

# Laboratorio de Sistemas Embebidos

## Manual de Instalación del SDK de C/C++ para la Programación del Sistema Raspberry Pi Pico

### **Profesor Teoría**

Ricardo Andrés Velásquez V. (randres.velasquez@udea.edu.co)

### **Profesor Laboratorio**

Luis Germán García M. (german.garcia@udea.edu.co)

Noviembre 30, 2021



## **1 Introducción**

Esta guía describe los pasos necesarios para instalar las herramientas de desarrollo requeridas para programar el sistema Raspberry Pi Pico en Windows empleando los lenguajes de programación C y C++.

El manual "Getting Started with Pico (Ver Ref. 1)" disponible en la página de Raspberry, ofrece instrucciones precisas sobre como realizar la instalación de las herramientas necesarias para la programación en C y C++ de la Raspberry Pi Pico tanto en Windows como en Linux. En Linux, el proceso es relativamente simple. Incluso, existe un *script* para distribuciones basadas en Linux-Debian que realiza todo el proceso de instalación de manera automática. Sin embargo, en Windows, el proceso es un poco complejo, requiriéndose incluso la necesidad de instalar una serie de herramientas de desarrollo para Visual Studio de gran tamaño.

En esta guía se describe una serie de pasos para poner en funcionamiento todo un entorno de programación para la Raspberry Pi Pico, empleando Visual Studio Code y herramientas de código abierto. Los pasos disponibles en esta guía fueron en su mayoría tomados de varios tutoriales disponibles en la WEB, principalmente del ofrecido por Shawn Hymel: How to Set Up Raspberry Pi Pico C/C++ Toolchain on Windows with VS Code (Ver Ref. 2).

## 2 Herramientas a Instalar

Las herramientas necesarias para la programación de microcontroladores RP2040 usando C/C++ (el MCU en el sistema Raspberry Pi Pico) se describen a continuación:

- a. **MinGW** (Minimalist GNU for Windows): colección de herramientas para el desarrollo de aplicaciones en Windows empleando software de código abierto.
- b. **GNU ARM Embedded Toolchain**: herramientas de desarrollo de código abierto para arquitecturas basadas en ARM.
- c. **CMake**: herramienta de código abierto para la gestionar el proceso de compilación de código.
- d. **Python**: el SDK para la Raspberry Pi Pico emplea algunos scripts en Python, por lo que se requiere instalar este lenguaje de programación.
- e. **GIT**: herramienta para la gestión de versiones la cual es requerida para descargar y gestionar el SDK de la Raspberry Pi Pico.
- f. **Visual Studio Code**: es un editor de texto multi-plataforma, cuya capacidad puede ser mejorada con la instalación de diversos *plugins*. Toda la programación y compilación de los programas para la Raspberry Pi Pico será realizada desde esta aplicación.
- g. **SDK Pico Developer Platform**: provee los archivos cabecera (.h), librerías (.lib) y demás herramientas necesarias para la programación de MCUs RP2040 empleando C, C++ y ensamblador.

## 3 Procedimiento para la Instalación

La instalación de las herramientas previamente descritas, junto con la configuración de algunas variables de entorno, se explica a continuación. Las herramientas se instalarán en la carpeta de su disco duro llamada **C:\RPPicoSDK\**. Usted puede cambiar el nombre si así lo desea; sin embargo, se recomienda emplear nombre sin espacios para evitar dificultades con algunas herramientas.

### 3.1 Creación de Carpetas

Cree las siguientes carpetas empleando el explorador de Windows o una terminal:

1. **Carpetas de descargas:** C:\RPPicoSDK\downloads
2. **MinGW:** C:\RPPicoSDK\mingw
3. **GNU ARM Embedded Toolchain:** C:\RPPicoSDK\armcc
4. **CMake:** C:\RPPicoSDK\cmake
5. **GIT:** C:\RPPicoSDK\Git
6. **SDK Pico Developer Platform:** C:\RPPicoSDK\sdk

### 3.2 Instalación de MinGW

Emplee el siguiente enlace para descargar la herramienta (en el momento de escritura de esta guía se descargó la versión: 8.1.0):

<https://www.mingw-w64.org/downloads/#mingw-builds>

Ejecute el archivo descargado para instalar el programa en la carpeta C:\RPPicoSDK\mingw. Emplee las opciones que se muestran en la Fig. 1. Para las demás ventanas, use las opciones por defecto.

