

Raspberry Pi Projects



Benchmark residents share an average of 19 daily interactions.

Ra

Se

Mc

Ca

Ta

Schedule Tour

Projects >> [Development Board - Kits Projects](#) >> Setting up the Raspberry Pi Pico for C/C++ Development c

Raspberry Pi Pico for C/C++ Development on Windows

Jhhammad Bilal

ent Board - Kits Projects

, studio

o the Raspberry Pi Pico for C/C++ Development on Windows

aspberry Pi Pico C/C++ Development Setup:



alled

4 Getting the SDK and examples

5 Building the example projects from the Command Line

6 Programming the Pico

7 Building the example projects from Visual Studio Code

Download Now

For windows 10, 11 32/64 bit



Summary of Setting up the Raspberry Pi Pico for C/C++ Development on Windows

This article provides a step-by-step guide to set up a Windows environment for [Raspberry Pi](#) Pico C/C++ development. It covers installing necessary tools like ARM GCC compiler, CMake, Visual Studio Code, Build Tools for Visual Studio 2019 with Windows 10 SDK, Python 3.7, and Git. The article explains how to clone the Pico SDK and example projects, build them via the command line or Visual Studio Code, and program the Pico by dragging the compiled UF2 file onto the board. It ensures a streamlined process for beginners to prepare and program the Raspberry Pi Pico using a Windows PC.

Parts used in the Raspberry Pi Pico C/C++ Development Setup:

- Raspberry Pi Pico Board

Hi there! I'm your AI assistant. Ask me anything or tell me what you're building. I'll help.

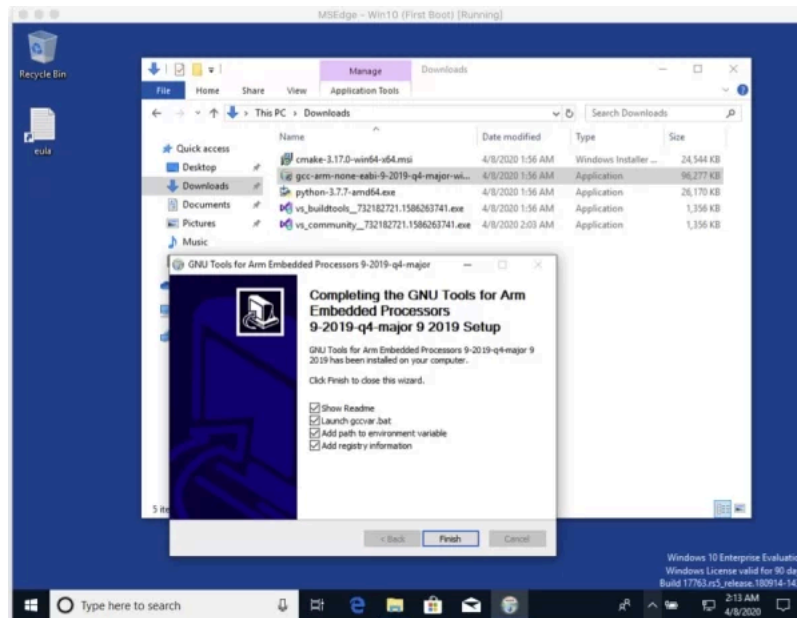
- Build Tools for Visual Studio 2019 (with Windows 10 SDK) ▾
- Python 3.7
- Git
- USB Cable for Pico connection

Documentation

Everything in this file originates from the comprehensive guide “[Getting started with Raspberry Pi Pico for C/C++ development](#)”. This document presents the same content, but reformatted into a user-friendly, numbered list for easier comprehension.

