

**INSTITUTO TECNOLÓGICO  
AUTÓNOMO DE MÉXICO**



---

Laboratorio de Principios de Mecatrónica  
**Práctica 1. Microcontroladores y Señales Digitales**

Gutiérrez Zapién, Mariel Sofía  
Gómez Cantú, Danya Carolina  
Hernández Alás, Nuria

Asignatura: Laboratorio de Principios de Mecatrónica  
Docente: M.I. Sergio Hernández Sánchez

Grupo: 01

Semestre: Otoño 2023

## 1. Introducción

Los microcontroladores desempeñan un papel esencial en el mundo de la electrónica y la ingeniería. Son componentes esenciales para controlar y automatizar diversas tareas, abarcando desde electrodomésticos hasta sistemas médicos y automóviles. En esta práctica, exploraremos las capacidades del microcontrolador ESP32, una potente plataforma de desarrollo que ofrece una amplia gama de opciones para proyectos mecatrónicos. A lo largo de esta práctica, diseñaremos y programaremos tres aplicaciones diferentes utilizando LEDs como componentes de salida.

## 2. Objetivos

- Conocer las plataformas, herramientas y ambientes más comunes para el desarrollo de sistemas mecatrónicos basados en microcontroladores.
- Identificar los elementos básicos de la arquitectura de un microcontrolador digital.
- Identificar los módulos que componen a la tarjeta de desarrollo Arduino y ESP32 y comparar su funcionalidad.

## 3. Marco Teórico

Un microcontrolador es un circuito integrado que contiene un procesador, memoria y periféricos en un solo chip. Se utiliza para controlar dispositivos y sistemas embebidos, desde electrodomésticos hasta dispositivos médicos y automóviles. Los microcontroladores ejecutan programas almacenados en su memoria para controlar y coordinar diferentes tareas [1]. Las principales características de cada placa pueden compararse en la Tabla 1, a partir de la siguiente página.

*Tabla 1: Principales características de las placas de desarrollo ESP32 vs Arduino UNO*

<b>Característica</b>	<b>ESP32</b>	<b>Arduino UNO</b>
Microcontrolador	ESP32 de Espressif Systems Microcontrolador de doble núcleo Xtensa LX6 [2]	ATmega328P de Microchip [2]
Arquitectura	Arquitectura de 32 bits [2]	Arquitectura de 8 bits [2]
Velocidad del reloj	El ESP32 está equipado con un procesador de doble núcleo Tensilica LX6, que funciona a una velocidad de reloj de hasta 240 MHz. [2]	Funciona a velocidades de reloj más bajas, típicamente alrededor de 16 MHz. [2]
Memoria Flash	Mayor capacidad de memoria flash, que puede variar según el modelo específico, pero suele estar en el rango de varios megabytes. [3]	Memoria flash mucho más limitada en comparación con el ESP32, generalmente alrededor de 32 KB. [3]
Conectividad inalámbrica	Incorpora Wi-Fi y Bluetooth, lo que permite la comunicación inalámbrica directa. [3]	No tiene conectividad inalámbrica incorporada (se puede agregar a través de módulos externos). [3]
GPIO y E/S	El ESP32 ofrece una mayor cantidad de pines GPIO (General Purpose Input/Output) y E/S en comparación con el Arduino UNO.  Hasta 32 pines GPIO. Hasta 18 pines analógicos. Hasta 16 pines con PWM. [3]	Aunque el Arduino UNO tiene un número limitado de pines GPIO en comparación con el ESP32, todavía proporciona suficiente flexibilidad para proyectos simples y medianos.  14 pines GPIO digitales. 6 pines analógicos. 6 pines con PWM. [3]
Consumo de energía	Puede ser configurado para un consumo de energía más bajo en aplicaciones de bajo	No es tan eficiente en términos de consumo de energía. [3]