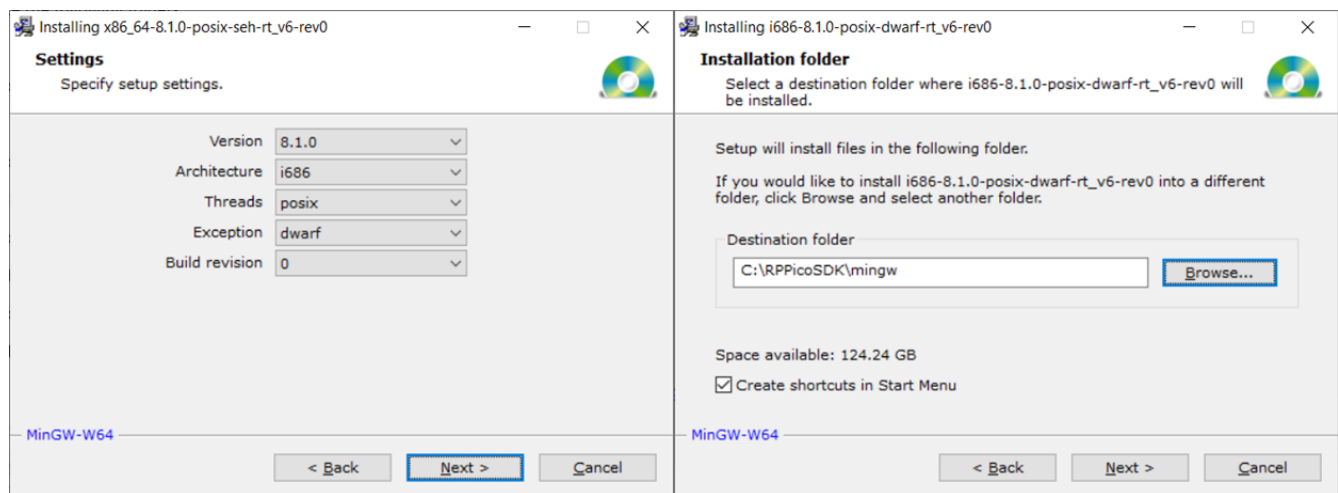


Fig. 1: Opciones de configuración y carpeta de instalación para MinGW

Cuando la instalación finalice, cree un archivo usando el block de notas con la siguiente ruta y nombre: C:\RPPicoSDK\mingw\mingw32\bin\make.bat. El archivo deberá tener la siguiente línea de código:

```
mingw32-make %*
```

### 3.3 Instalación de GNU ARM Embedded Toolchain

Emplee el siguiente enlace para descargar la herramienta (en el momento de escritura de esta guía se descargó la versión: 10.3-2021.10-win32):

<https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads>

Ejecute el archivo descargado para instalar el programa en la carpeta C:\RPPicoSDK\armcc. Emplee las opciones que se muestran en la Fig. 2. Note que el programa de instalación agrega la versión actual al final de la carpeta. En la última ventana, debe seleccionar la opción "Add path to environment variable". Para las demás ventanas, use las opciones por defecto.

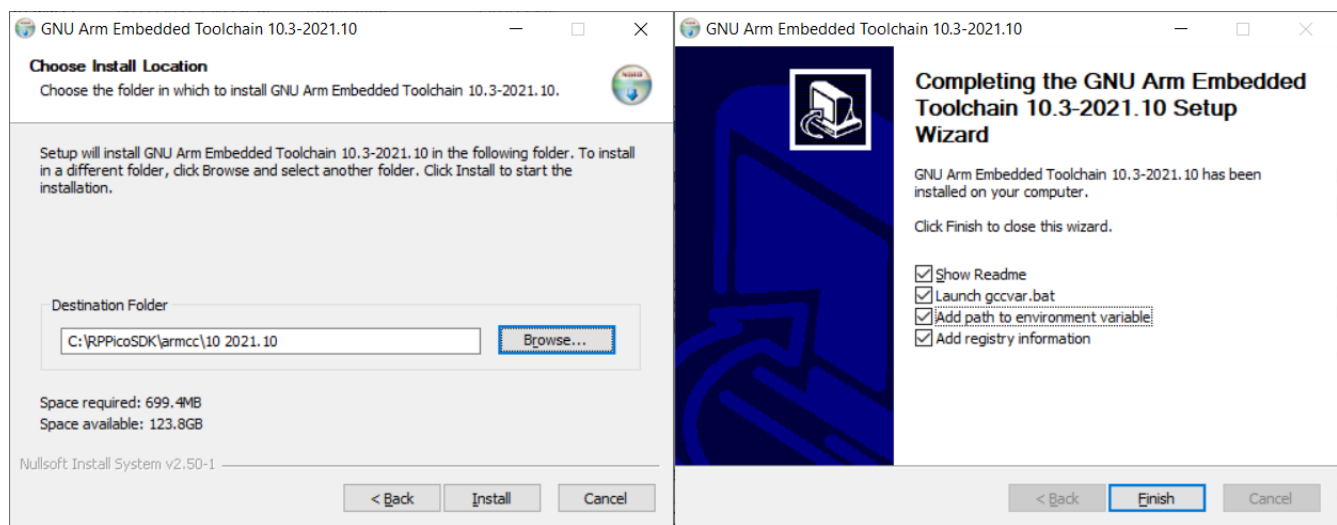


Fig. 2: Opciones de configuración y carpeta de instalación para ARM CC

### 3.4 Instalación de CMake

Emplee el siguiente enlace para descargar la herramienta (en el momento de escritura de esta guía se descargó la versión: 3.22.0-windows-x86\_64):

<https://cmake.org/download/>

Ejecute el archivo descargado para instalar el programa en la carpeta C:\RPPicoSDK\cmake. Emplee las opciones que se muestran en la Fig. 3. Debe seleccionar la opción "Add CMake to the system PATH for all users". Para las demás ventanas, use las opciones por defecto.

