Getting started with Pico on PC

Version: 2.0.5
Date: 11-Feb-2025

Important – these instructions are for a PC running Windows 10 or 11; they will not work for a Linux computer or for a Mac.

Contents

Introduction

As you step through this guide, please ensure that you check that your setup passes each of the checkpoints marked like this.

If you have already set up your computer, but you are encountering problems, please start by walking through each of the checkpoints.

Installing the software

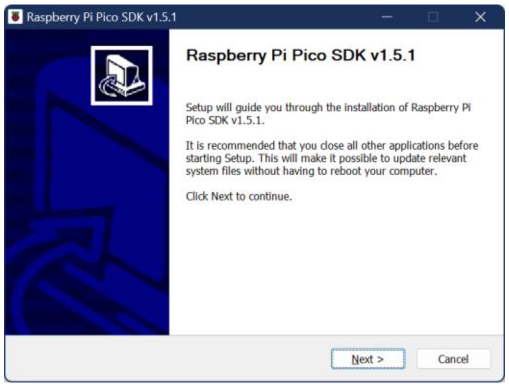
Download the installer

Go to https://github.com/raspberrypi/pico-setup-windows/releases/tag/v1.5.1 and download the installer, which is named **pico-setup-windows-x64-standalone.exe**.

Run the installer

Note - You must have administrator privileges on your PC to run the installation program.

Follow the default settings as you install the program:

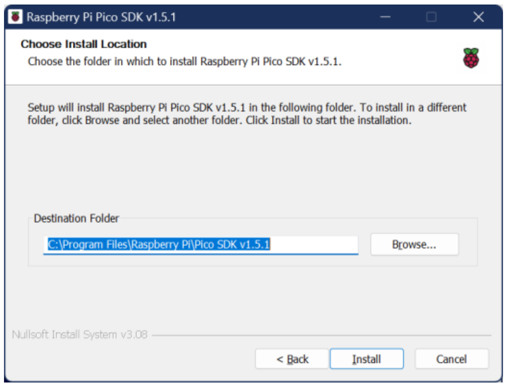


Raspberry Pi Pico SDK v1.5.1

Setup will guide you through the installation of Raspberry Pi Pico SDK v1.5.1.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

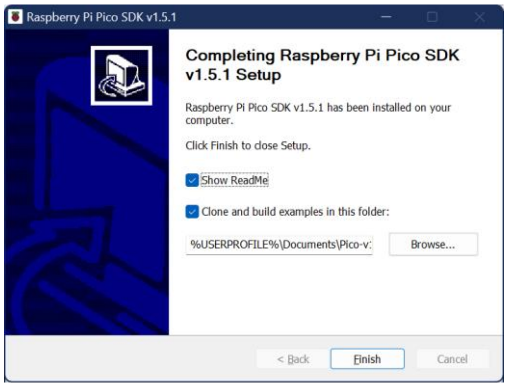


Choose Install Location

Choose the folder in which to install Raspberry Pi Pico SDK v1.5.1.

Setup will install Raspberry Pi Pico SDK v1.5.1 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files\Raspberry Pi\Pico SDK v1.5.1



Completing Raspberry Pi Pico SDK v1.5.1 Setup

Raspberry Pi Pico SDK v1.5.1 has been installed on your computer.

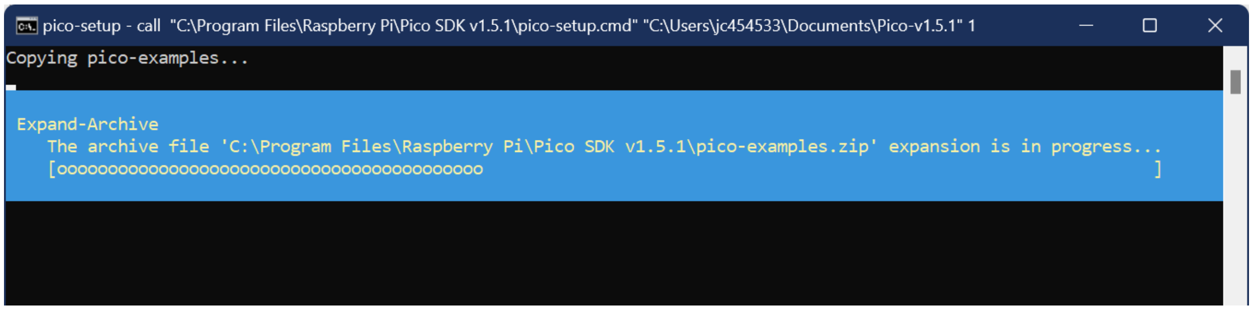
Click Finish to close Setup.

