**UNIVERSIDAD AUTÓNOMA DE QUERÉTARO**

# INGENIERÍA BIOMÉDICA

# PRACTICAS DE LABORATORIO

**Periodo: agosto-diciembre 2022**

**Docente: Ing. José de Jesús Santana Ramírez**

**Ayudante: Ricardo Carrillo Guzmán**

**Integrantes del equipo:**

**Cruz Márquez Héctor**

**Flores Moreno Ángel**

# Práctica 1

## Experimento 1

Para este experimento, se utiliza uno de los LED´s integrados en Tiva LaunchPad. El LED estará controlado por uno de los pines GPIO del microcontrolador Tiva. Para lograr la tasa de parpadeo requerida, el pin GPIO que controla el LED deberá alternarse entre los estados lógico alto y lógico bajo con el retraso requerido de 100 ms. El diagrama de bloques básico para la configuración del experimento se muestra en la Fig. 1. El Tiva LaunchPad tiene un LED RGB incorporado, controlado por los pines GPIO PF1, PF2 y PF3.

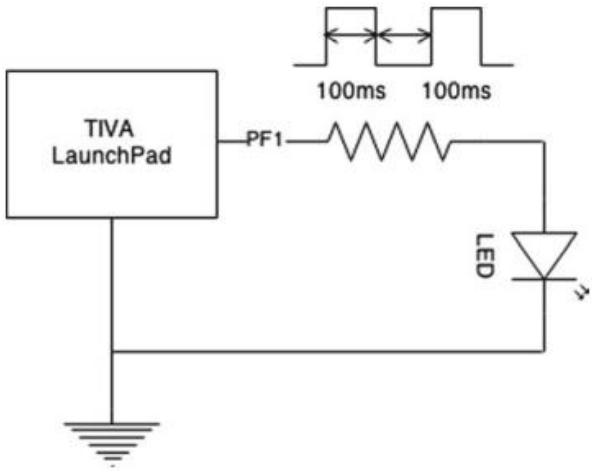



Fig 1. Diagrama de Conexión.

Primera parte:

Compilando el programa del Blinky con el make se obtuvo el parpadeo con la tiva. En el puerto F en el pin 3.