### 3.5 Instalación de Python

Emplee el siguiente enlace para descargar la herramienta (en el momento de escritura de esta guía se descargó la versión: 3.10.0-amd64):

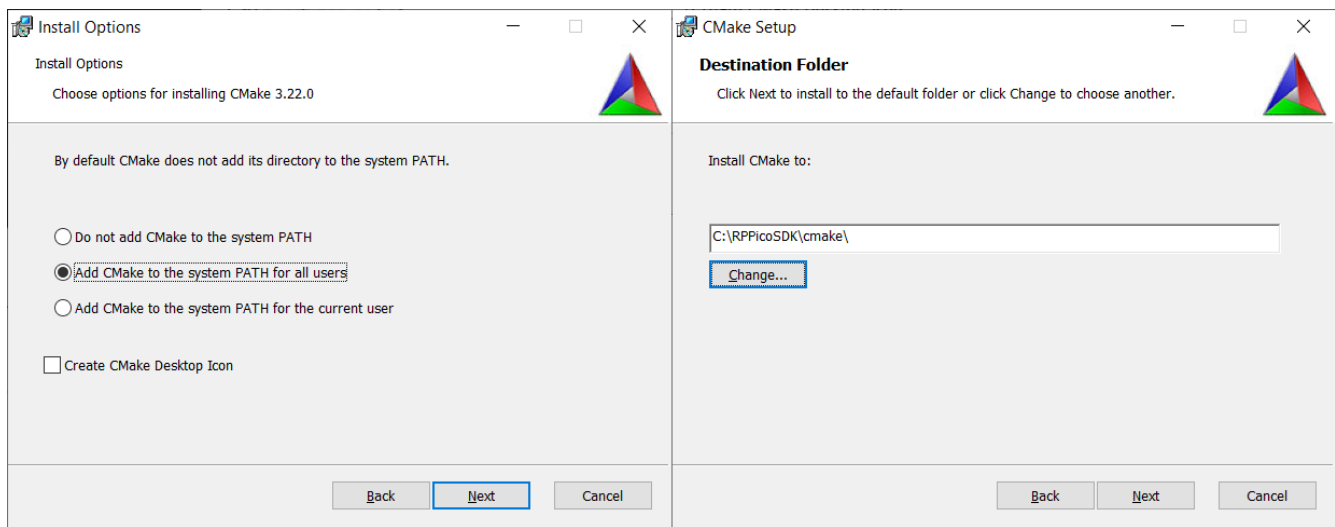


Fig. 3: Opciones de configuración y carpeta de instalación para CMake

<https://www.python.org/downloads/>

Ejecute el archivo descargado para instalar el programa en una de las carpetas del sistema (puede dejar la sugerida por el programa de instalación). Debe seleccionar la opción "Add Python to PATH". Para las demás ventanas, use las opciones por defecto.

### 3.6 Instalación de GIT

Emplee el siguiente enlace para descargar la herramienta (en el momento de escritura de esta guía se descargó la versión: 2.34.1-64-bit):

<https://git-scm.com/download/win>

Ejecute el archivo descargado para instalar el programa en la carpeta C:\RPPicoSDK\Git. Emplee las opciones que se muestran en la Fig. 4. Para las demás ventanas, use las opciones por defecto.

### 3.7 Descarga de Pico SDK y Ejemplos

Empleando la terminal **Git Bash** (la encuentra en el Menú Inicio de Windows) que se instaló anteriormente, ejecute los siguientes comandos:

1. Cámbiese a la carpeta C:\RPPicoSDK\sdk: `cd /c/RPPicoSDK/sdk/`
2. Cree la carpeta pico: `mkdir pico`
3. Cámbiese a la carpeta pico: `cd pico`
4. Clone SDK: `git clone -b master https://github.com/raspberrypi/pico-sdk.git`

