Guía de Instalación de Entorno de Desarrollo

Sistemas con Microprocesadores 14/10/2026

Introducción

Visual Studio Code es un entorno de desarrollo multiplataforma, es decir que puede ejecutarse en los principales sistemas operativos de escritorio como Linux, Windows y MacOS. Tiene la capacidad de compilar y depurar proyectos para múltiples objetivos como ser computadoras de escritorio, dispositivos móviles y sistemas embebidos. La mayoría de los IDE utilizan a *make* para el proceso de compilación y a *gdb* para el proceso de depuración, permitiendo acceder a los mismos de una forma simple y cómoda.
En la siguiente sección se muestra una guía paso a paso para instalar y configurar todas las herramientas necesarias para escribir, compilar, grabar y depurar un programa para la placa edu-unt-rv.
Además, al final del documento se detallan errores comunes que pueden suceder durante la instalación, y como resolverlos.

Instalación y configuración de las herramientas

1) **Windows Subsystem for Linux (WSL)**
   Para acceder a un entorno de Linux virtualizado y facilitar el desarrollo, utilizaremos WSL, una herramienta integrada desde Windows 10 2004 en adelante para tener una VM de Linux dentro de Windows.
   Abra *PowerShell* en modo de administrador; para ello, haga clic con el botón derecho y seleccione ”Ejecutar como administrador”, escriba el comando:
   wsl ——install
   Siga los pasos y reinicie la máquina. Para más información, referencie a este link.

2) **usbipd: Herramienta para compartir dispositivos USB**
   WSL no tiene acceso a los dispositivos USB de Windows, por lo que se necesita una herramienta externa. Esta herramienta es usbipd, y debe ser descargada de este link. Para ello, presione para descargar donde dice ”usbipd-win_*.*.*_x64.msi”, y siga los pasos de instalación.

3) **Visual Studio Code**
   Para editar y compilar el código usaremos el IDE *Visual Studio Code*. El mismo puede descargarse desde este link, donde pueden encontrarse versiones para distintos Sistemas Operativos, pero la Cátedra solo dará soporte para instalaciones bajo Windows y Linux (el cual debe seguir la instalación ignorando la parte de WSL y usbipd). Una vez descargado el archivo, ejecutarlo y seguir los pasos de la instalación, que son bastante sencillos.

4) **Extensión WSL para VS Code**
   Una vez instalado el *Visual Studio Code*, es necesario instalar desde el marketplace del mismo la extensión *WSL*. Esta permite conectarse a un entorno WSL como si se estuviera programando nativamente. En la figura 1 se puede observar como se ve esta extensión desde el mismo IDE.

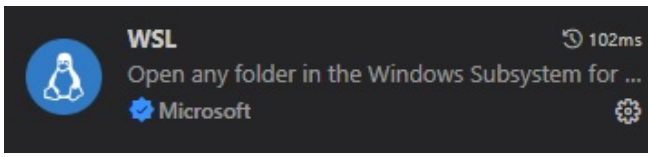


Figura 1: Visualización de la extensión *WSL* dentro del *Visual Studio Code*.

Luego de la instalación, se debe conectar a la instancia de WSL. Para ello, presione en el botón ><ubicado en la esquina inferior izquierda del programa, denotado en la figura 2 y elija en el menú superior la opción ”Connect to WSL” o ”Conectar a WSL”. Si le pide usuario y contraseña, ingresar la configurada durante la instalación de WSL.



Figura 2: Botón para la conexión a WSL

5) **Extensiones para VS Code**
Dentro del entorno de WSL (de ahora en adelante, todo será dentro del mismo, exceptuando que se especifique lo contrario), se deben instalar distintas extensiones gratuitas. Estas son *C/C++*, *C/C++ Extension Pack* y *USBIP Connect*. La primera es usada para depurar los códigos, la segunda sirve para resaltar la sintaxis y la tercera provee una interfaz fácil de usar para usbipd. En la Figura 3 se puede observar cómo se ven estas extensiones desde el mismo IDE.