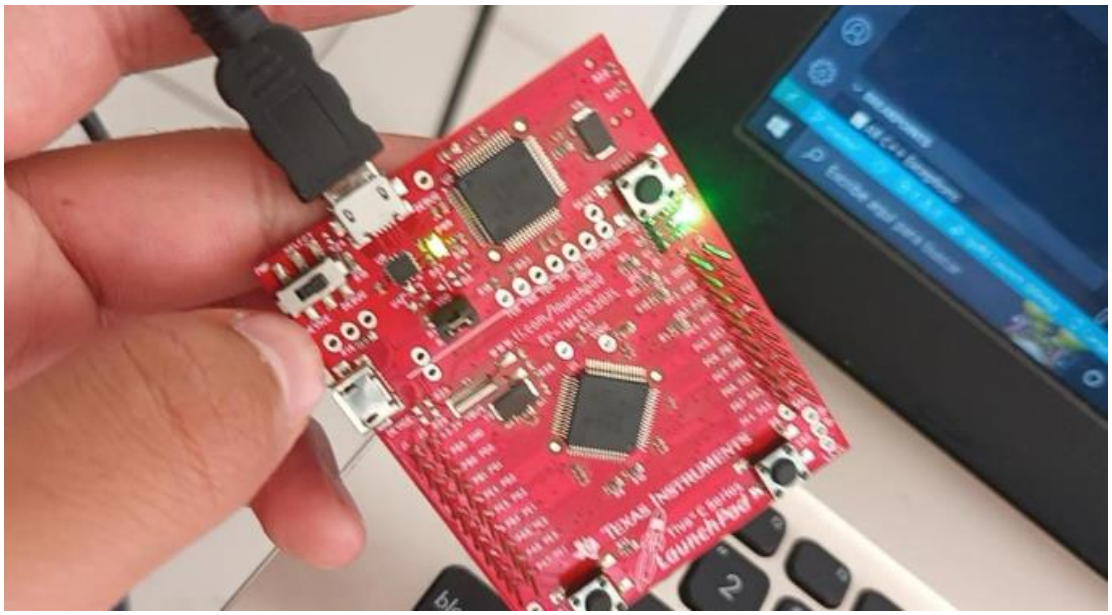



*Ilustración 2 Se muestra cuando el Led esta encendido*

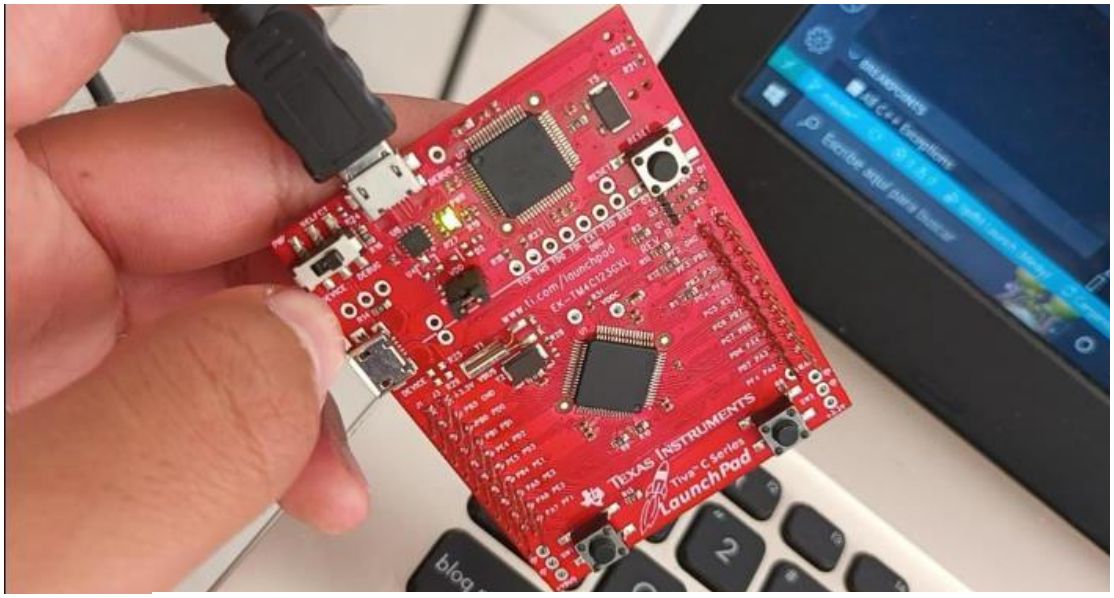



*Ilustración 1 Se muestra cuando el Led no está encendido.*

Segunda parte:
En este caso se hace una que el código de blinky sea una sola cuenta para ver en el osciloscopio en cuanto dura un periodo. Teniendo en cuenta esto.
Parte principal del código en Blinky.c

```
//*****************************************************************************
int
main(void)
{
    volatile uint32_t ui32Loop;
    volatile uint32_t valor;
    //
    // Enable the GPIO port that is used for the on-board LED.
    //
    SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;

    //
    // Do a dummy read to insert a few cycles after enabling the peripheral.
    //
    ui32Loop = SYSCTL_RCGC2_R;

    //
    // Enable the GPIO pin for the LED (PF3).  Set the direction as output, and
    // enable the GPIO pin for digital function.
    //
    GPIO_PORTF_DIR_R = 0x08;
    GPIO_PORTF_DEN_R = 0x08;

    //
    // Loop forever.
    //
    valor= 1;
    while (1)
    {
        //
        // Turn on the LED.
        //
        GPIO_PORTF_DATA_R |=0x08;// correponde F3

        //
        // Delay for a bit.
        //
        For (ui32Loop = 0; ui32Loop < valor; ui32Loop++)
        {
        }
```

```
        //
        // Turn off the LED.
        //
        GPIO_PORTF_DATA_R &= ~(0x08);

        //
        // Delay for a bit.
        //
        for (ui32Loop = 0; ui32Loop < valor; ui32Loop++)
        {
        }
    }
}
```

Al saber que es para una cuenta y que la variable *valor* equivale a 1, y al observarlo en el osciloscopio se observa que tiene un periodo de 100 µS, se hace una regla de tres para establecer el valor de la variable *valor*, que equivaldrá a 1,000,000.