	consumo. [3]	
Soporte para desarrollo	Requiere el uso de la plataforma de desarrollo ESP-IDF o el entorno de Arduino (compatible con algunas limitaciones). [4]	Utiliza el entorno de desarrollo Arduino, que es ampliamente conocido y fácil de usar. [4]
Protocolos de comunicación	Ambas placas soportan I2C, SPI y UART (USART) para comunicación serial. [4]	Ambas placas soportan I2C, SPI y UART (USART) para comunicación serial. [4]
Configuración Pull-up y Pull-down*	Ambas placas permiten configurar pines con resistencias Pull-up y Pull-down internas.	Ambas placas permiten configurar pines con resistencias Pull-up y Pull-down internas.
Voltajes y corrientes	El voltaje de trabajo de los pines es de 3.3V, mientras que el voltaje externo que se le puede suministrar llega hasta los 12V en el pin VIN. La corriente máxima por pin es de alrededor de 12 mA. [5]	El voltaje de trabajo de los pines es de 5V, mientras que el voltaje externo que se le puede suministrar es de 7 a 12V en el conector de alimentación. La corriente máxima por pin de alrededor de 20 mA. [5]

Las resistencias *pull-up* y *pull-down* establecen un estado lógico en un pin o entrada de un circuito lógico cuando se encuentra en estado de reposo. Como bien indica su nombre la resistencia pull up establece un estado HIGH y las resistencias pull down establecen un estado LOW cuando el pin se encuentra en reposo. Esto evita los falsos estados que se producen por el ruido generado por los circuitos electrónicos [6].

#### 4. Material y equipo utilizado

- a. 1 ESP32 Dev Kit V1
- b. 1 cable USB A - micro USB
- c. 1 protoboard grande
- d. 2 LED rojo
- e. 2 LED amarillo
- f. 2 LED verde
- g. 1 Push-Button
- h. 6 resistor 100  $\Omega$
- i. 1 resistor 10 k  $\Omega$
- j. Jumpers macho - macho

## 5. Experimentos y simulaciones

### a. Actividad 1 – Blink

Para esta actividad, se busca encender y apagar un LED, para lo cual, se conectará una resistencia de  $100\ \Omega$  al ánodo de un LED y el ánodo del LED estará conectado al GPIO 15 de una ESP32 Dev Kit v1. Apóyese de la figura 1 y 2 para conectar lo que se solicita.

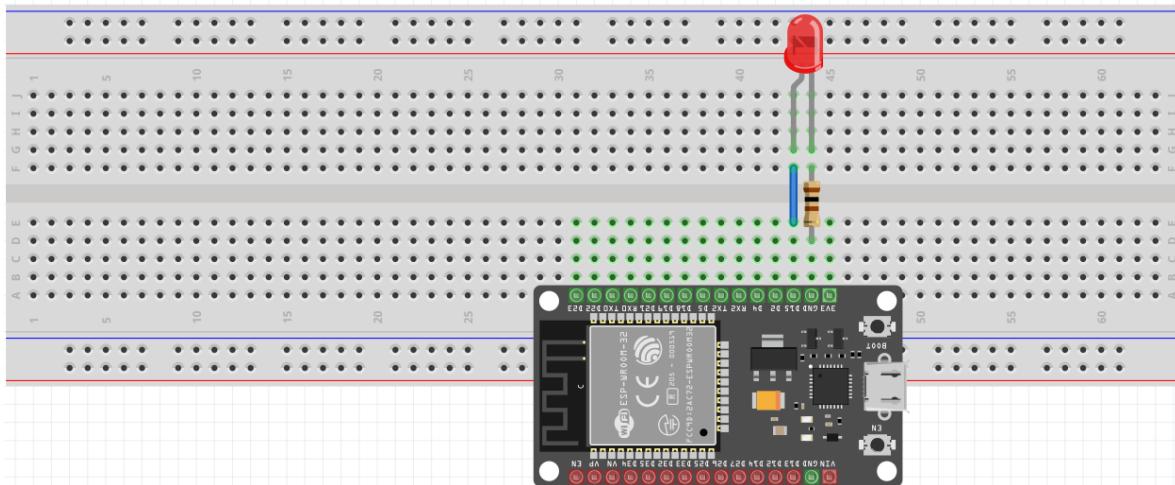


Figura 1. Circuito para encender y apagar un LED.