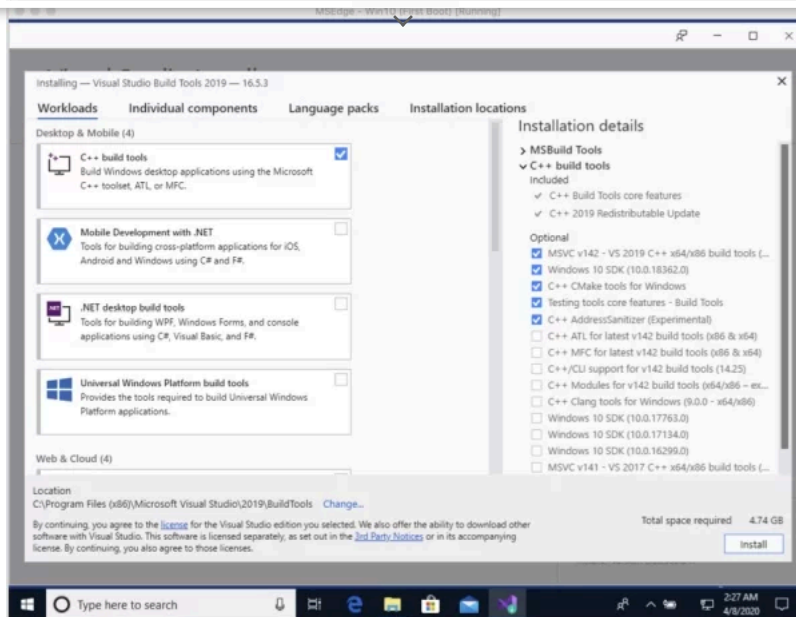
Getting everything installed

1. Install the [ARM GCC compiler](#).
 - Ensure proper installation by clicking the box to register the ARM compiler’s path as an environmental variable within the Windows shell when requested to do so. For reference, please examine the image provided below.



1. Install [CMake](#).
 - When prompted, add CMake to the system PATH for all users.
2. Install [Visual Studio Code](#)
3. Install [Build Tools for Visual Studio 2019](#)
 - When the Built Tools for Visual Studio installer prompts you, be sure to select the “C++ build tools only” installation option.
 - You must install the full “Windows 10 SDK” package as the SDK will need to build the `picoasm` and `e1f2uf2` tools locally. Removing it from the list of installed items will mean that you will be unable to build Raspberry Pi Pico binaries.
 - This takes a while.

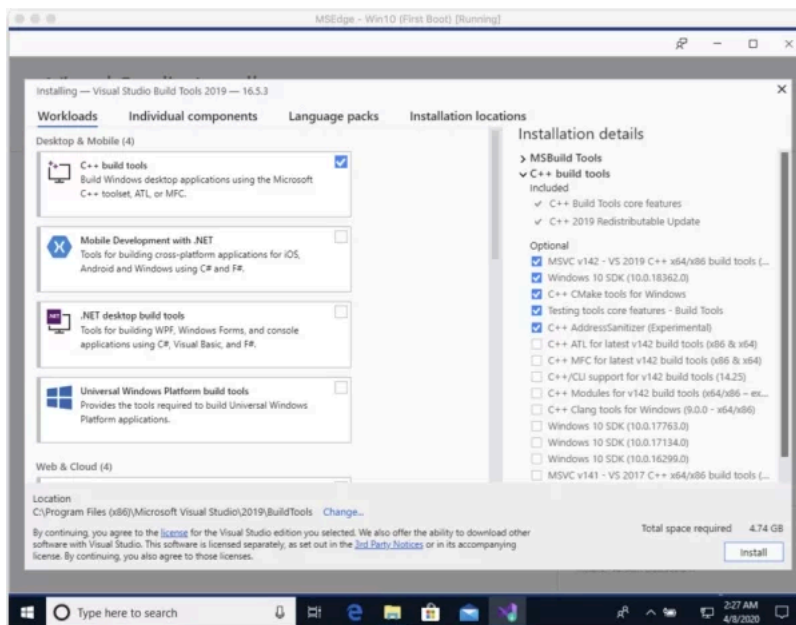
Hi there! I'm your AI assistant. Ask me anything or tell me what you're building. I'll help.



Note checkboxes on right side of image during Build Tools install

Install [Python 3.7](#)

- When prompted by the installer, add Python 3.7 to the system PATH for all users.