☑ Show ReadMe

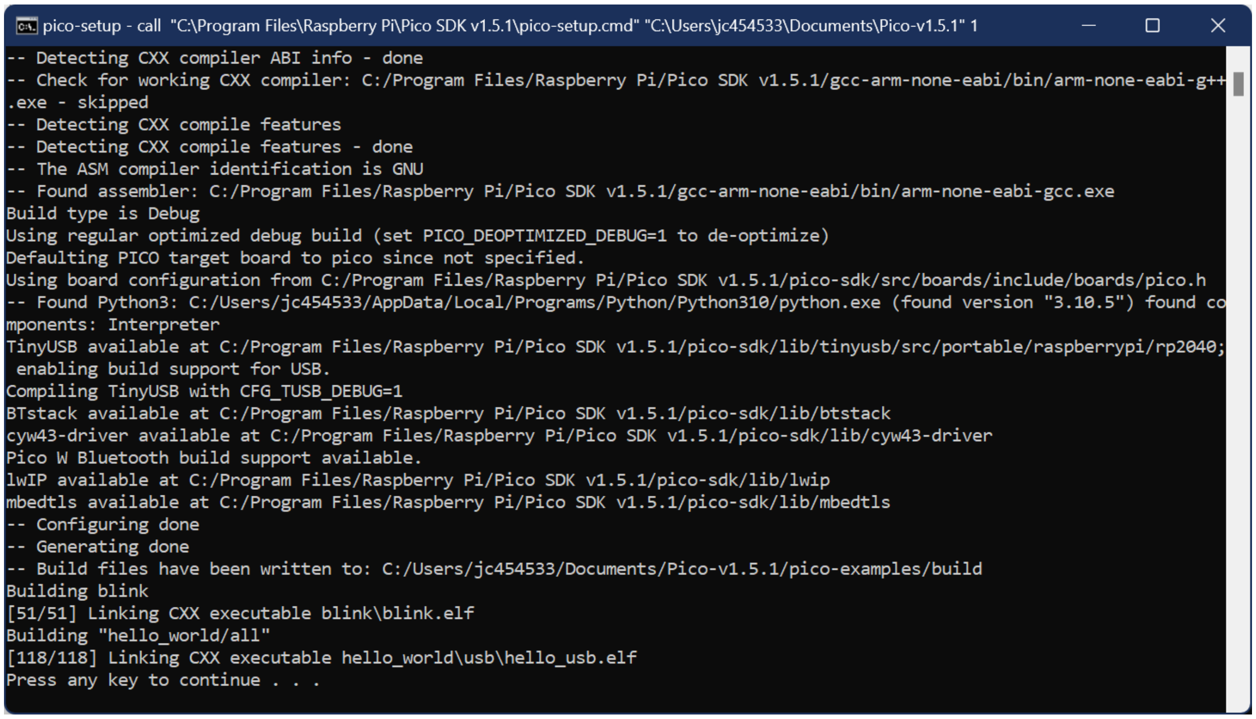
☑ Clone and build examples in this folder:

%USERPROFILE%\Documents\Pico-v:

When the installation is complete, the examples will be copied and built:



```
pico-setup - call  "C:\Program Files\Raspberry Pi\Pico SDK v1.5.1\pico-setup.cmd" "C:\Users\jc454533\Documents\Pico-v1.5.1" 1
Copying pico-examples...

Expand-Archive
   The archive file 'C:\Program Files\Raspberry Pi\Pico SDK v1.5.1\pico-examples.zip' expansion is in progress...
   [oooooooooooooooooooooooooooooooooooooooooo                                                    ]
```


pico-setup - call "C:\Program Files\Raspberry Pi\Pico SDK v1.5.1\pico-setup.cmd" "C:\Users\jc454533\Documents\Pico-v1.5.1" 1

```
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/gcc-arm-none-eabi/bin/arm-none-eabi-g++
.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- The ASM compiler identification is GNU
-- Found assembler: C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/gcc-arm-none-eabi/bin/arm-none-eabi-gcc.exe
Build type is Debug
Using regular optimized debug build (set PICO_DEOPTIMIZED_DEBUG=1 to de-optimize)
Defaulting PICO target board to pico since not specified.
Using board configuration from C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/src/boards/include/boards/pico.h
-- Found Python3: C:/Users/jc454533/AppData/Local/Programs/Python/Python310/python.exe (found version "3.10.5") found co
mponents: Interpreter
TinyUSB available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/tinyusb/src/portable/raspberrypi/rp2040;
 enabling build support for USB.
Compiling TinyUSB with CFG_TUSB_DEBUG=1
BTstack available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/btstack
cyw43-driver available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/cyw43-driver
Pico W Bluetooth build support available.
lwIP available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/lwip
mbedtls available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/mbedtls
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/jc454533/Documents/Pico-v1.5.1/pico-examples/build
Building blink
[51/51] Linking CXX executable blink\blink.elf
Building "hello_world/all"
[118/118] Linking CXX executable hello_world\usb\hello_usb.elf
Press any key to continue . . .
```

Checkpoint 1

Did the examples build process run and complete successfully? Do the last five lines resemble the lines in the picture above?

The Windows Command Prompt

<u>Important</u>: if you have not used the Windows command line environment before, read this section.

From time to time in this course, you will need to work with the Windows Command Prompt.

The Windows Command Line, often referred to as the Command Prompt or simply "`cmd`," is a text-based interface used to interact with the operating system. It allows users to execute commands, navigate through directories, and manage files directly from the keyboard, providing a more powerful and efficient way to perform tasks compared to the graphical interface.