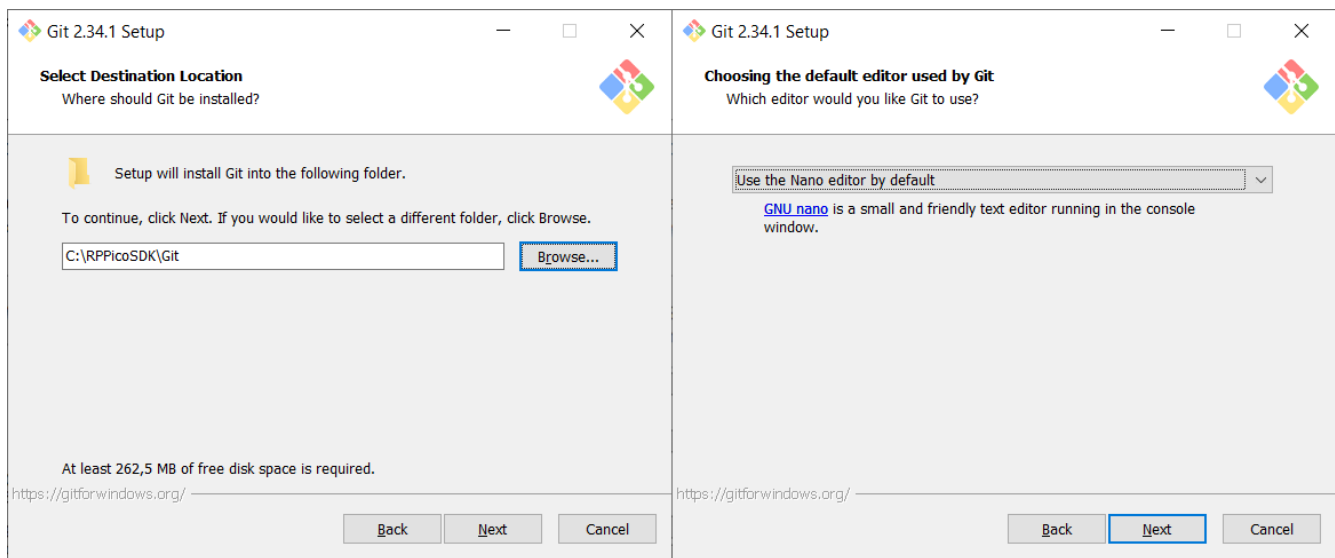


Fig. 4: Opciones de configuración y carpeta de instalación para GIT

5. Cámbiese a la carpeta pico-sdk: **cd pico-sdk**
6. Ejecute: **git submodule update --init**
7. Devuélvase a la carpeta anterior: **cd ..**
8. Clone ejemplos: **git clone -b master https://github.com/raspberrypi/pico-examples.git**

La Fig. 5 muestra el resultado de ejecutar cada uno de los anteriores comandos.

### 3.8 Instalación de Visual Studio Code

Emplee el siguiente enlace para descargar la herramienta (en el momento de escritura de esta guía se descargó la versión: 1.62.3):

<https://code.visualstudio.com/>

Ejecute el archivo descargado para instalar el programa en una de las carpetas del sistema (puede dejar la sugerida por el programa de instalación). Use las opciones por defecto para la instalación.

### 3.9 Actualización de Variables de Entorno

Algunas de las herramientas instaladas previamente han modificado ciertas variables de entorno con el fin de permitir el uso fácil de sus comandos desde cualquier terminal. Sin embargo, herramientas como MinGW y el SDK no lo han hecho, por lo que es necesario que usted realice los cambios manualmente.

```

lgerm@Laptop-Predator-LGGM MINGW64 /c/RPPicoSDK/sdk
$ cd /c/RPPicoSDK/sdk/

lgerm@Laptop-Predator-LGGM MINGW64 /c/RPPicoSDK/sdk
$ mkdir pico

lgerm@Laptop-Predator-LGGM MINGW64 /c/RPPicoSDK/sdk
$ cd pico

lgerm@Laptop-Predator-LGGM MINGW64 /c/RPPicoSDK/sdk/pico
$ git clone -b master https://github.com/raspberrypi/pico-sdk.git
Cloning into 'pico-sdk'...
remote: Enumerating objects: 4600, done.
remote: Counting objects: 100% (1781/1781), done.
remote: Compressing objects: 100% (884/884), done.
remote: Total 4600 (delta 1058), reused 1198 (delta 735), pack-reused 2819
Receiving objects: 100% (4600/4600), 2.39 MiB | 4.57 MiB/s, done.
Resolving deltas: 100% (2254/2254), done.

lgerm@Laptop-Predator-LGGM MINGW64 /c/RPPicoSDK/sdk/pico
$ cd pico-sdk/

lgerm@Laptop-Predator-LGGM MINGW64 /c/RPPicoSDK/sdk/pico/pico-sdk (master)
$ git submodule update --init
Submodule 'tinysub' (https://github.com/hathach/tinysub.git) registered for path 'lib/tinysub'
Cloning into 'C:/RPPicoSDK/sdk/pico/pico-sdk/lib/tinysub'...
Submodule path 'lib/tinysub': checked out '4bfab30c02279a0530e1a56f4a7c539f2d35a293'

lgerm@Laptop-Predator-LGGM MINGW64 /c/RPPicoSDK/sdk/pico/pico-sdk (master)
$ cd ..

lgerm@Laptop-Predator-LGGM MINGW64 /c/RPPicoSDK/sdk/pico
$ git clone -b master https://github.com/raspberrypi/pico-examples.git
Cloning into 'pico-examples'...
remote: Enumerating objects: 1854, done.
remote: Counting objects: 100% (1160/1160), done.
remote: Compressing objects: 100% (657/657), done.
remote: Total 1854 (delta 721), reused 764 (delta 502), pack-reused 694
Receiving objects: 100% (1854/1854), 6.58 MiB | 7.26 MiB/s, done.
Resolving deltas: 100% (968/968), done.