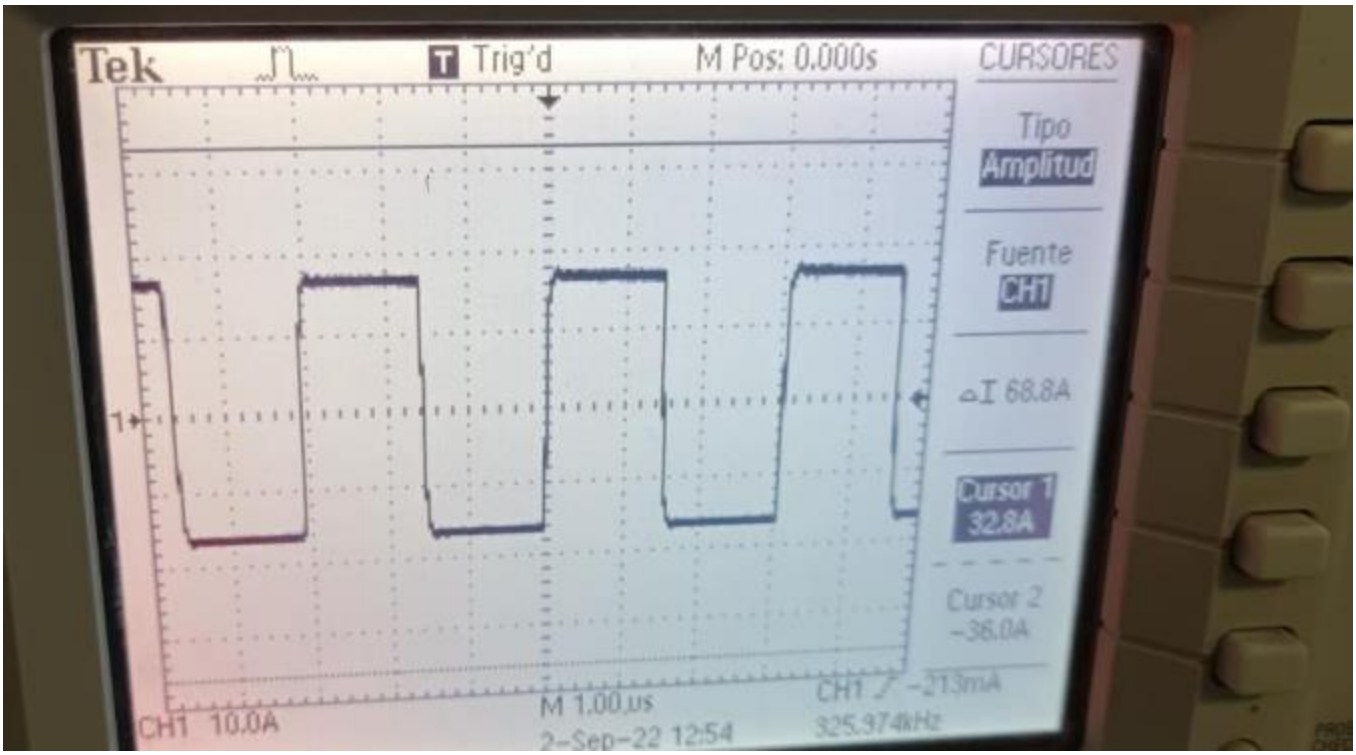



*Ilustración 3 Se muestra la medición en el osciloscopio*

# Experimento 2

Para este experimento, solo se requiere un LED, el LED rojo está controlado por PF1. El fragmento esquemático del Tiva LaunchPad (Fig. 2) muestra la configuración alta activa del LED RGB de ánodo común integrado que utiliza transistores NPN. Tal configuración es necesaria porque el microcontrolador **Tiva solo es capaz de suministrar hasta 18 mA** de corriente como máximo en dos pines ubicados en un lado físico del paquete del dispositivo. Esta limitación de corriente máxima se supera mediante el uso de la configuración de transistor NPN. En esta configuración, la lógica alta encenderá el LED y la lógica baja lo apagará.