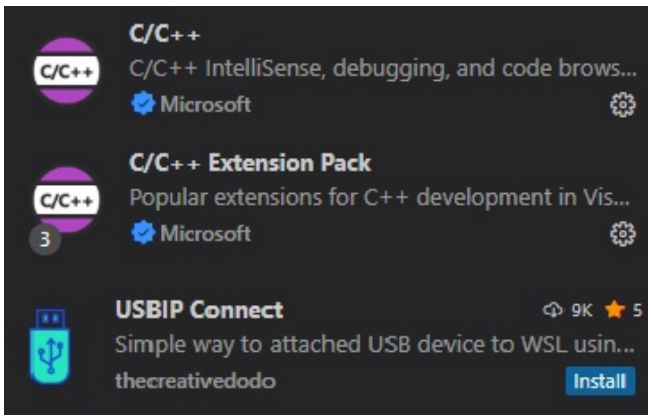


Figura 3: Visualización de las extensiones *C/C++*, *C/C++ Extension Pack* y *USBIP Connect* dentro del *Visual Studio Code*.

6) **Primera conexión a la placa**
Para realizar la primera conexión con la placa de desarrollo, presione en el botón ”Attach” en la barra inferior, y elija el dispositivo de la placa (CMSIS DAP...). Al hacer esto, recibirá un mensaje de error abajo a la derecha, donde le indicará abrir una ventana de PowerShell como administrador (de la misma manera que cuando instaló WSL) y ejecutar el comando dado por el mensaje (el comando tiene la forma usbipd bind...).
Luego de ejecutar el comando, volver a VSCode e intentar presionar de vuelta el botón y elegir el dispositivo. Si no da error, el dispositivo debería dejar de aparecer como dispositivo en Windows, y no debería dar ningún mensaje de error.

7) **Herramientas para RISC-V**
El siguiente paso es descargar e instalar el paquete de herramientas de compilación de RISC-V, conocidas como *GNU RISC-V Embedded Toolchain*.
En este paquete se incluyen varias herramientas, entre las que creemos importante mencionar a:

- GNU C/C++ Compiler (*gcc*): el compilador por defecto.
- GNU Project Debugger (*gdb*): el depurador que nos ayudará a analizar nuestros códigos.
- GNU Binutils: con muchas otras herramientas, como el enlazador (*linker*) y el ensamblador (*assembler*).

Además, se debe instalar Git para el manejo de repositorios, curl para la instalación de las herramientas, build-essential para tener paquetes básicos de compilación y openocd para el flash y debug de la placa.
Para facilitar la instalación, se provee un script de instalación, el cual debe ser descargado este link, y dentro del explorador de archivos de Windows, ir a la nueva pestaña (debajo de Red) denominada Ubuntu, y pegar este archivo en la carpeta /home/%USUARIO%/. Luego, desde la terminal de VSCode, posicionada aquí (aparece como ~/), ejecutar el comando bash installMicro.sh.

Creación del Proyecto

En la Figura 4 se puede observar la pantalla de inicio al abrir el IDE. Si ya tenían abierto el *Visual Studio Code*, les recomendamos que lo cierren y vuelvan a abrirlo (recordando hacer la conexión de WSL como la segunda parte del paso 4 de la instalación), para que refresque todos los cambios realizados.
Para empezar, en la misma seleccionamos la opción *Open Folder* ya sea en el botón en la sección *Start* al centro de la pantalla o en el menú *File*.