```

Fig. 5: Comandos para descargar SDK y Ejemplos

Para editar las variables de entorno, tanto del usuario como del sistema, abra la barra de búsqueda de Windows y escriba var. De click en la opción llamada **Edit the system environment variables** (Editar las variables de entorno del sistema) como se muestra en la Fig. 6. Se abre la ventana que se muestra a la izquierda en la Fig. 7. De click en el botón **Environment Variables** (Variables de Entorno) para que se abra la ventana que se muestra a la derecha de la misma figura.

Sobre la lista llamada **User variables for ...** (Variables de usuario para ...), edite la variable llamada **Path**, seleccionándola y dando click en el botón **Edit**. En la ventana que aparece (Fig. 8):

1. Agregue la entrada: C:\RPPicoSDK\mingw\mingw32\bin.
2. Verifique que exista: C:\RPPicoSDK\armcc\jversión instalada\bin.

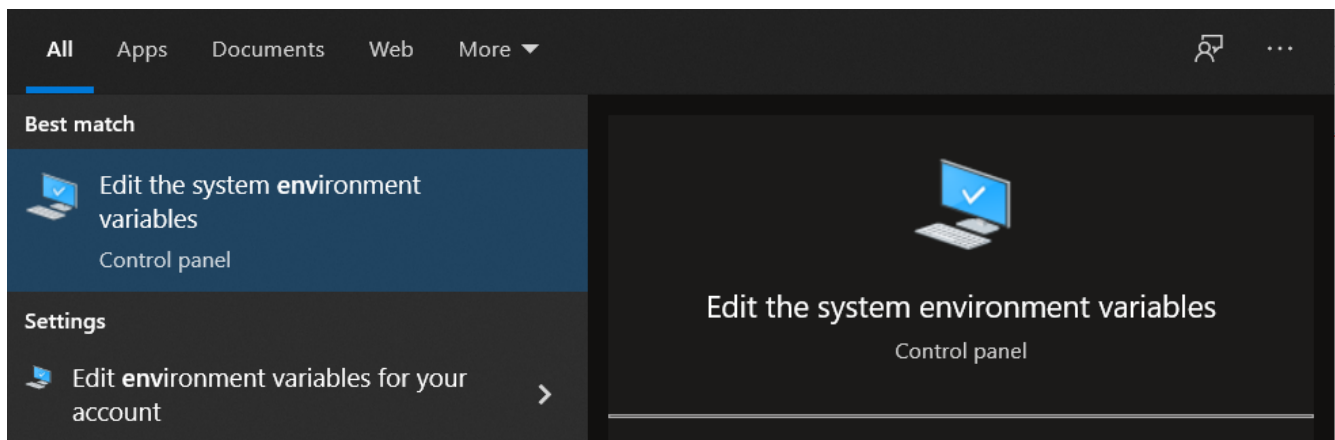


Fig. 6: Edición de variables de entorno - comando

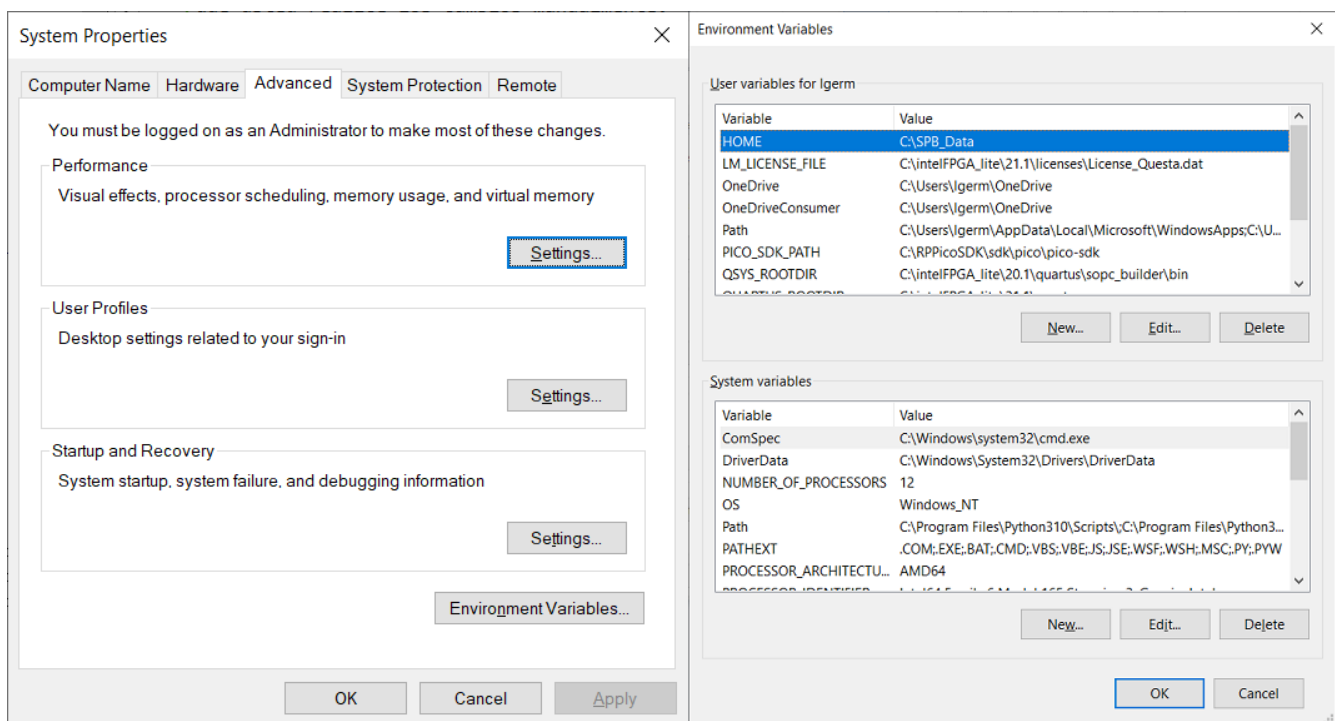


Fig. 7: Variables de entorno

Sobre la misma lista llamada **User variables for ...** (Variables de usuario para ...), agregue una nueva variable llamada **PICO\_SDK\_PATH** con el siguiente valor (como se muestra en la Fig. 9):

C:\RPPicoSDK\sdk\pico\pico-sdk

Finalmente, sobre la lista llamada **System variables** (Variables del sistema), edite la variable llamada **Path**, seleccionándola y dando click en el botón **Edit**. En la ventana que aparece,



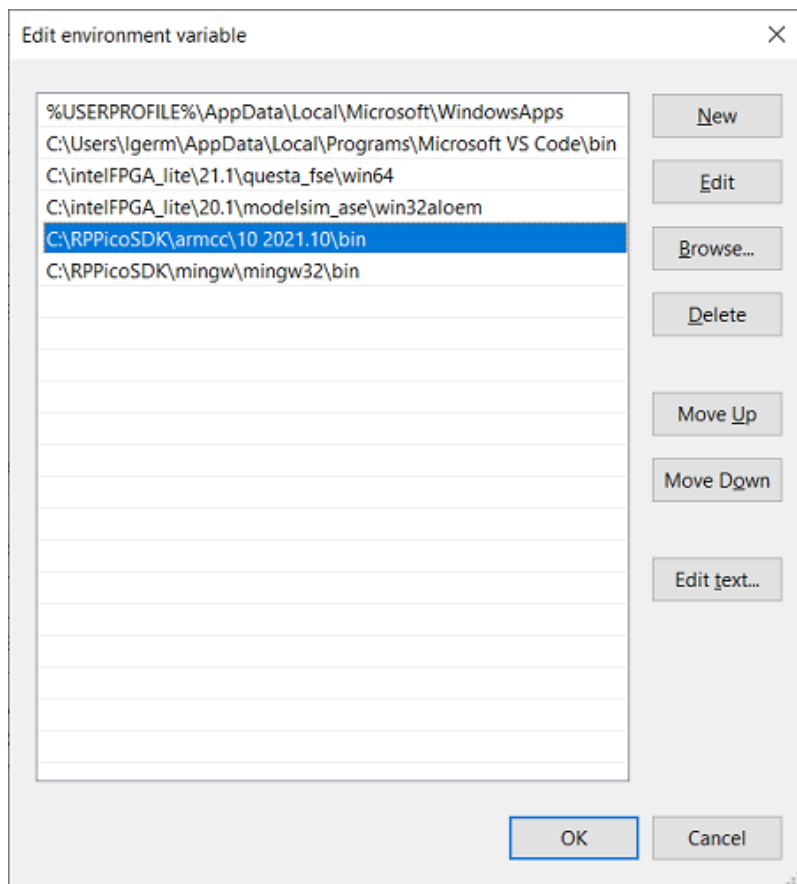


Fig. 8: Editando variable PATH

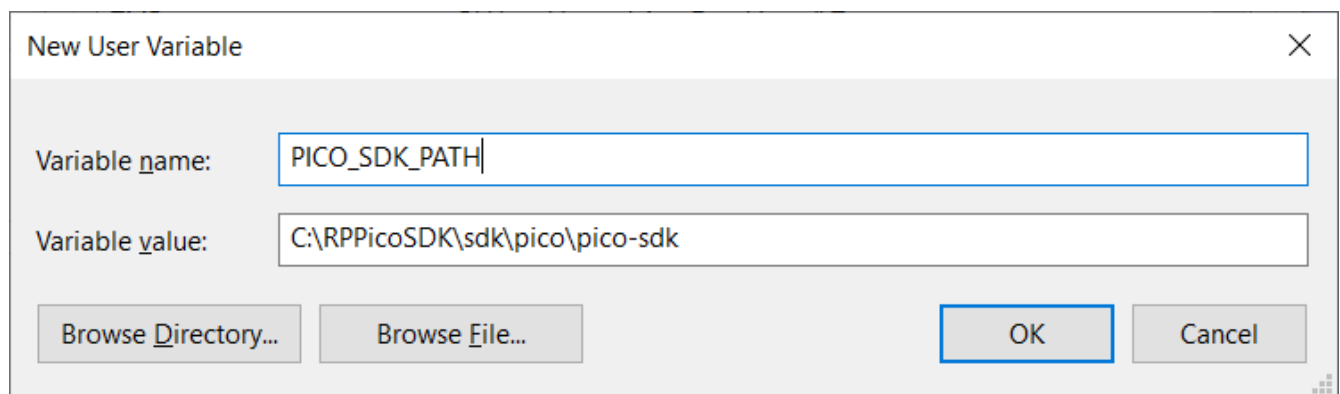


Fig. 9: Agregando variable PICO\_SDK\_PATH

verifique que existan las siguientes dos entradas como se muestra en la Fig. 10:

1. C:\RPPicoSDK\cmake\bin
2. C:\RPPicoSDK\Git\bin

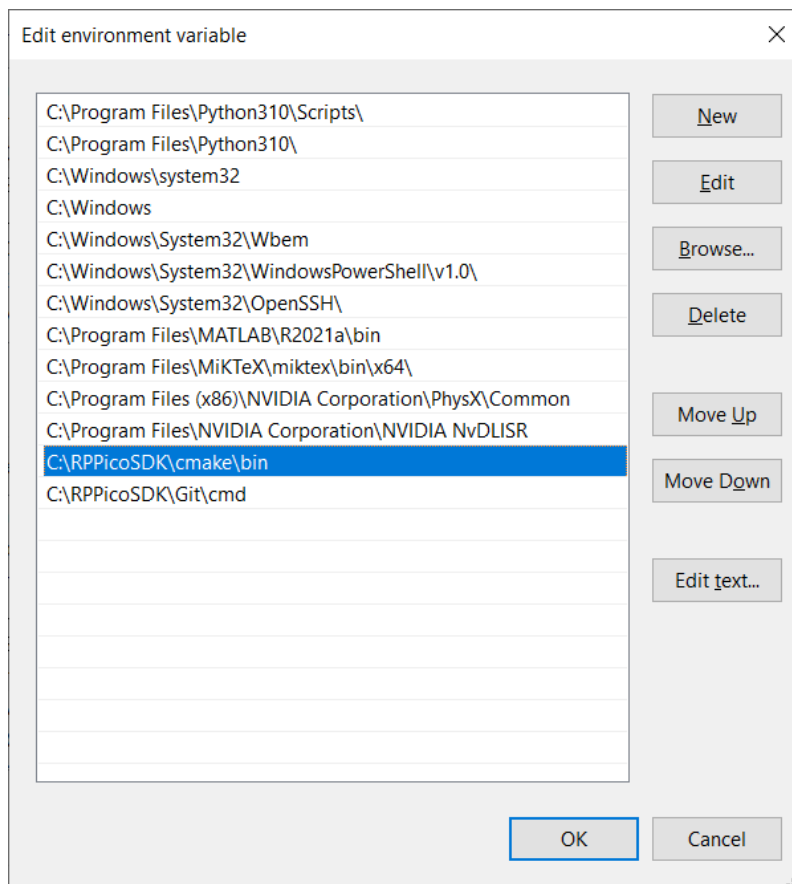


Fig. 10: Editando variable PATH

## 4 Compilación y Programación de un Primer Ejemplo del SDK

Con las herramientas que se han instalado, es posible realizar, a través de comandos en una terminal o el uso de VS Code, la edición y compilación de programas para el Microcontrolador RP2040 del sistema Raspberry Pi Pico. En este caso, emplearemos la terminal Git Bash dentro del entorno de VS Code para realizar la compilación de uno de los ejemplos disponibles en el SDK.


1. Abra **VS Code**. Una vez abierto, de click en el menú **Terminal** y seleccione **New Terminal**. La terminal por defecto es PowerShell; sin embargo, es posible cambiarla por la terminal Bash que viene con Git o usar la terminal Bash que viene con WSL (Windows Subsystem for Linux). Si Git se instaló en la carpeta por defecto, debería estar disponible en VS Code con el nombre Git Bash. En caso no le aparezca, siga los siguientes pasos para agregarla:
  - a. De click en el menú **File** → **Preferences** → **Settings** o presione **Ctrl + ,**.
  - b. En la parte superior derecha, de click en el ícono llamado **Open Settings (JSON)**.

- c. Agregue las siguientes líneas de código al final del archivo que aparece, pero antes de la llave que cierra (}):

```
"terminal.integrated.profiles.windows": {  
    "My Git Bash": {  
        "path": "C:\\\\RPPicoSDK\\\\Git\\\\bin\\\\bash.exe"  
    }  
},  
"terminal.integrated.defaultProfile.windows": "My Git Bash"
```

- d. Cambie la terminal PowerShell por My Git Bash.