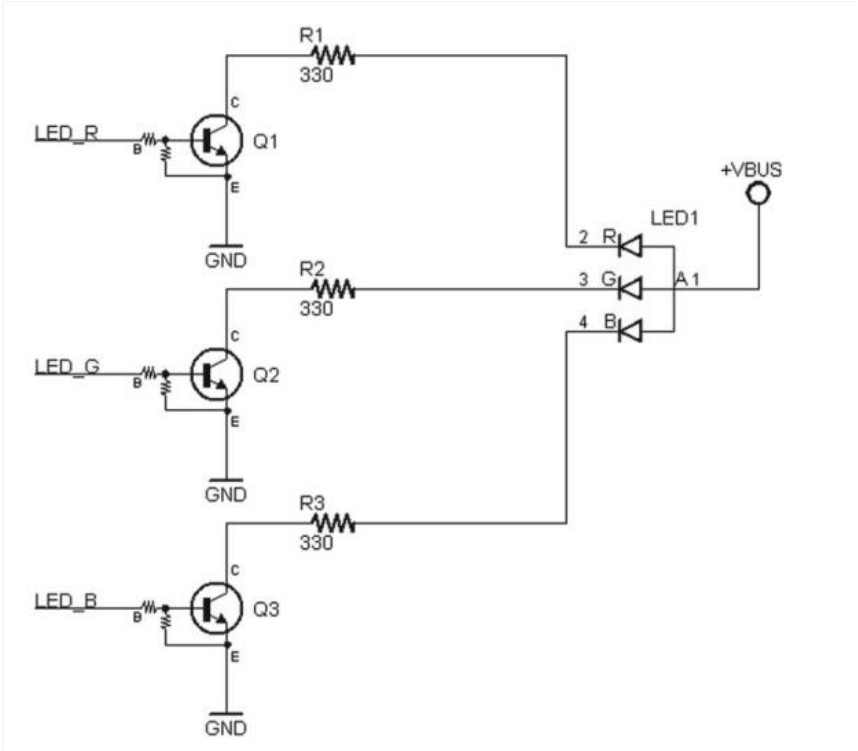
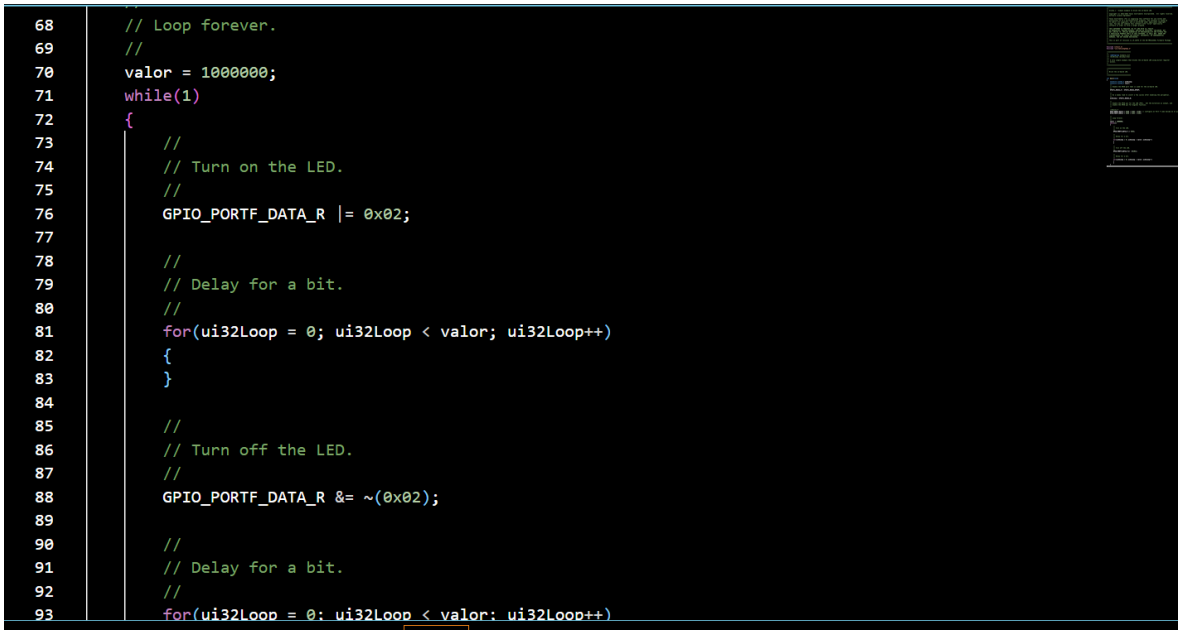