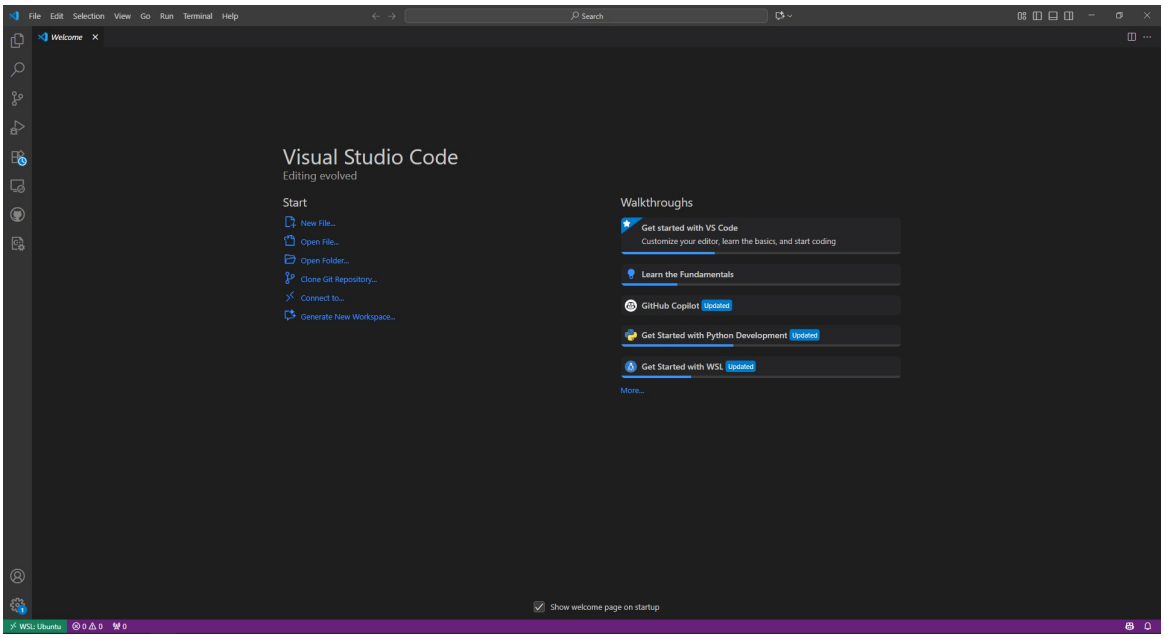


Figura 4: Pantalla de inicio de *Visual Studio Code*.

Esto nos lleva a la pantalla de la Figura 5 donde se pide un nombre de la carpeta y una ubicación de la misma. Seleccionamos la carpeta en la que se creó el entorno de trabajo (paso 7 de la sección anterior). Finalmente presionamos el botón *Ok*.

