

**Instituto Politécnico Nacional**

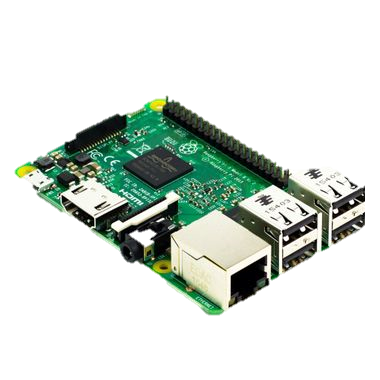
**Escuela Superior de Ingeniería Mecánica y Eléctrica**

**Cuaderno de prácticas**

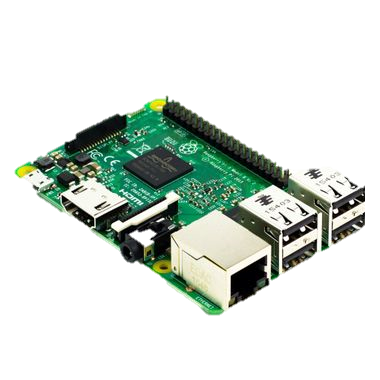
**del curso de**

**Sistemas Embebidos I**

**Profesor:   
Roberto Galicia Galicia**



Patrón de fondo

Descripción generada automáticamente

Patrón de fondo

Descripción generada automáticamente

**CDMX Enero del 2025**

Prólogo

El avance vertiginoso de la tecnología ha obligado a modificar y, en algunos casos, cambiar drásticamente la forma de enseñar cursos de tecnología, como en el caso de los microprocesadores, microcontroladores y, actualmente, los sistemas embebidos. El uso de las TIC’s es imprescindible para impartir estos cursos. Se ha hecho común que el estudiante utilice hoy la inteligencia artificial para complementar sus conocimientos y resolver dudas o ejercicios planteados por el profesor, o como ejercicios complementarios de los temas cubiertos. Con mayor razón, la práctica docente nos obliga a hacer uso de la inteligencia artificial de manera diestra y desarrollar herramientas propias para mejorar la enseñanza. La retroalimentación es importante, ya que permite conocer el grado de aprendizaje del alumno y poder formular ejercicios que mejoren o corrijan la enseñanza.

Es necesario incorporar el uso de los paradigmas de programación más utilizados, cubrir el estado del arte de los dispositivos y herramientas de desarrollo, lenguajes de programación, tarjetas de sensores y actuadores. Este cuaderno de prácticas pretende contar con los elementos y herramientas más recientes que permitan formar alumnos con las mayores capacidades para enfrentar el campo laboral.

Índice

[Practica 1 Blink ~ HolaMundo 4](#_Toc179775686)

# Practica 1 Blink ~ HolaMundo

**Objetivo**

Desarrollar y ejecutar un programa en C y Python que integre las funcionalidades de control de hardware (Blink) y la interacción con el sistema mediante la impresión en consola ("Hola Mundo") en microcontroladores Raspberry Pi Pico y ESP32. Adicionalmente, se implementará una barra de LEDs para visualizar el parpadeo, lo cual permitirá a los participantes observar de manera tangible el funcionamiento del código, facilitando la comprensión de conceptos fundamentales en programación y control de hardware en un contexto práctico y educativo.

**Descripción de la práctica**

En esta práctica, los estudiantes armarán una barra de LEDs y desarrollarán un programa que combina dos funcionalidades esenciales en la programación de microcontroladores: el control de un LED (Blink) y la interacción con el sistema a través de mensajes en la consola ("Hola Mundo"). Esta combinación se implementará en dos lenguajes de programación, C y Python, y se llevará a cabo en dos plataformas populares de microcontroladores: Raspberry Pi Pico y ESP32.

La práctica comenzará con la configuración de los pines GPIO de cada microcontrolador para controlar los LEDs de la barra. Los LEDs parpadearán a intervalos regulares de tiempo, y durante cada cambio de estado, se imprimirá el mensaje "Hola Mundo" en la consola. Esto permitirá a los estudiantes observar tanto el comportamiento físico del hardware (parpadeo de los LEDs) como la interacción del software (impresión del mensaje en la consola).

El propósito es que los estudiantes aprendan a manejar las interfaces de hardware de los microcontroladores, familiarizándose con la gestión de pines digitales (GPIO) y, al mismo tiempo, integren la lógica de software necesaria para generar salidas en consola. Esta práctica también les permitirá comparar cómo se realizan estas tareas en C, un lenguaje de bajo nivel, frente a Python, un lenguaje de alto nivel, además de entender las particularidades de trabajar con dos microcontroladores distintos.