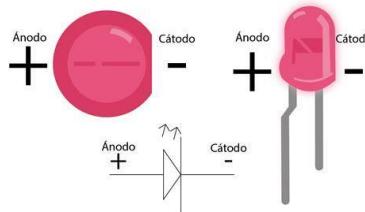
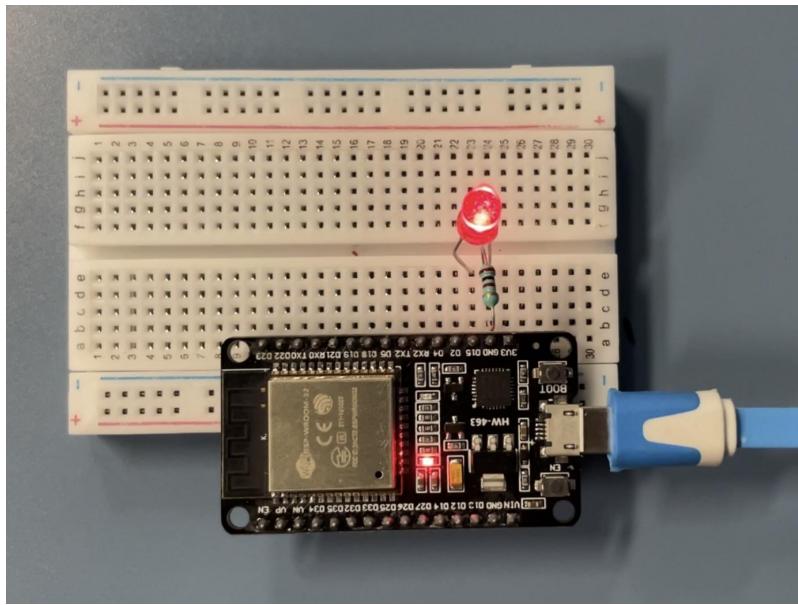


Figura 2. Identificación del ánodo y cátodo de un LED.

A continuación, coloque una fotografía del circuito armado en la figura 3 donde se muestre el LED encendido.



*Figura 3. Vista superior del circuito armado con LED.*

Enseguida, en su cuenta de GitHub, deberá crear una carpeta nombrada **P\_Mecatronica**, y dentro de ella otra carpeta cuyo nombre sea **Practica01** y guarde dentro, el código generado cuyo nombre será **PIA1-Blink**, y coloque enseguida el enlace de dicho archivo de manera que el profesor pueda verlo y evaluar su código. Este procedimiento descrito deberá hacerse para cada una de las actividades de cada práctica de laboratorio.

*Enlace a su código:*

[https://github.com/marielsgtzz/Mecatronica/tree/main/P\\_Mecatronica/Practica01/PIA1-Blink](https://github.com/marielsgtzz/Mecatronica/tree/main/P_Mecatronica/Practica01/PIA1-Blink)

El código deberá encender un LED, dejarlo encendido por 1 segundo y posteriormente apagarlo por 1 segundo. Esto se deberá repetir indefinidamente. Deberá comentar cada línea de código de tal forma que explique su funcionamiento, recuerde que para comentar una línea se pondrá doble diagonal //.

A continuación, en alguna nube que tenga disponible coloque el video de esta primera actividad o si lo prefiere, puede subirlo a YouTube de forma pública o privada de forma que pueda visualizarse el funcionamiento de todo el sistema completo. Esto deberá realizarlo en todas las actividades de todas las prácticas.