Fig 2. Diagrama de conexión transistor led RGB

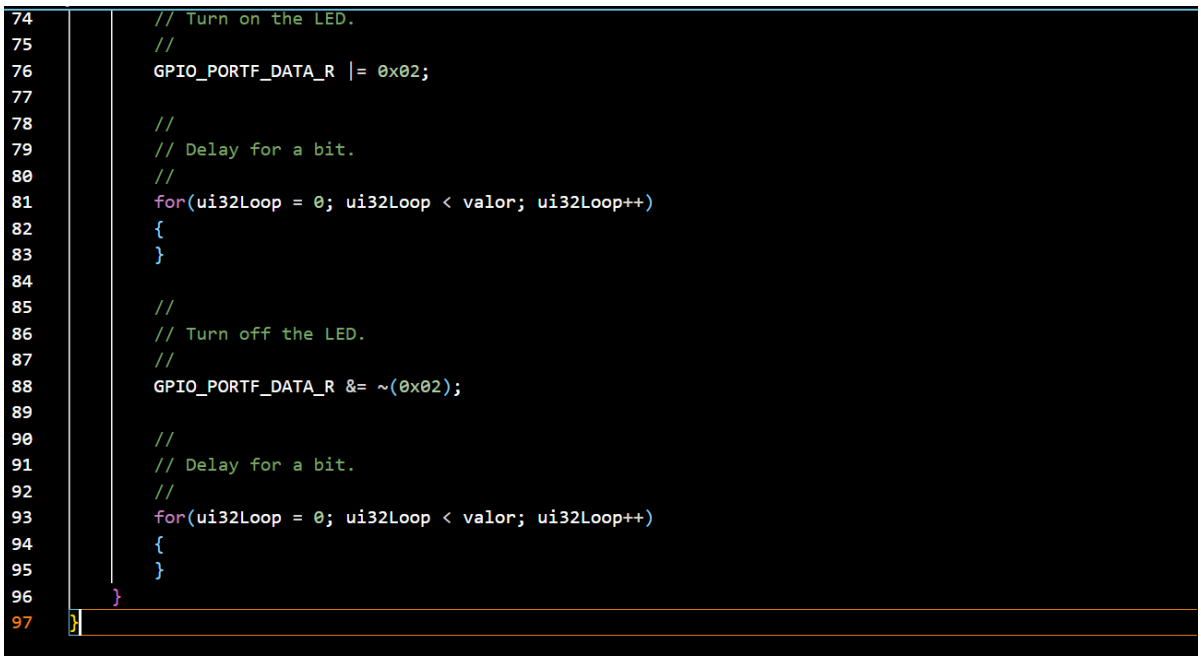
**Objetivo(s):** Parpadea un LED en Tiva LaunchPad con un retraso de 100 ms.

Primeramente para el experimento numero 2 identificamos en el código que puertos GPIO vamos a ocupar para la interaccion de pines de nuestra triva con nuestro protoboard y la parte fisica del experimento.

En esta imagen se muestra en el código la variable “valor”, la cual sirve como intervalo para prender y apagar nuestros leds en cierto tiempo, de acuerdo al periodo que la tiva manda a través del puerto F en el pin 1. El valor que ocupamos fue “1000000”.



```
68      // Loop forever.
69      //
70      valor = 1000000;
71      while(1)
72      {
73          //
74          // Turn on the LED.
75          //
76          GPIO_PORTF_DATA_R |= 0x02;
77
78          //
79          // Delay for a bit.
80          //
81          for(ui32Loop = 0; ui32Loop < valor; ui32Loop++)
82          {
83          }
84
85          //
86          // Turn off the LED.
87          //
88          GPIO_PORTF_DATA_R &= ~(0x02);
89
90          //
91          // Delay for a bit.
92          //
93          for(ui32Loop = 0; ui32Loop < valor; ui32Loop++)
```



```
74          // Turn on the LED.
75          //
76          GPIO_PORTF_DATA_R |= 0x02;
77
78          //
79          // Delay for a bit.
80          //
81          for(ui32Loop = 0; ui32Loop < valor; ui32Loop++)
82          {
83          }
84
85          //
86          // Turn off the LED.
87          //
88          GPIO_PORTF_DATA_R &= ~(0x02);
89
90          //
91          // Delay for a bit.
92          //
93          for(ui32Loop = 0; ui32Loop < valor; ui32Loop++)
94          {
95          }
96      }
97  }
```

En esta imagen mostramos la conexión fisica que realizamos para la elaboracion del experimento, en la cual usamos dos leds un transistor y dos resistencias.