**Materiales**

* 1 x Raspberry Pi Pico (con puerto micro-USB y soporte de micropython o C).
* 1 x ESP32 (con puerto micro-USB y soporte de micropython o C).
* 1 x Cable USB tipo A a micro-USB (para conectar la Raspberry Pi Pico al computador).
* 1 x Cable USB tipo A a micro-USB o tipo A a USB-C (según la versión del ESP32 que utilices).
* 1 x barra de LEDS de 12 segmentos de alto brillo (Ánodo o cátodo común)
* 1 x cabezal(header) hembra
* 12 x resistencias de 1/8 de watt
* 1 x Protoboard
* 1 x Cautín
* 1 x multímetro (opcional)
* Flux
* Tercera mano (opcional)
* Soldadura
* 1 x Fuente de alimentación de 5V

**Software**

* IDE para Python:
  + Para ambas plataformas: Thonny con soporte de MicroPython o Visual Studio Code con la extensión de Python.
* IDE para C:
  + Para Raspberry Pi Pico: Raspberry Pi Pico SDK o Visual Studio Code con la extensión de C/C++.
  + Para ESP32: Espressif IDF (Espressif IoT Development Framework).

**¿Cómo calcular el valor de las resistencias?**

Para calcular la resistencia que se debe usar con una barra de LEDs, se aplica la ley de Ohm, que es:

Donde:

* es el voltaje de alimentación (el que proporciona el microcontrolador).
* es el voltaje de trabajo del LED.
* es la corriente ideal de operación que queremos para el LED.

De acuerdo con la hoja de datos de la Figura 1:

Tabla

Descripción generada automáticamente

Figura 1 Características eléctricas de barra 12 segmentos

La tensión directa máxima de led rojo es de , y su corriente de operación ideal es de . Tanto la Raspberry Pi Pico como el ESP32 suministran 3.3V a través de sus pines GPIO. Para asegurar un funcionamiento adecuado del LED y evitar daños, es necesario calcular la resistencia adecuada utilizando la ley de Ohm.

Dado que ambos dispositivos proporcionan un voltaje superior al de trabajo del LED, se aplicará la siguiente fórmula:

Para operar un LED ultrabrillante rojo con la Raspberry Pi Pico y el ESP32, se debe utilizar una resistencia de 140 ohmios para limitar la corriente a 10 mA y proteger el LED. Sin embargo, dado que 140 ohmios no es un valor estándar, se recomienda emplear una resistencia de 150 ohmios, lo que limitará la corriente a aproximadamente 8.6 mA. Este valor está dentro del rango aceptable para el LED ultrabrillante rojo, lo que proporciona una buena protección y asegura su funcionamiento adecuado.

**Barra LEDS Ánodo Común**

Diagrama

Descripción generada automáticamente

Figura 2 Diagrama barra de LEDs ánodo común

En un circuito de ánodo común, Figura 2, todos los ánodos de los LEDs están conectados a una fuente de voltaje común, mientras que los cátodos se conectan a los pines de control o a tierra. Esto permite que la corriente fluya a través de los LEDs cuando se aplica un voltaje a los cátodos. En este tipo de configuración, el voltaje positivo se aplica al conjunto de LEDs, y cada LED se activa conectando su cátodo a tierra (o a un voltaje más bajo).

**Funcionamiento:**

Para encender un LED específico, se establece el pin correspondiente a 0V (tierra), permitiendo que la corriente fluya desde el voltaje común de 3.3V a través del LED y hacia el pin que se ha llevado a tierra.

Cuando el pin se establece a un voltaje alto (3.3V), el LED se apaga, ya que no hay diferencia de potencial a través del LED.

Es importante asegurarse de que cada LED tenga la resistencia adecuada en serie para limitar la corriente a un nivel seguro (por ejemplo, utilizando resistencias de 68 ohmios, como se discutió anteriormente).

**Barra LEDS Cátodo Común**

Diagrama

Descripción generada automáticamente

Figura 3 Diagrama barra de LEDs cátodo común

En un circuito de cátodo común, Figura 3, todos los cátodos de los LEDs están conectados a un punto de referencia común, generalmente tierra (GND), mientras que los ánodos de los LEDs se conectan a los pines de control. En esta configuración, el LED se enciende cuando se aplica un voltaje positivo a su ánodo, permitiendo que la corriente fluya desde el ánodo al cátodo común.

**Funcionamiento:**

Para encender un LED, se aplica un voltaje positivo al pin de control correspondiente (por ejemplo, del pin 1 al pin 12). Cuando el pin de control se lleva a un estado alto, se genera una diferencia de potencial entre el ánodo y el cátodo, permitiendo que el LED se ilumine.

Si no hay voltaje aplicado al ánodo o el pin de control está en un estado bajo, el LED permanece apagado, ya que no hay corriente que fluya a través del LED.

Se debe agregar una resistencia limitadora de corriente en serie con cada LED para evitar daños por sobre corriente.