2. En la terminal Bash, ejecute los siguientes comandos para crear un alias al comando mingw32-make (solo es necesario una vez):

```
echo "alias make=mingw32-make.exe" >>  .bashrc  
source ~/.bashrc
```

3. Prepare los archivos requeridos para poder realizar el proceso de compilación de cualquiera de los ejemplos:

```
cd /c/RPPicoSDK/sdk/pico/pico-examples/  
mkdir build  
cd build  
cmake -G "MinGW Makefiles" ..
```

4. Compile el ejemplo Blink disponible en el SDK:

```
cd /c/RPPicoSDK/sdk/pico/pico-examples/build/blink/  
make
```

El comando make se encargará de compilar los archivos en C/C++ y generar varios archivos (blink.bin, blink.elf, blink.uf2, etc.) que podrán ser empleados para descargar el ejecutable en la Raspberry Pi Pico.

5. Conecte la Raspberry Pi Pico a su PC usando un cable micro-USB mientras mantiene presionado el botón BOOTSEL. Lo anterior hará que la Raspberry Pi Pico ingrese en modo Bootloader. Identifique la letra de unidad asignada por Windows (ejemplo H:). Ejecute el siguiente comando para copiar el ejecutable a la Raspberry Pi Pico:

```
cp blink.uf2 /letra-unidad-asignada-por-windows/
```


El comando anterior descargará el ejecutable de la aplicación Blink y reiniciará el MCU de la Raspberry Pi Pico. Usted deberá ver el LED parpadeando.

## 5 Compilación y Programación de MyBlink

Con las herramientas que se han instalado, es posible realizar, a través de comandos en una terminal o el uso de VS Code, la edición y compilación de programas para el Microcontrolador RP2040 del sistema Raspberry Pi Pico. En este caso, emplearemos la terminal Git Bash dentro del entorno de VS Code para realizar la compilación de uno de los ejemplos disponibles en el SDK.

1. Abra **VS Code**. Instale el plugin **CMake Tools** para facilitar el proceso de compilación en VS Code:
  - a. De click en el botón **Extensions** ubicado en la parte izquierda de la ventana.
  - b. Escriba en el recuadro el texto **cmake**. Instale **CMakeTools**.
2. Cree una carpeta para el proyecto MyBlink usando el Explorador de Windows u otra aplicación.
3. Vaya al menú File → Open Folder... y seleccione la carpeta que acaba de crear.
4. En el EXPLORER de VS Code, seleccione MYBLINK y de click en New File. Asígnele el nombre main.c.
5. Agregue el código que aparece en la parte izquierda de Fig. 11.
6. Cree un segundo archivo llamado: CMakeLists.txt.
7. Agregue el código que aparece en la parte derecha de Fig. 11.
8. Abra una terminal Bash, ejecute los siguientes comandos:

```
mkdir build
cd build
cmake -G "MinGW Makefiles" ..
```


9. Compile el proyecto: make
10. Conecte la Raspberry Pi Pico a su PC usando un cable micro-USB mientras mantiene presionado el botón BOOTSEL. Lo anterior hará que la Raspberry Pi Pico ingrese en modo Bootloader. Identifique la letra de unidad asignada por Windows (ejemplo H:). Ejecute el siguiente comando para copiar el ejecutable a la Raspberry Pi Pico:

```
cp myblink.uf2 /letra-unidad-asignada-por-windows/
```

## 6 Referencias

- a. Getting Started with Pico Manual  
<https://datasheets.raspberrypi.org/pico/getting-started-with-pico.pdf>
- b. How to Set Up Raspberry Pi Pico C/C++ Toolchain on Windows with VS Code  
<https://shawnhymel.com>

<b>C</b> main.c <pre> 1  #include &lt;stdio.h&gt; 2  #include "pico/stdlib.h" 3 4  const uint led_pin = 25; 5 6  int main() { 7      // Inicialización del Pin para el LED 8      gpio_init(led_pin); 9      gpio_set_dir(led_pin, GPIO_OUT); 10 11     // Loop infinito 12     for (;;) { 13         // Blink LED 14         gpio_put(led_pin, true); 15         sleep_ms(500); 16         gpio_put(led_pin, false); 17         sleep_ms(500); 18     } 19 } 20 21 22 23 24 25 26</pre>	<b>M</b> CMakeLists.txt <pre> 1  # Versión mínima requerida de CMake 2  cmake_minimum_required(VERSION 3.12) 3 4  # Incluir funciones de SDK de PICO 5  include(\$ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake) 6 7  # Nombre del proyecto y estándares del lenguaje aceptados 8  project(myblink C CXX ASM) 9  set(CMAKE_C_STANDARD 11) 10 set(CMAKE_CXX_STANDARD 17) 11 12 # Inicialización de varios elementos del proyecto 13 pico_sdk_init() 14 15 # Indicarle a CMake los archivos de nuestro programa 16 add_executable(\${PROJECT_NAME} 17       main.c 18 ) 19 20 # Crear archivos de salida 21 pico_add_extra_outputs(\${PROJECT_NAME}) 22 23 # Enlazar a las librerías requeridas 24 target_link_libraries(\${PROJECT_NAME} 25       pico_stdlib 26 )</pre>
--	---

Fig. 11: Código de MyBlink: main.c y CMakeLists.txt