*Enlace a su video:*

[https://drive.google.com/drive/folders/1-I8ITdTPnt6EgK3NpK15Dbh\\_vNOt5NMG](https://drive.google.com/drive/folders/1-I8ITdTPnt6EgK3NpK15Dbh_vNOt5NMG)

### b. Actividad 2 – Botón y Frecuencia

Como segunda actividad, cree un nuevo sketch en el IDE de Arduino que deberá guardar como **PIA2-Boton-Freq** y guardar en su carpeta de **Practica01** de su repositorio. Con base en el circuito de la actividad 1, se pide que se añada un *push-button* en la configuración *pull-down* utilizando una resistencia de 10 kΩ de tal manera que al ser presionado, envíe una señal de un uno lógico o de 5 V a la tarjeta ESP32 y cuando no esté presionado, se envíe un cero lógico o 0 V. Utilice el diagrama de la figura 4 para poder armarlo.

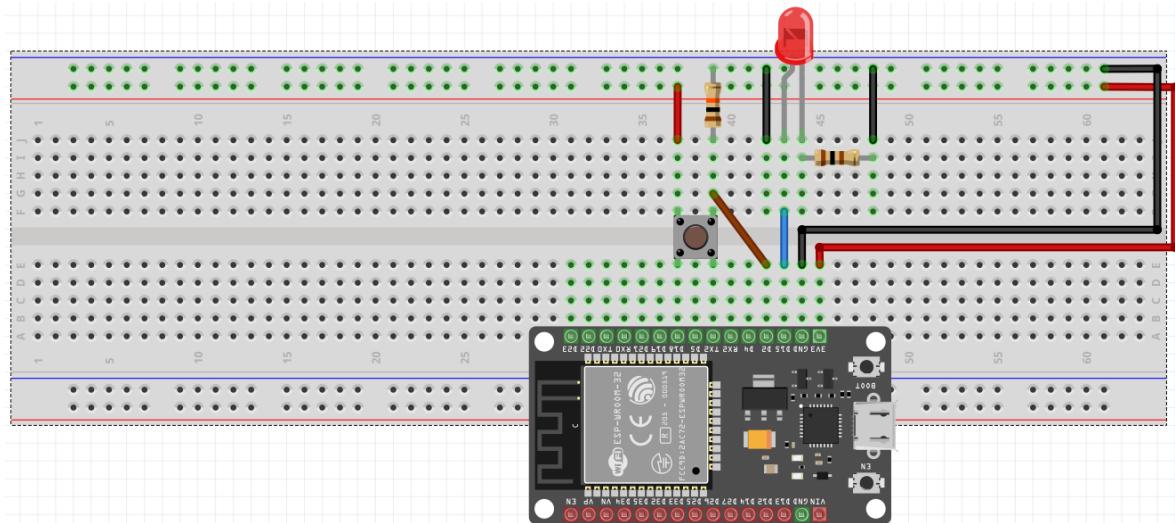
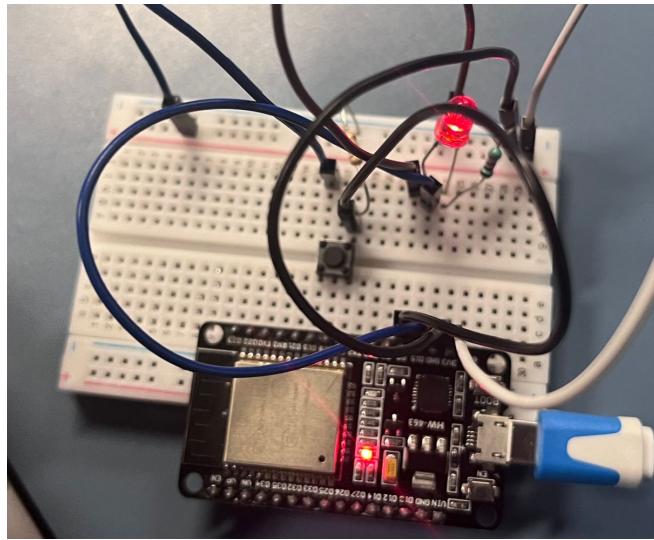


Figura 4. Circuito para probar el push-button y frecuencia de LED.

