.