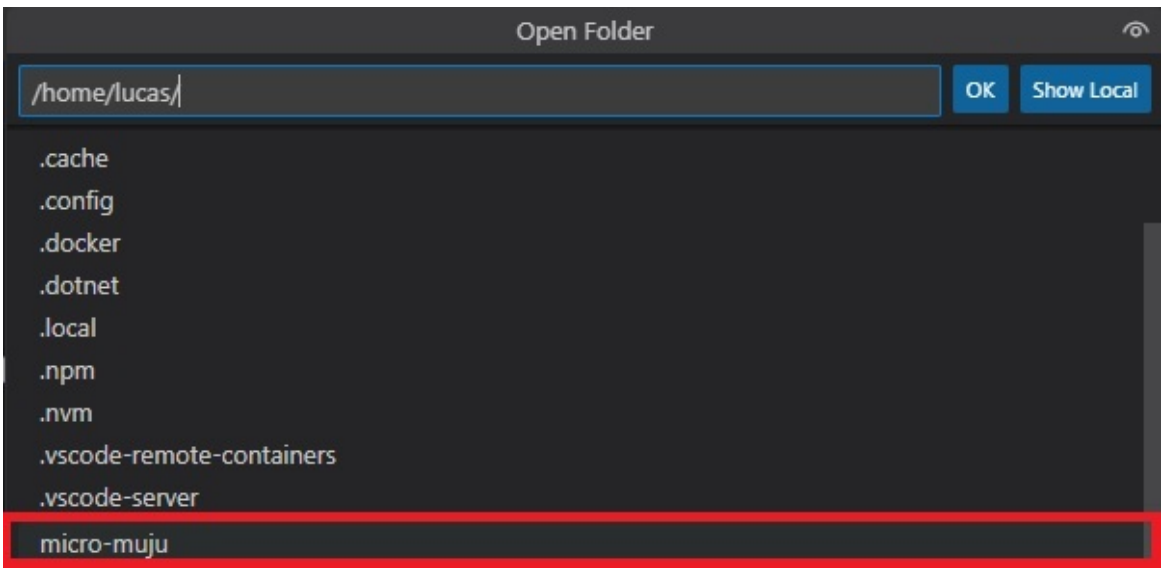


Figura 5: Ubicación y nombre de la Carpeta de Trabajo.

En la Carpeta de Trabajo podemos observar los subdirectorios *examples* y *modules*, y los archivos *proyecto.mk* y *makefile*.

Creación y edición del código

En el panel izquierdo de la pantalla vemos el directorio que hemos abierto. El siguiente paso es la creación de un proyecto con el código que deseamos compilar, para ello se debe presionar click derecho en el directorio raíz y seleccionar la opción *New Folder*, y darle un nombre a la nueva carpeta, por ejemplo *proj1*, como se ve en la figura 6.

