

# **MEMORIA DAD 2023-2024**

Marina Delgado Trinidad

Younes Yousra Sierra

**ENLACE GITHUB:**

**[https://github.com/YounesYS/Entregables\\_DAD](https://github.com/YounesYS/Entregables_DAD)**

## **LUZ INTELIGENTE**





## INDICE

1. DESARROLLO DEL PROYECTO .....	3
2. OBJETIVO .....	4
3. DINÁMICA.....	5
4. DIFICULTADES .....	6
5. CONCLUSION .....	7




## DESARROLLO DEL PROYECTO

En este proyecto, hemos tratado de implementar todos los conocimientos adquiridos en la asignatura “**Desarrollo de Aplicaciones Distribuidas**” para poder así, desarrollar un proyecto que nos permita automatizar labores de nuestra vida cotidiana.


Para ello hemos hecho uso de los siguientes entornos:

1. Eclipse IDE for Enterprise Java and Web Developers 
2. Visual Studio Code 
3. Heidi SQL 

Junto con los siguientes lenguajes de programación:

1. Java 
2. C++ 
3. MySQL 

Extensión que hemos añadido en VS Code para añadir a nuestro proyecto las librerías necesarias:

1. Platformio 

Y por último tenemos:

1. **MQTT Explorer:** Sistema de back-end que coordina los mensajes entre los diferentes clientes



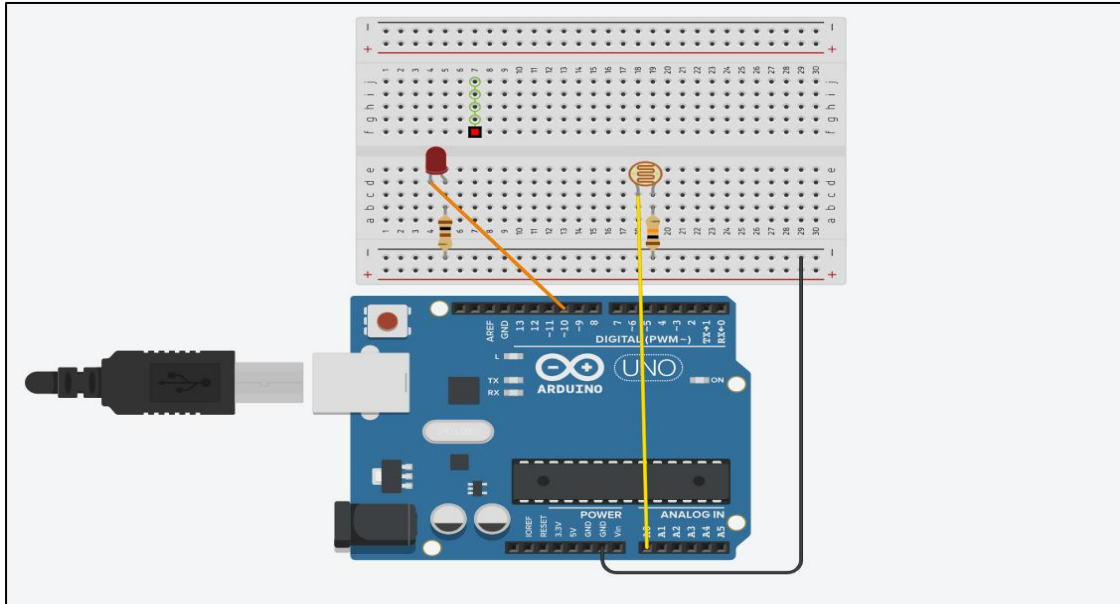
2. **Mosquitto:** Broker MQTT OpenSource ampliamente utilizado que debido a su ligereza permite emplearlo en ámbitos de diversa índole, incluso si éstos son de pocos recursos.



## OBJETIVO

Nuestro principal objetivo es el de automatizar aspectos de nuestra vida cotidiana, más específicamente, nos hemos centrado en el desarrollo de una luz inteligente que se activa cuando detecta que no hay la suficiente luminosidad como para permitir una visión clara, además de atenuarse en caso de que haya suficiente luz, pero queramos claridad extra.

Nuestro proyecto consta de los siguientes componentes:



### Esquema en tinkercad

- LDR:** Sensor conectado a un pin analógico-digital de 12 bits.
- LED:** Actuador conectado a un pin analógico-digital con señal PWM, que nos ha permitido simular un comportamiento analógico en la salida del LED.
- RESISTENCIAS:** De pull-down, de  $220\Omega$  para el led y  $10k\Omega$  para LDR.

En nuestro proyecto de luz inteligente esto se traduce en que en base a la luminosidad que registre el LDR, podemos hacer que el actuador se encienda y que además brille con una intensidad inversamente proporcional y variable. Entiéndase como variable un rango de valores mucho mayor que si usáramos el LED en un pin digital(0 o 1) .