4th checkbox from top

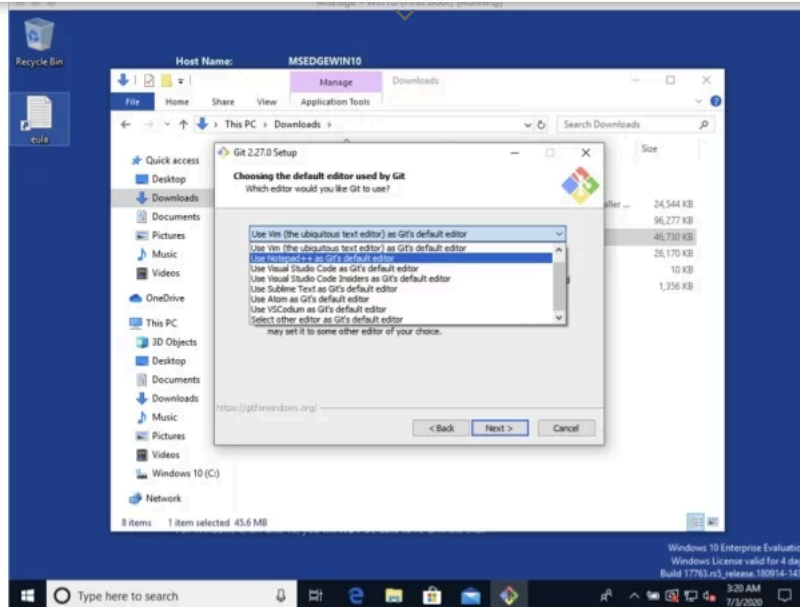
- You should be additionally disable the MAX_PATH length when prompted at the end of the installation.
- When you're setting up Pwnagotchi, make sure to select the "Custom installation" option, navigate to the "Optional Features" section, and subsequently choose "Install for all users" under the "Advanced Features" tab.
- It is possible that you will need to make a symbolic link so that the Makefile can find Python 3. To do so, type cmd in the Run Window so that the Developer Command Prompt icon appears in the Start Menu. Select the small arrow to the right of the icon, and then select "Run as administrator." Navigate to C:\Program Files\Python37 and make a symlink by running

```
C:\Program Files\Python37 > mklink python3.exe python.exe
```

Only do this if your build fails because Make can't find your python installation.

1. Install [Git](#).

Hi there! I'm your AI assistant. Ask me anything or tell me what you're building. I'll help.



Setting default editor to Notepad++ instead of vim

- Tick the checkbox to allow Git to be used from third-party tools.
- Check the box "Checkout as is, commit as-is" (unless you have a strong reason not to)
- Select "Use Windows' default console window"
- Select "Enable experimental support for pseudo consoles"

Getting the SDK and examples

1. Open Windows PowerShell, and create a directory where you'd like to store all the Pico examples and the SDK. I put mine in C:\Users\vha3\Pico
2. Run the following set of commands:

```
C:\Users\vha3\Pico> git clone -b master https://github.com/raspberrypi/pico-sdk.git
C:\Users\vha3\Pico> cd pico-sdk
C:\Users\vha3\Pico\pico-sdk> git submodule update --init
C:\Users\vha3\Pico\pico-sdk> cd ..
C:\Users\vha3\Pico> git clone -b master https://github.com/raspberrypi/pico-examples.git
```

Building the example projects from the Command Line

1. From the Windows Menu, select Windows > Visual Studio 2019 > Developer Command Prompt
2. Navigate to the directory where you've installed the Pico examples and SDK. For me, it was C:\Users\vha3\Pico.
3. Set the path to the SDK by running the following:

```
C:\Users\vha3\Pico > setx PICO_SDK_PATH "..\..\pico-sdk"
```