**Desarrollo Experimental**

***Montado de la barra de leds***

Para determinar la polaridad de la barra de LEDs, ya sea de cátodo común o ánodo común, se procede de la siguiente manera. Primero, se coloca la barra de LEDs en un protoboard y se utiliza una resistencia limitadora de 560 ohmios en serie con la barra para proteger los LEDs. A continuación, se conecta una fuente de alimentación de 5V, Figura 4.

Para probar si la barra es de cátodo común, se conecta el terminal negativo (GND) de la fuente de alimentación a una de las terminales del protoboard y se conecta la resistencia entre esta terminal y el extremo correspondiente de la barra de LEDs. Luego, se conecta el terminal positivo de la fuente al otro extremo de la barra (los ánodos de los LEDs). Si los LEDs encienden, se confirma que la barra es de cátodo común.

En caso de que los LEDs no enciendan, se debe invertir la polaridad de la conexión para verificar si la barra es de ánodo común. En este caso, se conecta el terminal positivo de la fuente de alimentación a la resistencia y esta a la barra de LEDs. Después, se conecta el terminal negativo de la fuente al otro extremo de la barra (los cátodos). Si los LEDs encienden con esta configuración, se concluye que la barra es de ánodo común.

Imagen que contiene escurridor de platos, cama, cocina, alimentos

Descripción generada automáticamente

Figura 4 Barra 12 segmentos

Una vez obtenida la polaridad de la barra de LEDs, se procede a colocar las 12 resistencias, asegurando que queden fijas y no se muevan. A continuación, se suelda una de las terminales de cada resistencia a un cable común, preferiblemente de color negro, que servirá como conexión a tierra (en caso de cátodo común) o como línea de alimentación positiva (en caso de ánodo común). Este cable común facilitará la correcta distribución de la corriente a través de las resistencias, asegurando un funcionamiento adecuado y estable de los LEDs, como se muestra en la Figura 5.

Las terminales de las resistencias que no se han soldado se van a soldar al cátodo de la barra de LEDs en caso de que la barra sea de cátodo común, o al ánodo en caso de que la barra sea de ánodo común. Antes de soldarlas, es importante medir cuidadosamente las longitudes de los cables para garantizar que el montaje sea estético y ordenado. Las resistencias deben alinearse correctamente, asegurando que los cables tengan una longitud uniforme y estén bien organizados.

Imagen que contiene alimentos, cuchillo

Descripción generada automáticamente

Figura 5 Resistencias soldadas

Las terminales restantes de la barra de LEDs, que corresponden a los ánodos en el caso de una barra de cátodo común o a los cátodos en el caso de una barra de ánodo común, se soldarán a un header (conector). Esta disposición facilitará una conexión ordenada y eficiente con otros componentes o pines de control, como los que se encuentran en un microcontrolador. El resultado final de estas conexiones se puede apreciar en la Figura 6.

Imagen que contiene firmar, pastel

Descripción generada automáticamente

Figura 6 Barra con cabezal soldado

Una vez realizadas las conexiones, se procede a realizar la prueba de funcionamiento, como se muestra en la Figura 7.

Imagen que contiene foto, mujer, tabla, diferente

Descripción generada automáticamente

Figura 7 Verificación de funcionamiento de la barra

**Posibles fallas:**

* Inversión de polaridad: Si se conectan incorrectamente los terminales de la barra de LEDs, se puede invertir la polaridad, lo que impediría que los LEDs se enciendan o incluso podría dañarlos.
* Resistencias incorrectas: Utilizar resistencias de un valor incorrecto puede llevar a que la corriente que fluye a través de los LEDs sea demasiado alta o baja. Esto podría resultar en un brillo excesivo, que puede quemar el LED, o en un brillo insuficiente, que puede hacer que no se encienda.
* Conexiones sueltas: Cualquier conexión suelta entre la barra de LEDs, las resistencias, o el header puede causar que los LEDs no funcionen adecuadamente. Las vibraciones o movimientos podrían aflojar estas conexiones.
* LEDs defectuosos: Algunos LEDs pueden estar defectuosos desde el inicio, lo que significa que no se encenderán independientemente de que el resto del circuito esté bien configurado.
* Cortocircuito: Si los cables o las conexiones se cruzan accidentalmente, esto podría provocar un cortocircuito que no solo impediría que los LEDs se enciendan, sino que también podría dañar el microcontrolador o la fuente de alimentación.
* Fuentes de alimentación inadecuadas: Si la fuente de alimentación no proporciona el voltaje adecuado o es incapaz de suministrar suficiente corriente, esto puede afectar el funcionamiento de los LEDs.
* Mala soldadura: Si las soldaduras no son de buena calidad (por ejemplo, si tienen franja de óxido o están mal formadas), pueden crear resistencia adicional o fallas de conexión, lo que puede interferir con el funcionamiento del circuito.