Una vez armado el circuito, genere un código en la IDE de Arduino de manera que cumpla con lo descrito en la tabla 1. Deberá comentar cada línea o cada sección de manera que se describa que está realizando a lo largo de su programa. Añada una fotografía de su circuito armado en la figura 5, el enlace de GitHub y su video de funcionamiento.

Tabla 1. Lógica que deberá seguir el LED.

Estado del Push-Button	Frecuencia de encendido del LED
LOW	1 Hz
HIGH	4 Hz



*Figura 5. Vista superior del circuito armado de control de frecuencia.*

*Enlace a su código:*

[https://github.com/marielsgtzz/Mecatronica/tree/main/P\\_Mecatronica/Practica01/P1A2-Boton-Freq](https://github.com/marielsgtzz/Mecatronica/tree/main/P_Mecatronica/Practica01/P1A2-Boton-Freq)

*Enlace a su video:*

<https://drive.google.com/drive/folders/1-I8wShO1RMNoInt5qQ4QyOUvHlr-jjzY>

### c. Actividad 3 – Semáforo

Finalmente, en esta última actividad conecte a su circuito 2 LEDs rojos, 2 LEDs amarillos y 2 verdes con sus respectivas resistencias de  $100 \Omega$ , de forma que estén acomodados como dos configuraciones de semáforos. Conecte el ánodo del LED a alguno de los pines digitales de su Arduino MEGA, como se muestra en la figura 6.

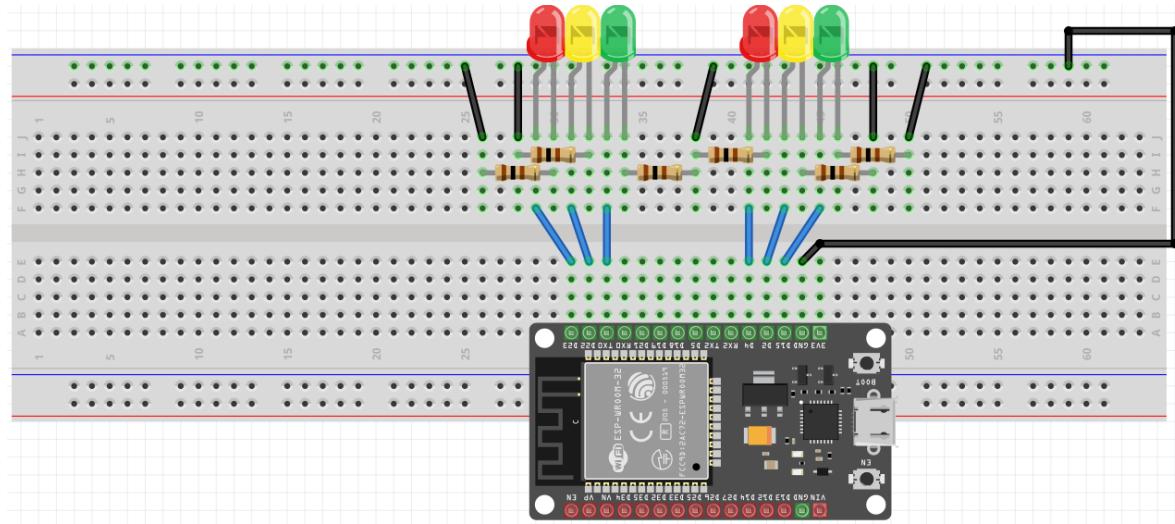


Figura 6. Circuito para probar el semáforo.

A continuación, genere el código necesario guardándolo como **P1A3-Semáforo**, que deberá seguir la lógica descrita por la figura 7 para lograr el funcionamiento de 2 semáforos. Recuerde comentar su código.