## DINÁMICA DEL PROYECTO:

1. El sensor LDR capta de forma continuada la luminosidad del ambiente.
2. Este valor se serializa y se envía a la base de datos.
3. La ApiRest consulta la base de datos y en concreto, mira la última entrada, donde el orden viene dado por el valuelid.
4. Del último valor que envió el sensor LDR, realizamos un mapeo para mandar al LED la intensidad con la que debe brillar. Este mapeo no sólo es necesario porque sean inversamente proporcionales, sino también porque el LDR está conectado a un pin analógico de 12 bits con rango [0, 4095] y el pin PWM del LED es de 8 bits con rango [0, 255]. Concretamente este cálculo lo hacemos como:

$$\text{i. } 255 - \text{ultimoValorLdr} * 255 / 4096$$

5. La API se encarga de publicar en MQTT, en el canal que coincide con el groupId, que es el mismo topic donde está suscrita la placa, el último valor recibido del LDR y el mapeo resultante del brillo del led.
6. La placa deserializa lo que envía la API y realizamos el cálculo de dos umbrales:
  - a. Cálculo del umbral de luminosidad.
  - b. Cálculo del umbral del LED.

Estos umbrales son el criterio que tiene el ESP-32 para determinar qué hacer sobre el actuador. Es en función del umbral que establecemos para el LED y del valor resultante del mapeo, que sabemos si el LED estaba encendido o apagado y cuál será su próximo estado. En caso de que se produzca un cambio en el estado del LED, se realizará un POST a la base de datos.

7. Actualizamos la variable que contiene el estado del led.
8. Repetición del proceso.

## **DIFICULTADES**

Afrontar este proyecto no ha sido tarea fácil debido a varios factores.

En primer lugar, los conceptos tenían que estar bien asentados para poder iniciar el proyecto, tarea nada fácil debido a los conceptos abstractos de las tecnologías, especialmente **MQTT** y **mosquitto**, puesto que eran nuevas para nosotros. Aun así, gracias a las pautas del profesor y a búsqueda extra por nuestra parte, logramos familiarizarnos con las tecnologías mencionadas.

Durante la implementación, nos encontramos con dificultades en lo relacionado al circuito. El mal contacto del conexionado de la placa y la protoboard ha sido una constante junto a la sensibilidad del LDR que no proporciona muy buenas prestaciones. No obstante, tras varios ajustes, conseguimos un funcionamiento relativamente bueno. Como consecuencia, es casi imperceptible la variación del brillo del led dependiendo de la luminosidad captada en el LDR. Para solventar dicho problema, hemos imprimido por pantalla los valores del LED para comprobar que efectivamente variaba su rango de 0 a 255

Pero la dificultad de mayor peso para el desarrollo y avance del proyecto ha sido, sin duda, la lógica que sigue la API para mandar sobre el firmware. Inicialmente, nuestra lógica era otra. En concreto, lo que enviábamos a la placa era el último valor del LDR del que se hizo POST y se pasaba tal cual para luego hacer el mapeo en el firmware. Pero nos encontramos con que:

1. **Limitaciones de los pines ADC2:** Al utilizar PWM en un pin analógico (ADC2), emplear digitalRead para determinar el estado del LED no era posible. Usar analogRead tampoco fue viable, ya que estos pines son compartidos con el módulo WiFi, lo que interfería con la comunicación y no nos dejaba hacer nuevos POST.
2. **Restricciones de los pines ADC1:** Ante esto pensamos en usar el otro tipo de pines analógico-digitales, los ADC1, pero estos no cuentan con señal PWM, por lo que el objetivo del proyecto no iba a ser posible alcanzarlo.

La solución alternativa que le hemos dado ha resultado en lo que tenemos actualmente, que ha sido calcular el brillo del LED en la APIRest para luego serializarlo y mandarlo a la placa, de tal forma que sólo tenemos que controlar el comportamiento del actuador con los umbrales.

## **CONCLUSIÓN:**

Tal y como nos propusimos, hemos implementado un servicio de luz inteligente que se acomoda a la luminosidad del ambiente en el que coloquemos el sistema y es que no es más, que la simulación de la luz en una estancia.

Realizar este proyecto nos ha parecido todo un reto y una muy buena oportunidad para aprender sobre nuevas tecnologías y entender cómo se fusiona el desarrollo software con lo hardware.

Además, gracias precisamente a estos conocimientos, nuestro proyecto podría ir incluso a más y convertirse en algo aún más completo, como podría ser una funcionalidad especial para la noche, donde la luz se active o se apague cuando detecte que alguien se encuentra en la estancia.