*Automatización de la codificación mediante inteligencia artificial.*

Para la realización de la práctica, se plantea utilizar diversas inteligencias artificiales, como ChatGPT, Gemini, Copilot y Blackbox AI, con el objetivo de obtener ejemplos de código que contengan las funciones de parpadeo (Blink) y la impresión de mensajes en consola ("Hola Mundo") para la Raspberry Pi Pico y ESP32, en los lenguajes de programación Python y C. A través de estas herramientas, se busca facilitar el aprendizaje y la implementación de conceptos fundamentales en la programación de microcontroladores.

Se utiliza el siguiente prompt: *"Give me a code that contains the Blink and Hello World function for the Raspberry Pico in Python and C language."* A partir de esta consulta, se obtienen diferentes respuestas que proporcionan ejemplos de código para implementar las funciones de parpadeo y la impresión de mensajes en consola.

En este caso, se realiza el análisis de los códigos generados por distintas IAs específicamente para la Raspberry Pi Pico. Una de las actividades que el alumno debe llevar a cabo es solicitar a las diferentes IAs el código necesario para la placa ESP32, seguido del análisis de los códigos proporcionados por cada IA.

BLACKBOX AI

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

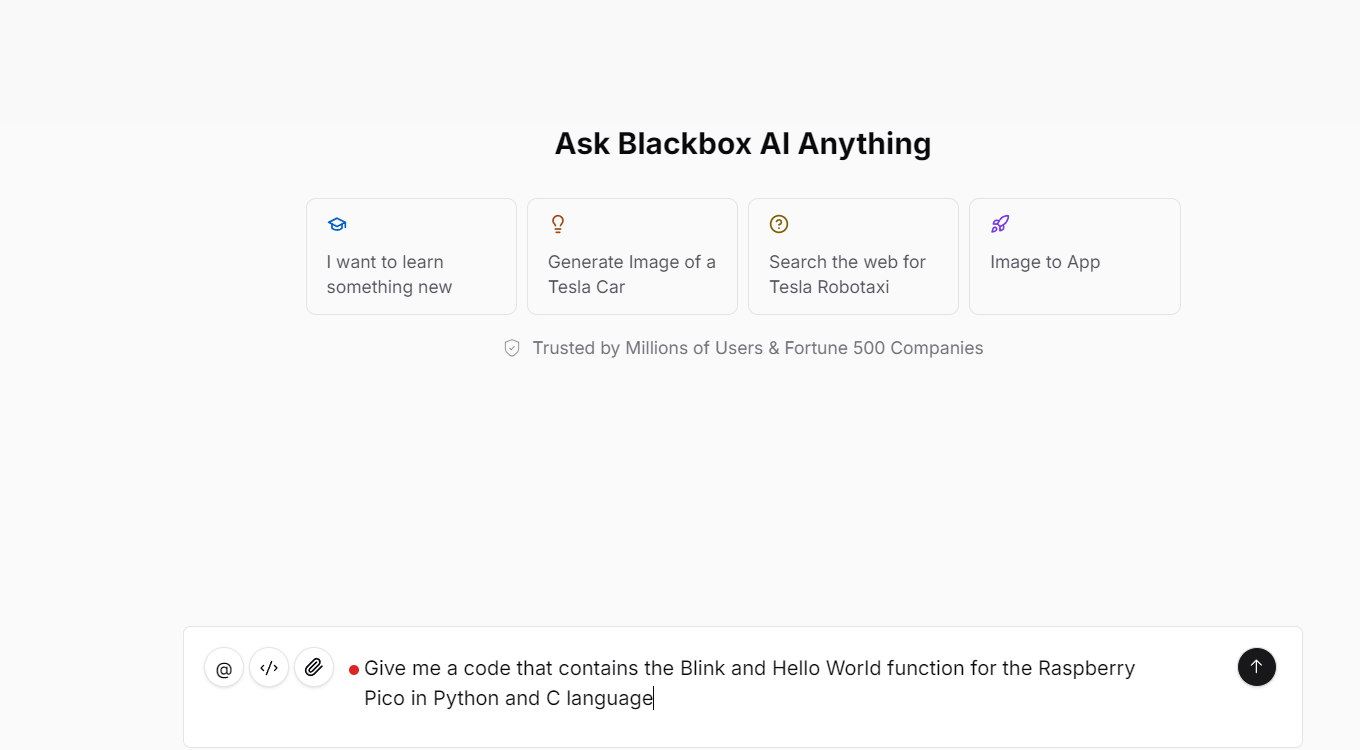
Descripción generada automáticamente**

Figura 8 Codigo en C Prompt para BlackBox AI Raspberry Pico "Blink Hola Mundo"

Figura 9 Codigo en Python Prompt para BlackBox AI Raspberry Pico "Blink Hola Mundo"

Figura 10 Prompt para BlackBox AI Raspberry Pico "Blink Hola Mundo"