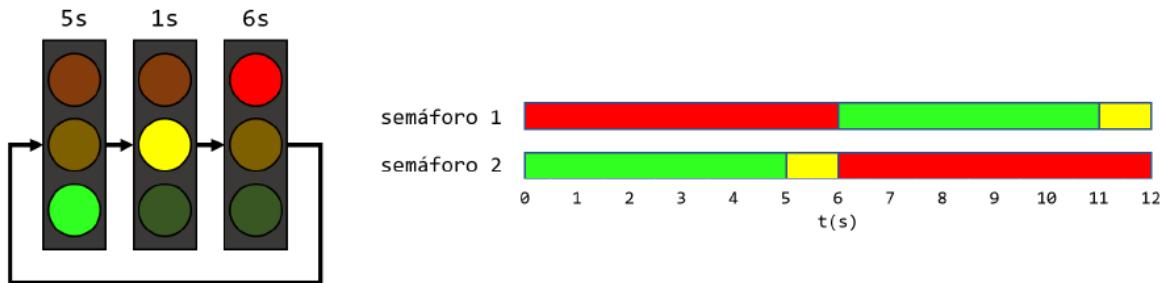
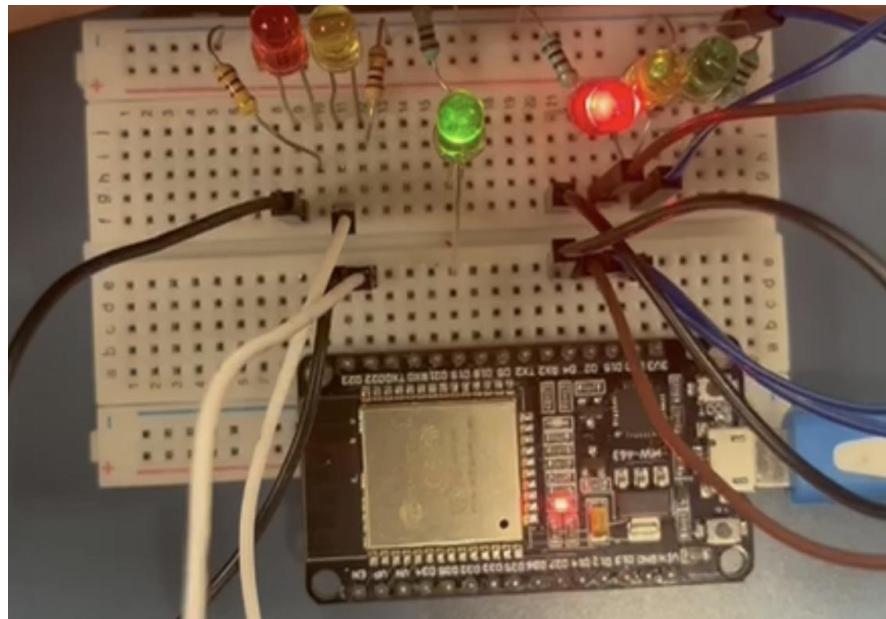


Figura 7. Lógica de encendido del semáforo.

Añada una fotografía de su circuito armado en la figura 8, el enlace de GitHub y su video de funcionamiento.



*Figura 8. Vista superior del circuito del semáforo armado.*

*Enlace a su código:*

[https://github.com/marielsgtzz/Mecatronica/tree/main/P\\_Mecatronica/Practica01/P1A3-Semaforo](https://github.com/marielsgtzz/Mecatronica/tree/main/P_Mecatronica/Practica01/P1A3-Semaforo)

*Enlace a su video:*

<https://drive.google.com/drive/folders/1-Pdi5SBjDSeAhhlBJs6-8Te5qL68p3jo>

## 6. Conclusiones

Esta práctica me permitió conocer mejor el ambiente de desarrollo de Arduino. Dado que había utilizado esta IDE hace algunos años, me resultó más sencillo familiarizarme con la arquitectura y funcionamiento de los microcontroladores de manera práctica; en este caso, del ESP32 y de Arduino MEGA. Las actividades nos permitieron entender de manera experimental el rol de estos microcontroladores en nuestro circuito, así como practicar la forma correcta de armado, e identificar las herramientas que caracterizaban a cada uno.