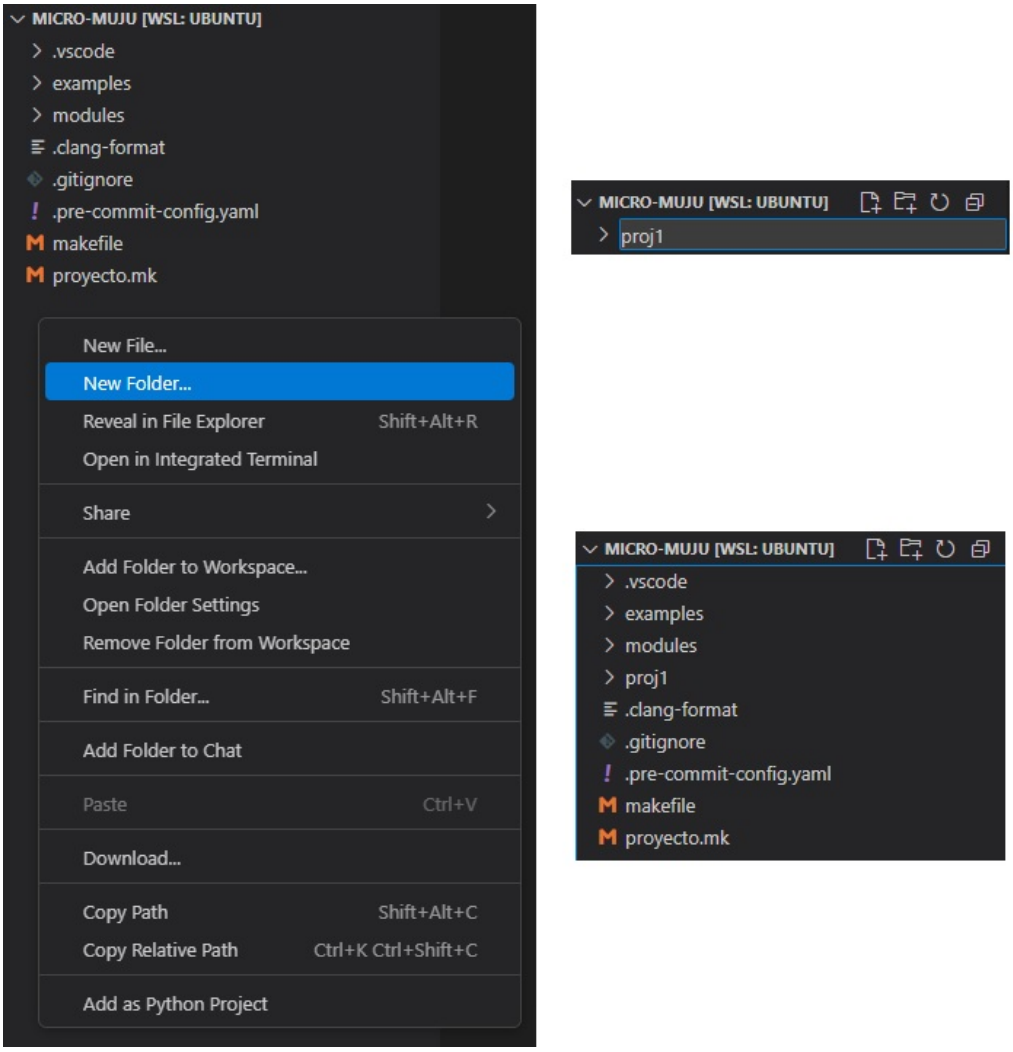


Figura 6: Creación de la carpeta del proyecto.

De la misma forma, para organizar mejor el proyecto, dentro de esta carpeta creamos dos carpetas más, una llamada *src* (donde se guardarán los archivos fuente) y otra llamada *inc* (donde se guardarán los archivos de cabecera), como se ve en la figura 7.

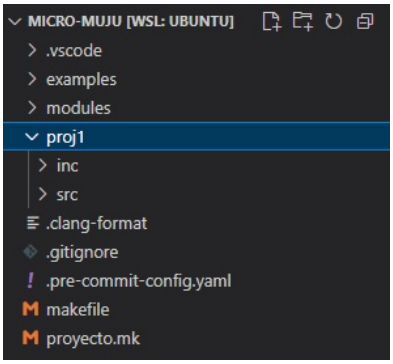


Figura 7: Organización de la carpeta del proyecto.

Siguiendo la organización, creamos los archivos de código del proyecto. Dentro de la carpeta src, deben estar los archivos con extensión *.c* y *.s* (código en C y en Assembler, respectivamente). Dentro de la carpeta inc, deben estar los archivos con extensión *.h*. Para crear estos archivos, se debe presionar click derecho en el directorio, elegir el nombre y extensión del archivo, y presionar *Enter*. En la figura 8 se muestra un ejemplo de cómo debería verse la organización del proyecto.

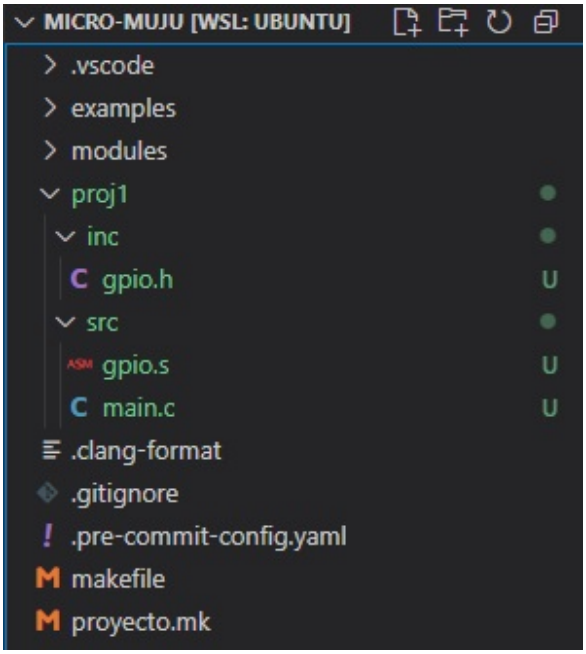


Figura 8: Ejemplo de organización del proyecto.

Compilación y depuración

Para una mejor comprensión, esta sección también se dividirá en varios pasos.