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Figura 11 UML codigo en Python Prompt para BlackBox AI Raspberry Pico "Blink Hola Mundo"

Figura 12 UML codigo en C Prompt para BlackBox AI Raspberry Pico "Blink Hola Mundo"

ChatGPT

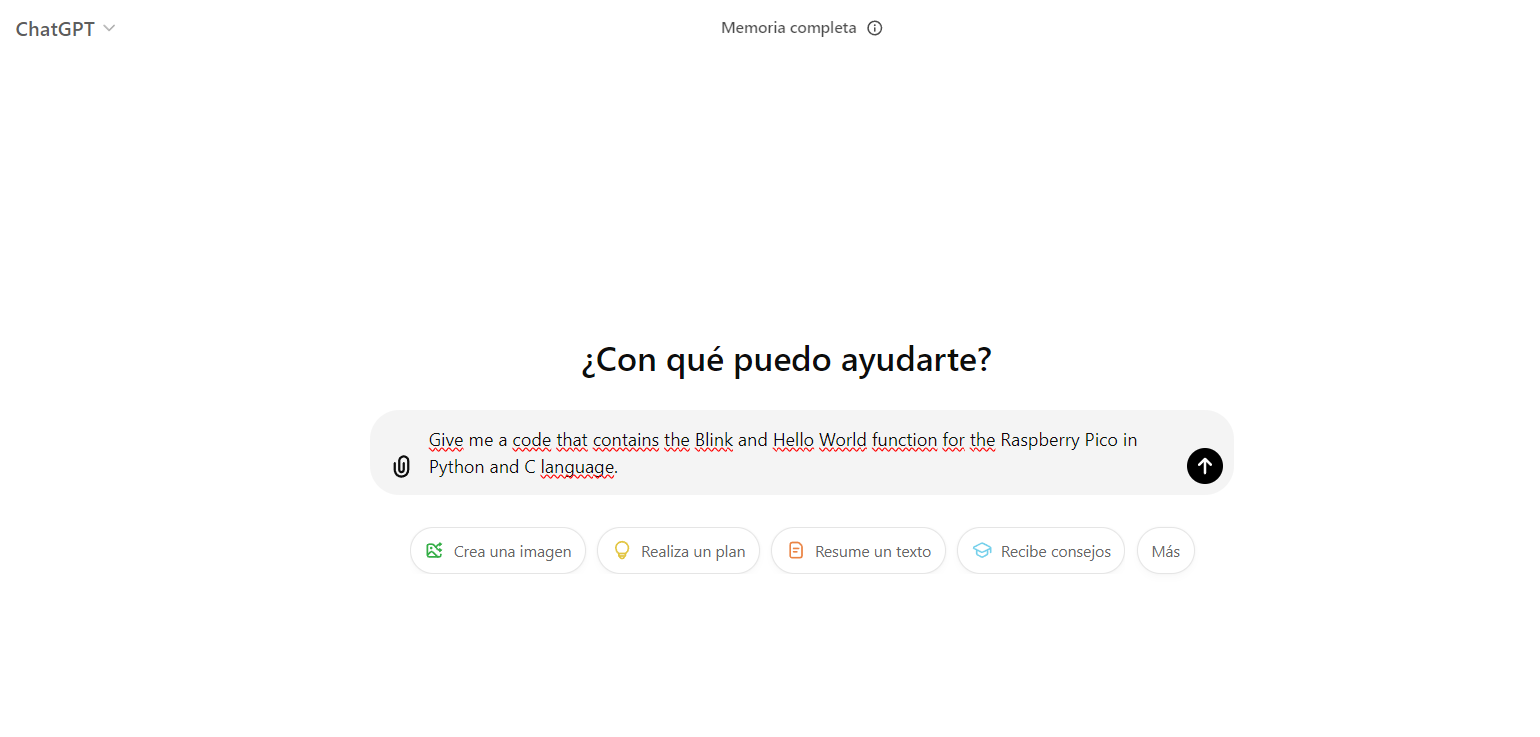


Figura 13 Prompt para ChatGPT Raspberry Pico "Blink Hola Mundo"

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 14 Codigo en Python Prompt para ChatGPT Raspberry Pico "Blink Hola Mundo"

Figura 15 Codigo en C Prompt para ChatGPT Raspberry Pico "Blink Hola Mundo"

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Figura 16 UML Codigo en Python Prompt para ChatGPT Raspberry Pico "Blink Hola Mundo"

Figura 17 UML Codigo en C Prompt para ChatGPT Raspberry Pico "Blink Hola Mundo"