De manera personal, me llamó la atención como es más útil emplear el Arduino MEGA en la Actividad 3 que el ESP32, no porque su funcionalidad no sea suficiente, sino por su tamaño y los componentes adicionales del circuito (en particular, la cantidad de LEDs). Aprender por medio de la práctica nos permite identificar más fácilmente las herramientas adecuadas en el futuro; así, podemos encontrar una solución con los recursos a nuestro alcance, y aprovechar de manera más eficiente aquellos que se adapten a nuestro problema.

- Danya

En esta práctica, se exploraron las diferencias entre las placas de desarrollo ESP32 y Arduino UNO, destacando sus características clave. Se observó que el ESP32 es más poderoso en términos de velocidad de procesamiento, memoria flash y cantidad de pines GPIO y analógicos. También ofrece conectividad inalámbrica incorporada. Por otro lado, el Arduino UNO es una opción más simple y económica, adecuada para proyectos básicos y educativos. Es importante tener en cuenta que, durante la Actividad 3, se realizó un cambio en la asignación de pines de salida debido a un problema específico, aunque el código estaba correctamente configurado.

- Nuria

Fue una experiencia interesante esta práctica, ya que me tocó trabajar por primera vez con microcontroladores. Se me hizo gratificante, ya que tengo un poco de experiencia armando circuitos, pero nunca había programado uno; me gustó el hecho de programar algo que armé.

Algo interesante que pasó es que nuestro circuito funcionaba casi como esperábamos, a excepción de un led. Verificamos que nuestro código y armado estaban bien, pero el led seguía sin apagarse cuando tenía que hacerlo. Le pedimos ayuda a una maestra y nos ayudó a resolver el problema. Nos empujó a hacer cuestionamientos sobre qué podría estar saliendo mal, y creo que eso nos dio herramientas para depurar nuestros códigos y armados mejor en el futuro. Nos percatamos de que el problema era uno de los pines, entonces cambiamos el led a una terminal que fuera igual a la de los otros leds (uno de ellos), y eso resolvió el problema.

- Mariel

## 7. Fuentes consultadas

- [1] Marmolejo, R. E. “Microcontrolador - qué es y para que sirve”, el 12 de noviembre de 2017. <https://hetpro-store.com/TUTORIALES/microcontrolador/> (Consultado el 24 de agosto de 2023).
- [2] Tailor, H. “Comparación de la velocidad de ESP32 vs Arduino”, el 22 de marzo de 2022. <https://www.makerguides.com/es/esp32-vs-arduino-speed-comparison/> (Consultado el 24 de agosto de 2023).
- [3] Raydiy. Arduino vs ESP32 - Which one is best? Feature Comparison & First Steps in Arduino IDE. [Video en línea]. Disponible en:  
<https://youtube.com/watch?v=-uef6wPrBBU&si=IqlvqHWAwaBQQ1qd>
- [4] Carrillo, M. ESP32 vs Arduino? [Video en línea]. Disponible en:  
[https://youtu.be/7hnq113eH0M?si=4NHyvGCWkYfBMi\\_w](https://youtu.be/7hnq113eH0M?si=4NHyvGCWkYfBMi_w)
- [5] Cruz, T. J. D. “ESP32 vs. Arduino Uno: Which One Should You Get?”, el 17 de agosto de 2022. <https://www.iottechtrends.com/esp32-vs-arduino-uno/> (Consultado el 24 de agosto de 2023).
- [6] L. del Valle Hernández, “Resistencia pull up y pull down con Arduino, para qué sirven”, el 8 de agosto de 2015.  
<https://programarfácil.com/blog/arduino-blog/resistencia-pull-up-y-pull-down/> (Consultado el 24 de agosto de 2023).