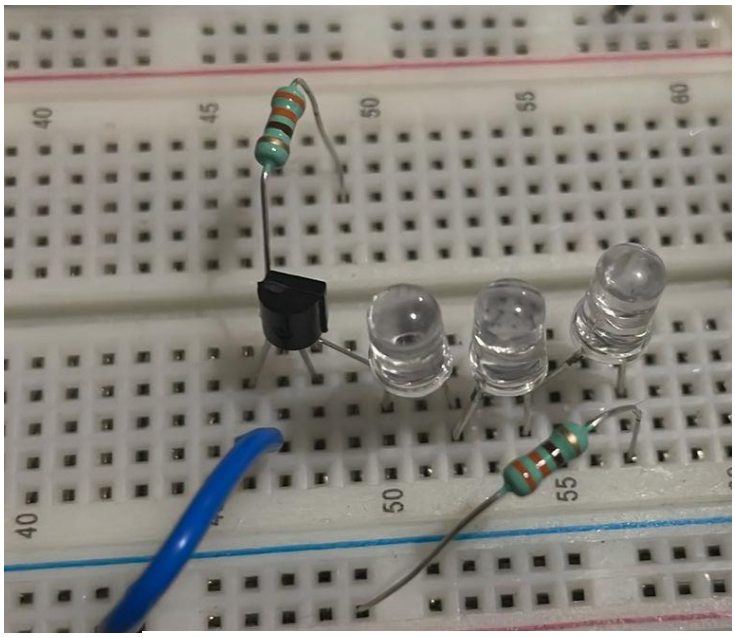



*Ilustración 4 Se muestra la conexión física*

En esta imagen se muestra como nuestrs leds estan energizados, fueron 3 leds en serie junto con el transistor y la comunicación de la tiva con el protoboard, en el cual conectamos el pin F1 de la tiva a la resistencia que salia de nuestro transistor para poder prender nuestros leds y asi establecer la comunicación requerida para que el experimento funcione de manera requerida.

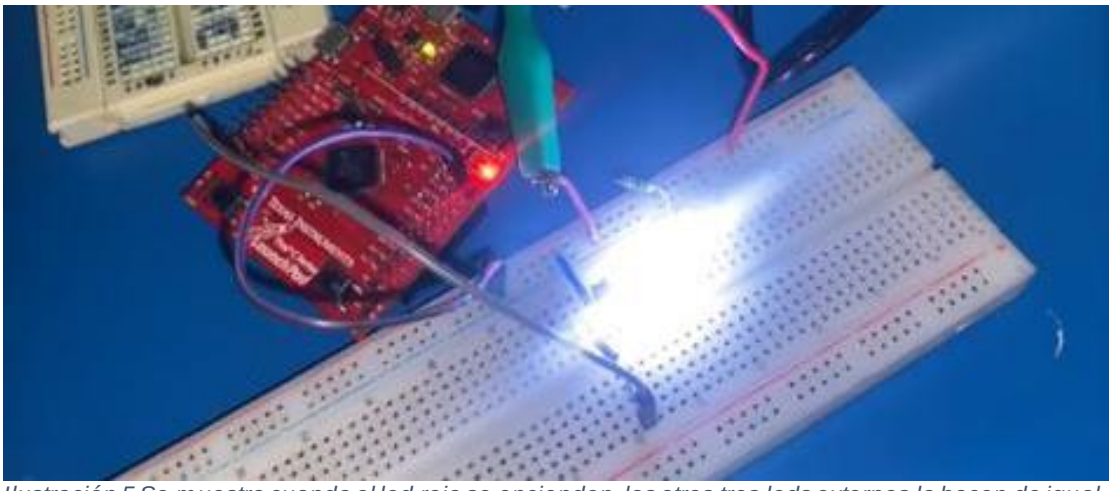


*Ilustración 5 Se muestra cuando el led rojo se encienden, los otros tres leds externos lo hacen de igual manera.*

**Actividad complementaria:**

Investigar:

## ¿Qué es el OpenOcd?

Open On-Chip Debugger (OpenOCD) tiene como objetivo proporcionar depuración, programación en el sistema y pruebas de exploración de límites para dispositivos de destino integrados.

Lo hace con la ayuda de un adaptador de depuración, que es un pequeño módulo de hardware que ayuda a proporcionar el tipo correcto de señalización eléctrica al objetivo que se está depurando. Estos son necesarios ya que el host de depuración (en el que se ejecuta OpenOCD) generalmente no tendrá soporte nativo para dicha señalización, o el conector necesario para conectarse al objetivo.

Dichos adaptadores de depuración admiten uno o más protocolos de transporte, cada uno de los cuales implica una señalización eléctrica diferente (y utiliza diferentes protocolos de mensajería además de esa señalización). Hay muchos tipos de adaptadores de depuración y poca uniformidad en su nombre. (También hay diferencias de nombres de productos).

Estos adaptadores a veces se empaquetan como dongles discretos, que genéricamente pueden llamarse dongles de interfaz de hardware. Algunas placas de desarrollo también las integran directamente, lo que puede permitir que la placa de desarrollo se conecte directamente al host de depuración a través de USB (y, a veces, también para alimentarlo a través de USB).