GEMINI

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

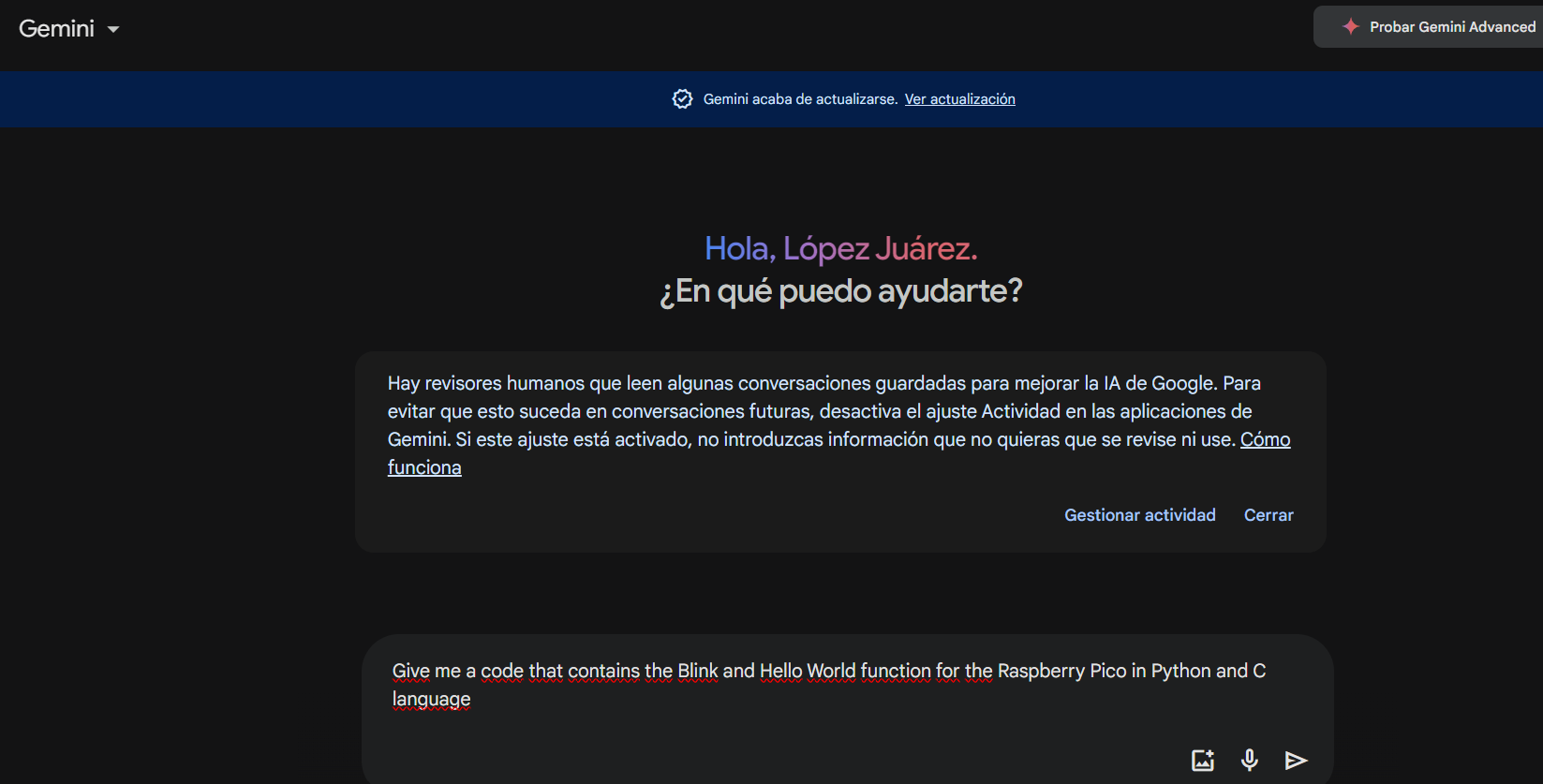
Descripción generada automáticamente

Figura 18 Codigo en Python Prompt para GEMINI Raspberry Pico "Blink Hola Mundo"

Figura 19 Codigo en C Prompt para GEMINI Raspberry Pico "Blink Hola Mundo"

Figura 20 Prompt para GEMINI Raspberry Pico "Blink Hola Mundo"

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Figura 21 UML Codigo en Python Prompt para GEMINI Raspberry Pico "Blink Hola Mundo"

Figura 22 UML codigo en C Prompt para GEMINI Raspberry Pico "Blink Hola Mundo"