4. Close the current command prompt window.
5. Open a new Command Prompt window by again navigating from the Windows Menu to Windows > Visual Studio 2019 > Developer Command Prompt. Closing/re-opening will set the environment variable that we configured above.
6. Navigate to the pico-examples folder. For me, this was in the directory C:\Users\vha3\Pico\pico-examples
7. Build the "Hello World" example by running:

```
C:\Users\vha3\Pico> cd pico-examples
C:\Users\vha3\Pico\pico-examples> mkdir build
C:\Users\vha3\Pico\pico-examples> cd build
C:\Users\vha3\Pico\pico-examples\build> cmake -G "NMake Makefiles" ..
C:\Users\vha3\Pico\pico-examples\build> nmake
```

8. Within the build directory, you will now find a hello_world directory. You will find directories for each of the other example projects too. These folders will contain the ELF, bin, and uf2 target files for each project. The uf2 target file can be dragged-and-dropped directly onto an RP2040 board attached to your PC via USB, as explained in the next section.

Hi there! I'm your AI assistant. Ask me anything or tell me what you're building. I'll help.

4. The Pico will automatically reboot, and start running the Blink example, flashing the LED.

Building the example projects from Visual Studio Code

1. Navigate via the Windows Menu to **Windows > Visual Studio 2019 > Developer Command Prompt**
2. When the command prompt window opens, type `C:\Users\vh33\Pico > code`. This will open Visual Studio Code with all of the correct environment variables set so that the toolchain is correctly configured.
3. Important reminder: When launching Visual Studio Code, avoid clicking its desktop icon or directly accessing it through the Start Menu. Instead, open it from a Developer Command Prompt Window where the necessary environment variables are already set. This will ensure a smooth and correct configuration of your build environment. To manually configure the environment later, you can adjust the CMake Tools Settings; however, opening VS Code from a Command Prompt Window is the simplest way to get started.
4. Click on the Extensions icon in the left-hand toolbar (or type `Ctrl + Shift + X`) and search for "CMake Tools"
5. Click the "CMake Tools" entry in the list, and then click the install button
6. Tweak your settings by clicking the spinning gear icon located at the bottom of the navigation panel on the left side of the interface.
7. In the Settings pane, click on "Extensions" and the "CMake Tools configuration."
8. Scroll down to "Cmake: Configure Environment" and click on "Add Item"
9. Set the Item to `PICO_SDK_PATH` and set the Value to `..\..\pico-sdk` as shown below



Cmake: Configure Environment

1. Scroll down to "Cmake: Generator" and enter "NMake Makefiles" into the box.
2. Close the Settings panel.
3. Navigate to **File > Open Folder** and navigate to the `pico-examples` repo, then hit "Okay."
4. You will be prompted to configure the project, as shown below. Click "Yes" and then select "GCC for arm-none-eabi" for your compiler.

Hi there! I'm your AI assistant. Ask me anything or tell me what you're building. I'll help.

1. To start building your projects, click the cog wheel-button labeled "Build" situated at the bottom of the window in the blue toolbar. This action will generate a build directory, trigger the CMake process, and then compile the example projects.
2. To load an example project onto the Pico, refer to the previous explanation and follow these steps: Within the build directory, select an example project folder and use the drag-and-drop function to transfer it to the Pico.

[← Previous Post](#)


[Next Post →](#)

About The Author

Muhammad Bilal

I am highly skilled and motivated individual with a Master's degree in [Computer Science](#). I have extensive experience in technical writing and a deep understanding of SEO practices.

Search



Raspberry PI Weekly Newsletter

Subscribe To A Specific Category

Get Notified Whenever There Is A New Project In Your Desired Category

Subscribe!

[Explore All Categories](#)

Useful Resources

[Blog](#)
[E-Books](#)
[News & Updates](#)
[Tutorials](#)
[Projects List](#)

Visit Us

Advance Search

[Advance Search](#)

Last Visited

[Automated Shrine with PyBot: Raspberry Pi Robot](#)
[Python](#)
[How To Design a Printed](#)
[part](#)
[led :](#)
[berr](#)
[2350: Df](#)
[n & Innc](#)

Hi there! I'm your AI assistant. Ask me anything or tell me what you're building. I'll help.



list
Sitemap



Hi there! I'm your AI assistant. Ask me anything or tell me what you're building, I'll help.

ACCEPT