Por ejemplo, un adaptador JTAG admite señalización JTAG y se utiliza para comunicarse con TAP compatibles con JTAG (IEEE 1149.1) en su placa de destino. Un TAP es un "Puerto de acceso de prueba", un módulo que procesa instrucciones y datos especiales. Los TAP están conectados en cadena dentro y entre chips y placas. JTAG admite operaciones de depuración y escaneo de límites.

También hay adaptadores SWD que admiten la señalización Serial Wire Debug (SWD) para comunicarse con algunos núcleos ARM más nuevos, así como adaptadores de depuración que admiten transportes JTAG y SWD. SWD solo admite la depuración, mientras que JTAG también admite operaciones de exploración de límites.

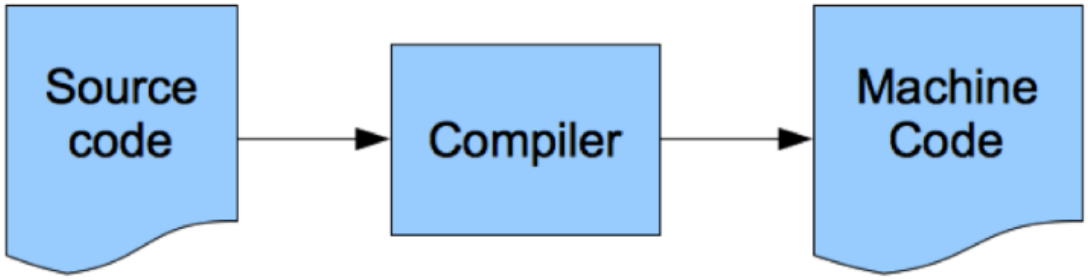
Para algunos chips, también hay adaptadores de programación que admiten transportes especiales que se utilizan solo para escribir código en la memoria flash, sin compatibilidad con la depuración en el chip o el escaneo de límites. (Al momento de escribir esto, OpenOCD no es compatible con dichos adaptadores que no son de depuración).

## ¿Qué es PeriPheral Driver Library?

La biblioteca de controladores periféricos (PDL) simplifica el desarrollo de software para la arquitectura MCU PSoC™ 6. El PDL reduce la necesidad de comprender el uso de registros y las estructuras de bits, lo que facilita el desarrollo de software para el amplio conjunto de periféricos disponibles.
La versión del software ModusToolbox™ de PDL está disponible en el Sitio GitHub de Infineon. No es compatible con PSoC™ Creator. La versión ModusToolbox™ de la PDL incluye compatibilidad con los nuevos controladores y dispositivos MCU PSoC™ 6. También es compatible con hosts macOS y Linux, así como con Windows. Los desarrolladores deben cambiar al paquete ModusToolbox™ según lo permitan los proyectos y los cronogramas.

## ¿Qué es un Compilador?

Es un Software que traduce un programa escrito en un lenguaje de programación de alto nivel (C / C ++, COBOL, etc.) en lenguaje de máquina. Un compilador generalmente genera lenguaje ensamblador primero y luego traduce el lenguaje ensamblador al lenguaje máquina.




## ¿Qué es Depuración?

Un depurador es una herramienta de desarrollo muy especializada que se asocia a la aplicación en ejecución y permite inspeccionar el código. En la documentación de depuración de Visual Studio, esto es normalmente de lo que hablamos cuando decimos "depuración".
El depurador es una herramienta esencial para buscar y corregir errores en las aplicaciones. Sin embargo, el contexto es clave, y es importante aprovechar todas las herramientas de que disponga para ayudarle a eliminar rápidamente errores. A veces, la "herramienta" adecuada puede ser un procedimiento de creación de código mejor. Si aprende a usar el depurador frente en lugar de alguna otra herramienta, también aprenderá a usar el depurador de forma más eficaz.

## ¿Cuál es la diferencia principal entre un microcontrolador y un microprocesador?

Un microcontrolador (a veces llamado MCU o unidad de microcontrolador) es un solo integrado circuito (IC) que se utiliza normalmente para una aplicación específica y está diseñado para implementar determinadas tareas. Los productos y dispositivos que deben controlarse automáticamente en determinadas situaciones,

como electrodomésticos, herramientas eléctricas, sistemas de control de motores de automóviles y computadoras, son excelentes ejemplos, pero los microcontroladores van mucho más allá de estas aplicaciones.

Un microprocesador es un componente electrónico que utiliza una computadora para realizar su trabajo. Es una unidad de procesamiento central en un solo chip de circuito integrado que contiene millones de componentes muy pequeños, incluidos transistores, resistencias y diodos que trabajan juntos.