1) **Definir el proyecto a compilar**
   Para la compilación del código primero se debe definir cual será el proyecto que contiene el código fuente a compilar. Para ello editamos el archivo *proyecto.mk* que se encuentra al final del explorador de archivos en la barra lateral izquierda. En este archivo debemos definir el nombre del proyecto a compilar manteniendo la ruta relativa a la carpeta de trabajo. En el ejemplo que está cargado en la Carpeta de Trabajo, está escrito **`PROYECTO ?= examples/demo`**. Cuando escriban sus propios códigos, deberían solamente modificar el nombre del proyecto a compilar.

2) **Abrir una terminal en el IDE**
   Después abrimos una terminal en la raíz del proyecto, ya sea presionando el botón *Terminal* en el panel inferior o presionando click derecho en el directorio raíz eligiendo la opción *Open new Terminal*. En la Figura 9 se muestra que se nos despliega una terminal de comandos en el panel inferior del IDE.

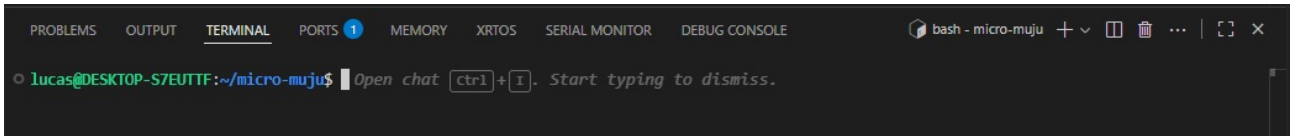


Figura 9: Terminal abierta en el panel inferior de la pantalla.

3) **Compilar**
   Para el paso siguiente, en la terminal abierta escribimos el comando **`make`**, y podremos ver en la salida de la misma terminal si se completó exitosamente la tarea, como se observa en la Figura 10. Si hubiera algún error, se indicaría en la misma salida de la terminal.