COPILOT

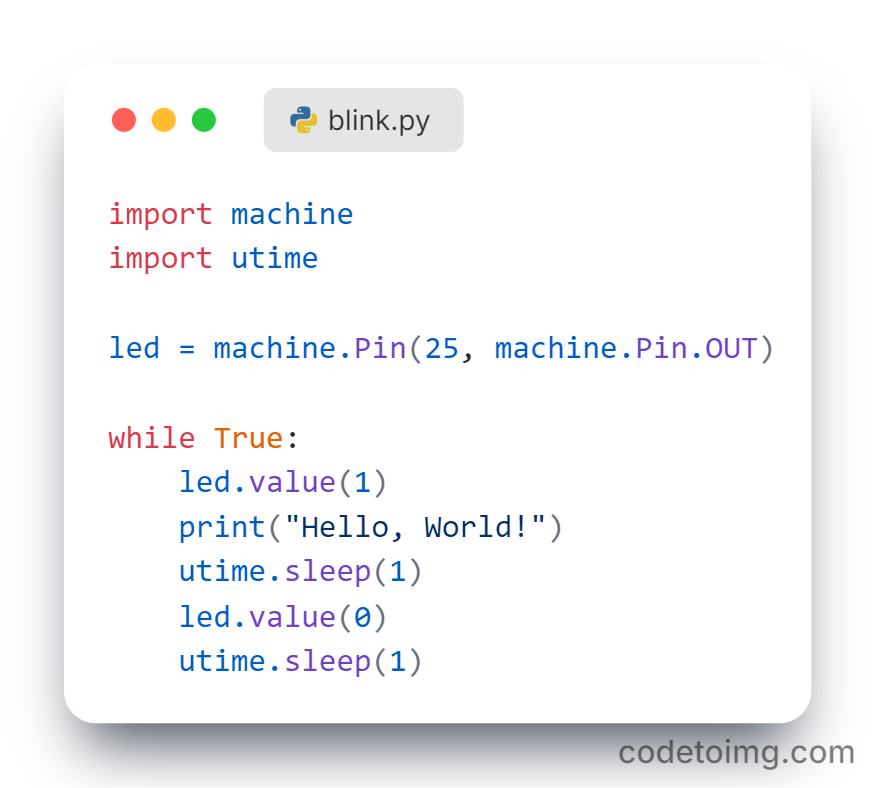
**** 

Figura 23 Codigo en Python Prompt para COPILOT Raspberry Pico "Blink Hola Mundo"

Figura 24 Prompt para COPILOT Raspberry Pico "Blink Hola Mundo"

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 25 Codigo en C Prompt para COPILOT Raspberry Pico "Blink Hola Mundo"

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Figura 26 UML Codigo en C Prompt para COPILOT Raspberry Pico "Blink Hola Mundo"

Figura 27 UML Codigo en Python Prompt para COPILOT Raspberry Pico "Blink Hola Mundo"

**Comparativa**

**En C**

La tabla comparativa de la Figura 28 analiza cómo distintas IAs generan código en lenguaje C utilizando el SDK de Pico para parpadear un LED y mostrar el mensaje "Hola, Mundo!" en la consola. Los aspectos evaluados incluyen la inicialización del pin LED, la impresión del mensaje, el uso de bucles infinitos, el intervalo de parpadeo y la claridad del código. Además, se destaca la experiencia del usuario según la implementación de cada IA.

**Tabla

Descripción generada automáticamente**

Figura 28 Comparativa de codigo en C de IAs

**En Python**

La tabla comparativa de la Figura 29 analiza cómo diferentes IAs generan código en Python utilizando MicroPython para parpadear un LED y mostrar "Hola, Mundo!" en la consola. Se evalúan aspectos clave como la correcta inicialización del pin LED, la ubicación y frecuencia de la impresión del mensaje, el uso de bucles infinitos, los intervalos de parpadeo y la claridad del código. También se considera la experiencia del usuario según la repetición oportuna del mensaje y la sincronización del parpadeo del LED.

****

Figura 29 Comparativa de codigo en Python de IAs

**Diagramas de apoyo**

Diagrama, Esquemático

Descripción generada automáticamente

Figura 30 Diagrama eléctrico de la barra de leds

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Figura 31 Diseño de la barra de leds

Calendario

Descripción generada automáticamente

Figura 32 Conexión de leds con resistencia limitadora

El circuito presentado de la Figura 33 consiste en una placa de desarrollo ESP32 conectada a doce (12) diodos LED de color rojo. Cada LED se encuentra en serie con una resistencia, lo cual es un elemento fundamental para limitar el flujo de corriente y evitar daños en los componentes. La placa ESP32 actúa como el cerebro de este sistema, siendo responsable de controlar el encendido y apagado de los LEDs de manera individual o en conjunto.

A través de su programación, el ESP32 puede enviar señales eléctricas a los LEDs a través de sus pines de salida. Estas señales, combinadas con el valor de las resistencias, determinan el brillo y el estado de encendido de cada LED. Es posible crear una amplia variedad de patrones luminosos, desde un simple encendido y apagado hasta secuencias complejas y animaciones.

Diagrama, Esquemático

Descripción generada automáticamente

Figura 33 Conexión de la barra de leds a una ESP32

El circuito de la Figura 34 está diseñado para controlar un conjunto de 12 LEDs rojos utilizando una placa de desarrollo Raspberry Pi Pico W. Cada LED está conectado en serie con una resistencia para limitar la corriente eléctrica y proteger los LEDs.