## ¿Cuáles son los tipos principales de arquitecturas (microcontroladores)?

Anotar principales características de cada uno

Principalmente, en la arquitectura Von Neumann, tanto los datos como las instrucciones transitan por el mismo bus debido a que estos se guardan en la misma memoria, su gran ventaja es ahorrar líneas de entrada-salida pero esto disminuye en cierta medida la velocidad de realizar los procesos.

Esta arquitectura es muy común en los computadores personales, y fue muy utilizado en la elaboración de microcontroladores hasta que se dieron a conocer las ventajas de la arquitectura Harvard.

En la arquitectura Harvard a diferencia de la arquitectura Von Neumann existe una memoria solo para los datos y una memoria solo para las instrucciones, de esta manera se utilizarán dos buses diferentes. Con esto se puede trabajar con las dos memorias al mismo tiempo y por ende la ejecución de los programas es mucho más rápida.

Actualmente, el uso de esta arquitectura en los microcontroladores es la más usada.

## ¿Cuáles son y en qué consisten las herramientas de GNU Toolchain?

GNU toolchain

El GNU toolchain es un término que agrupa a una serie de proyectos que contienen las herramientas de programación producidas por el proyecto GNU. Estos proyectos forman un sistema integrado que es usado para programar tanto aplicaciones como sistemas operativos.

El GNU toolchain es un componente vital en el desarrollo del núcleo Linux, el desarrollo del BSD y software para sistemas embebidos. Partes del toolchain de GNU también son usadas en Solaris Operating Environment y la programación de Microsoft Windows con Cygwin y MinGW/MSYS.

Proyectos que son incluidos en el GNU toolchain

- GNU make - automatización de la estructura y de la compilación.
- La GNU Compiler Collection (GCC) – compiladores para varios lenguajes.
- El GNU Binutils – enlazador, ensamblador y otras herramientas.
- El GNU Debugger (GDB) - un depurador interactivo.
- El GNU build system (autotools) – Autoconf, Autoheader, Automake, Libtool - generadores de makefiles.

**Material:**

- Un osciloscopio .
- Un multímetro digital.
- Una protoboard.
- Resistencias de carbón de valores distintos.
- Un pulsador.
- Tiva LaunchPad
- Leds Ultrabrillantes
- Transistor 2n2222

**Desarrollo de la práctica:**

**Experimento 1: Control de la Corriente con un potenciómetro (arreglo en serie).**

Solicite la revisión del experimento.

**Experimento 2: Arreglos paralelo y mixto.**

Solicite la revisión del experimento.

**Experimento 3: Uso del generador de funciones y del osciloscopio.**

Solicite la revisión del experimento.

**Experimento 4: Uso del capacitor y desfase entre señales.**

Solicite la revisión del experimento.

## Referencias:


- OpenOCD
  https://openocd.org/doc/html/About.html
  Consultado 30 de agosto del 2022.
- Biblioteca de controladores periféricos (PDL) de PSoC™ 6 para PSoC™ Creator
  https://www.infineon.com/cms/en/design-support/software/device-driver-libraries/psoc-6-peripheral-driver-library-pdl-for-psoc-creator/
  Consultado 30 de agosto del 2022.
- ¿Qué es un compilador en programación?

https://www.europeanvalley.es/noticias/que-es-un-compilador-en-programacion/#:~:text=%C2%BFQu%C3%A9%20es%20un%20compilador%3F,lenguaje%20ensamblador%20al%20lenguaje%20m%C3%A1quina.
Consultado 30 de agosto del 2022.

- ¿Qué es la depuración?
  https://docs.microsoft.com/es-es/visualstudio/debugger/what-is-debugging?view=vs-2022
  Consultado 30 de agosto del 2022.
- Microprocesador vs Microcontrolador: ¿Cuál es la diferencia?
  https://www.shunlongwei.com/es/microprocessor-vs-microcontroller-what-is-the-difference/
  Consultado 30 de agosto del 2022.
- Arquitectura de microcontroladores
  https://www.vistronica.com/blog/post/arquitectura-de-microcontroladores.html
  Consultado 30 de agosto del 2022.
- GNU toolchain
  https://es-academic.com/dic.nsf/eswiki/512844
  Consultado 30 de agosto del 2022.

## HOJA DE REVISIÓN (Práctica 1)

Favor de llenar con tinta azul (sólo los nombres de los integrantes).

| Integrante | No. Experimento | | | | Cuestionario |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

_____

Firma del encargado de la práctica