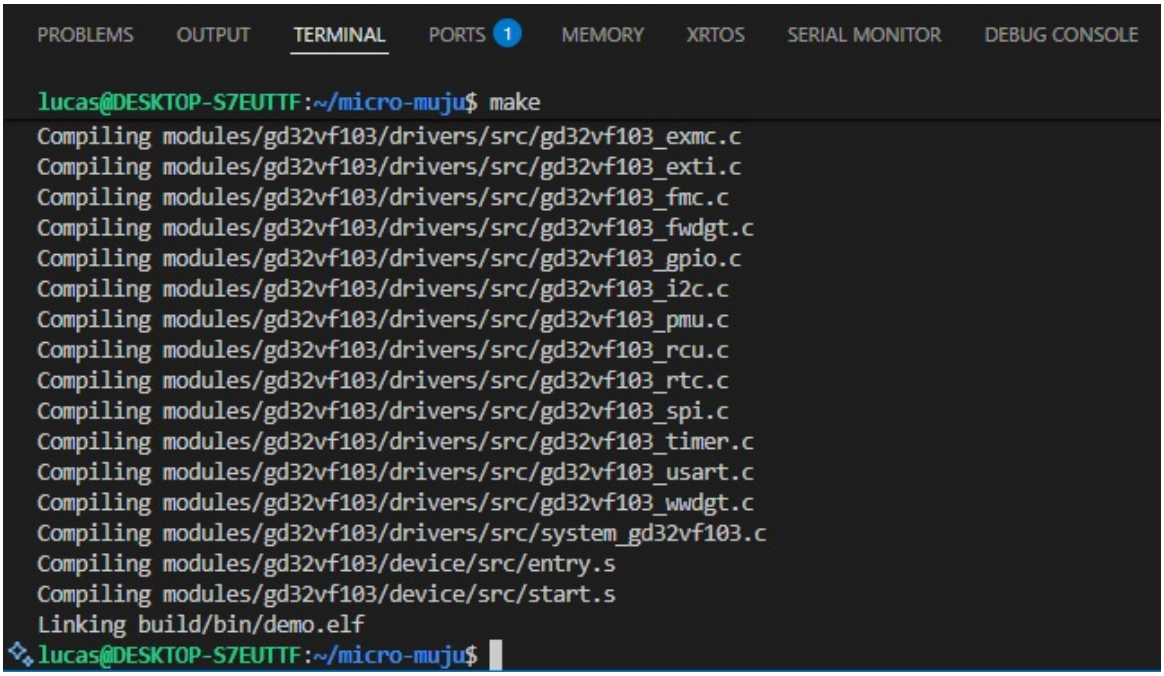


Figura 10: Salida por terminal.

4) **Conectar a la placa**
Si el comando make no entregó errores, ya deberíamos tener los archivos binarios creados dentro de la Carpeta de Trabajo. Por lo tanto, el paso siguiente es empezar con su ejecución, y para ello, debemos conectarnos a la placa.
Para ello, en la barra inferior del IDE, presionamos el botón *Attach*, y en la ventana que se abre seleccionamos el dispositivo correspondiente a la placa (CMSIS DAP...), como se realizó en el paso 6 de la instalación.

5) **Configurar depuración**
Con la placa conectada, el siguiente paso es inicializar el servidor de depuración. Para ello, en la misma terminal donde compilamos, debemos ejecutar el comando **`make gdbserver`**. Este comando inicia el servidor de depuración OpenOCD, el cual se encargará de comunicarse con la placa y permitir la depuración del código. En la Figura 11 se puede observar la salida de la terminal al ejecutar este comando.

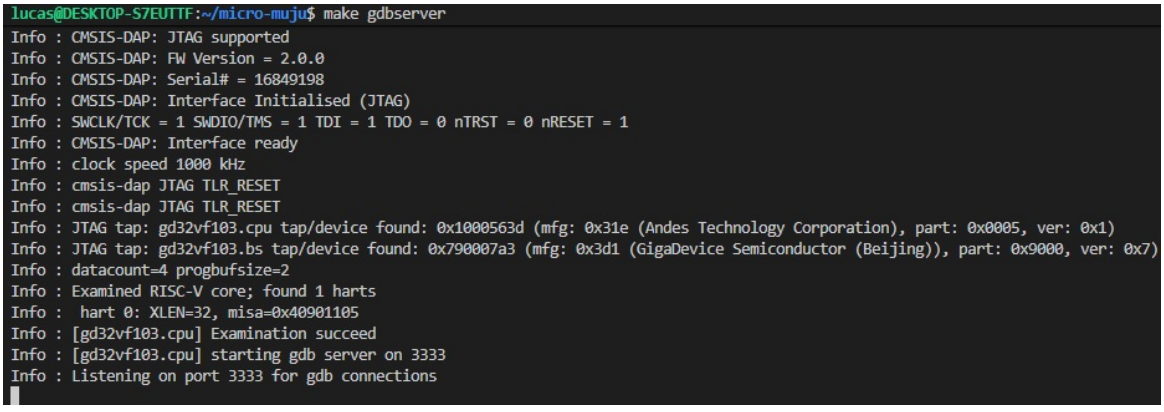


Figura 11: Salida de la terminal al iniciar el servidor de depuración.

6) **Iniciar depuración**
A continuación podemos iniciar la depuración, entrando primero en el menú de depuración, presionando en el icono del panel lateral izquierdo que tiene forma de triángulo con un insecto (rectángulo rojo en la figura 12), o presionando *Ctrl + Shift + D*.
Luego, se debe presionar en el botón *Start debugging* (triángulo verde a la izquierda del ser-

vidor de depuración, rectángulo azul en la figura 12).
A partir de este punto podemos manejar la ejecución de nuestro código con los botones para continuar hasta el próximo *breakpoint* (*Continue (F5)*), para avanzar paso a paso por las instrucciones (*Step Into (F11)*), para reiniciar la ejecución (*Restart*) o para finalizar la ejecución (*Stop*).
Adicionalmente, en el Panel de ejecución de la izquierda podrán observarse las variables y los registros del procesador, con sus valores en decimal. Toda esta información puede observarse en la Figura 12.