Diagrama, Esquemático

Descripción generada automáticamente

Figura 34 Conexión de la barra de leds a una raspberry pi pico

ACTIVIDADES EXPERIMENTALES

1.- Prueba y ejecución de los programas propuestos para la Raspberry Pi Pico.

2.- Basado en el ejemplo de la Raspberry realiza un programa para la esp32 en C y en Python que funcione de manera similar a lo propuesto.

3.- Modificar el programa anterior para que el LED parpadee solo si el usuario ingresa un comando específico ("encender" o "apagar") en la consola.

4.- Crear una animación tipo "marquesina" donde los LEDs conectados a la ESP32 o Raspberry Pi Pico se enciendan uno tras otro en secuencia y luego vuelvan a comenzar.

5.- Simular un vúmetro donde los LEDs representen el nivel de intensidad, encendiendo más LEDs a medida que aumenta la entrada o un valor simulado.

6.- Implementar un patrón de encendido y apagado de LEDs en forma de señalización alterna, como luces de advertencia o emergencia.

DESARROLLO DE EJEMPLOS EN BASE A LA PRACTICA

En esta actividad, los estudiantes desarrollarán programas en C y Python que combinen el control de LEDs (Blink) y la impresión de mensajes de texto (Hola Mundo). A través de estas propuestas, se busca que los estudiantes comprendan cómo interactuar con hardware (como la ESP32) y manejar entradas y salidas de datos de manera eficiente, utilizando técnicas de formateo de cadenas en Python y control de hardware en C.

***Instrucciones Generales:***

Los estudiantes deben crear programas que no solo controlen los LEDs conectados a la ESP32 (u otro microcontrolador), sino que también integren mensajes informativos o interactivos en consola, tanto en C como en Python.

* Formato de Mensajes y Opciones por Teclado.
* Opciones de Patrones de Parpadeo.
* Opciones de Simulaciones con PWM.
* Interacción con Entradas de Usuario:

Cuestionario

1. ¿Qué significa "Blink" en el contexto de la programación de microcontroladores?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. ¿Cómo se usa la función "printf" en C y las f-strings en Python para imprimir "Hola Mundo"?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. ¿Qué función cumple la declaración pinMode (C) o Pin() (Python) en los programas de Blink?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. En la Raspberry Pi Pico, ¿cómo configuras un pin GPIO en C para que funcione como salida y controle un LED?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. ¿Cómo podrías modificar el programa "Hola Mundo" en Python para que el nombre del usuario se ingrese desde el teclado y se imprima como "Hola, [Nombre]" en la consola de la Raspberry Pi Pico?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Si tienes un LED con una caída de voltaje de 2V y deseas conectarlo a un GPIO de 3.3V en la Raspberry Pi Pico, ¿cuál debe ser el valor de la resistencia limitadora de corriente si deseas que el LED funcione con 20 mA?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. ¿Cómo puedes modificar el programa de Blink para que el LED parpadee en un patrón específico, como 2 segundos encendido y 1 segundo apagado, en Raspberry Pi Pico?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Propuesta Creativa: Diseña un programa en C o Python para la ESP32 o Raspberry Pi Pico que controle tres LEDs de forma que simulen el comportamiento de un semáforo. El programa también debe imprimir en consola los mensajes correspondientes, como "Semáforo en verde", "Semáforo en amarillo", "Semáforo en rojo". ¿Cómo lo implementarías?

Proporcione un diagrama de flujo o pseudocódigo que explique el diseño del programa.

Enlaces

* Documentacion Doxygen
  + <https://willowy-melomakarona-913792.netlify.app/index.html>
* GITHUB
  + Datasheet
    - <https://github.com/SEmbebidosICEESIMEZ/datasheets.git>
  + Practica 1
    - <https://github.com/SEmbebidosICEESIMEZ/Practica1.git>
* Enlaces TEAMS
  + Curso Python
    - [CURSO\_PYTHON\_COMPU | General | Microsoft Teams](https://teams.microsoft.com/l/team/19%3Az_9yYGvMK--UaOdoLEUNdKtHxeJ2Q7TGkulktjJlULI1%40thread.tacv2/conversations?groupId=12ed7bec-4170-419f-b629-22dd0705bbe9&tenantId=f94bf4d9-8097-4794-adf6-a5466ca28563)
  + MICROS\_6CM4\_24B
    - [MICROS\_6CM4\_24B | General | Microsoft Teams](https://teams.microsoft.com/l/team/19%3AvDnArMUhwkIi-6GTwrITNgpe5VwFY9VwczCt0merW641%40thread.tacv2/conversations?groupId=46f41c71-bcc9-4388-96df-b56a3a8567fc&tenantId=f94bf4d9-8097-4794-adf6-a5466ca28563)