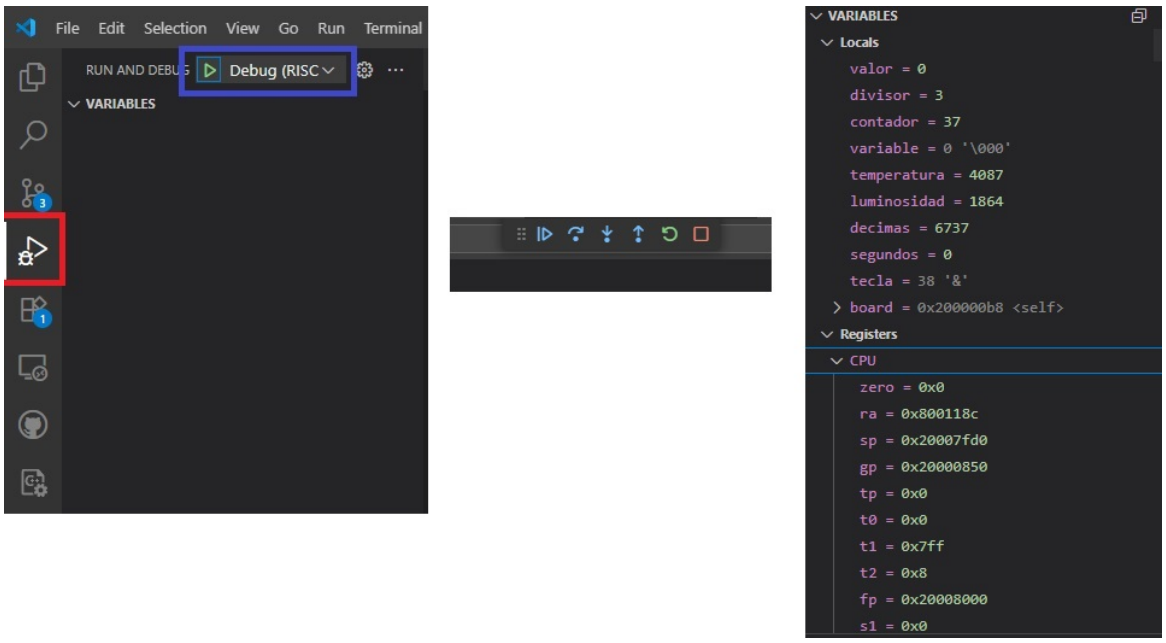


Figura 12: Depurando el proyecto.

7) **Finalizar**
Para terminar y cerrar correctamente, primero debemos detener la depuración con el botón *Stop*, volver a la terminal donde se corrió el último comando y presionar *Ctrl + C* para terminar el servidor de depuración.

Errores comunes

WSL

Si al instalar WSL recibe el error:
(No se pudo iniciar la operación porque no se instaló una característica requerida. Código de error: Wsl/InstallDistro/Service/RegisterDistro/CreateVm/HCS/HCS_E_SERVICE_NOT_AVAILABLE)
Ejecute 2 veces el comando wsl ——install

USB en WSL

unable to find a matching CMSIS-DAP device
Este error corresponde a un problema de permisos en los USB con windows. Ejecutar en el command de windows con permiso de administrador el siguiente comando: usbipd list y deberia aparecer en DEVICE CMSIS-DAP v2, anotar el numero de BUS-ID ejecutar: usbipd attach –wsl –busid x-x
y en la terminal de visual studio ejecutar: sudo chmod -R 777 /dev/bus/usb
con el comando lsusb deberia aparece en wsl CMSIS-DAP