When you open the Command Prompt, you're greeted with a prompt that typically displays the current directory path, followed by a greater-than symbol (>), and then a blinking cursor. For example:

```
C:\Users\YourName>
```

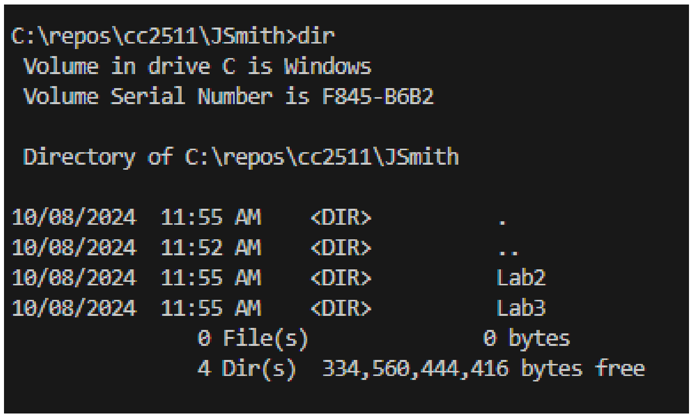
This prompt indicates that you are in the `C:\Users\YourName` directory and ready to enter commands.

Commands

Two fundamental commands in the Windows Command Line are `dir` and `cd`:

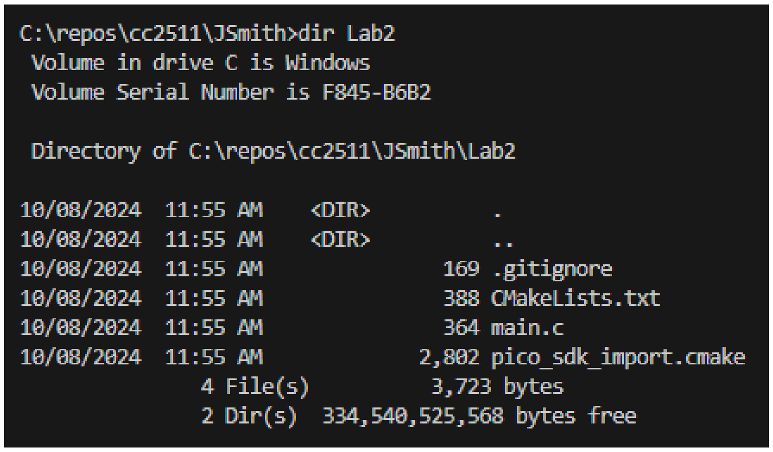
dir (Directory Listing)

The `dir` command is used to list the contents of the current directory. When you type `dir` and press Enter, you'll see a list of all files and subdirectories within the current directory, along with details such as file sizes and dates.



```
C:\repos\cc2511\JSmith>dir
 Volume in drive C is Windows
 Volume Serial Number is F845-B6B2

 Directory of C:\repos\cc2511\JSmith

10/08/2024  11:55 AM    <DIR>          .
10/08/2024  11:52 AM    <DIR>          ..
10/08/2024  11:55 AM    <DIR>          Lab2
10/08/2024  11:55 AM    <DIR>          Lab3
               0 File(s)              0 bytes
               4 Dir(s)  334,560,444,416 bytes free
```

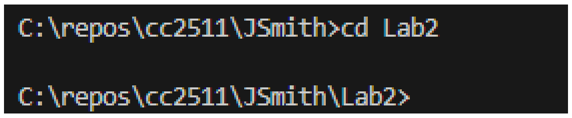
If you type a name after the command, you will see a listing of the named sub-folder:



```
C:\repos\cc2511\JSmith>dir Lab2
 Volume in drive C is Windows
 Volume Serial Number is F845-B6B2

 Directory of C:\repos\cc2511\JSmith\Lab2

10/08/2024  11:55 AM    <DIR>          .
10/08/2024  11:55 AM    <DIR>          ..
10/08/2024  11:55 AM               169 .gitignore
10/08/2024  11:55 AM               388 CMakeLists.txt
10/08/2024  11:55 AM               364 main.c
10/08/2024  11:55 AM             2,802 pico_sdk_import.cmake
               4 File(s)          3,723 bytes
               2 Dir(s)  334,540,525,568 bytes free
```

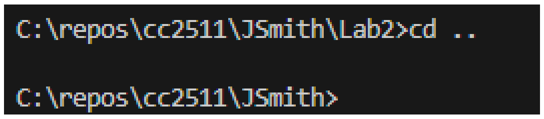
cd (Change Directory)

The cd command is used to change the current directory. To navigate to a different folder, type cd followed by the path of the directory you want to enter. If the directory is within the current one, you can simply type its name. e.g.:



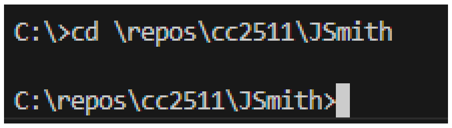
```
C:\repos\cc2511\JSmith>cd Lab2

C:\repos\cc2511\JSmith\Lab2>
```

To move up one level, use: `cd ..`



```
C:\repos\cc2511\JSmith\Lab2>cd ..

C:\repos\cc2511\JSmith>
```

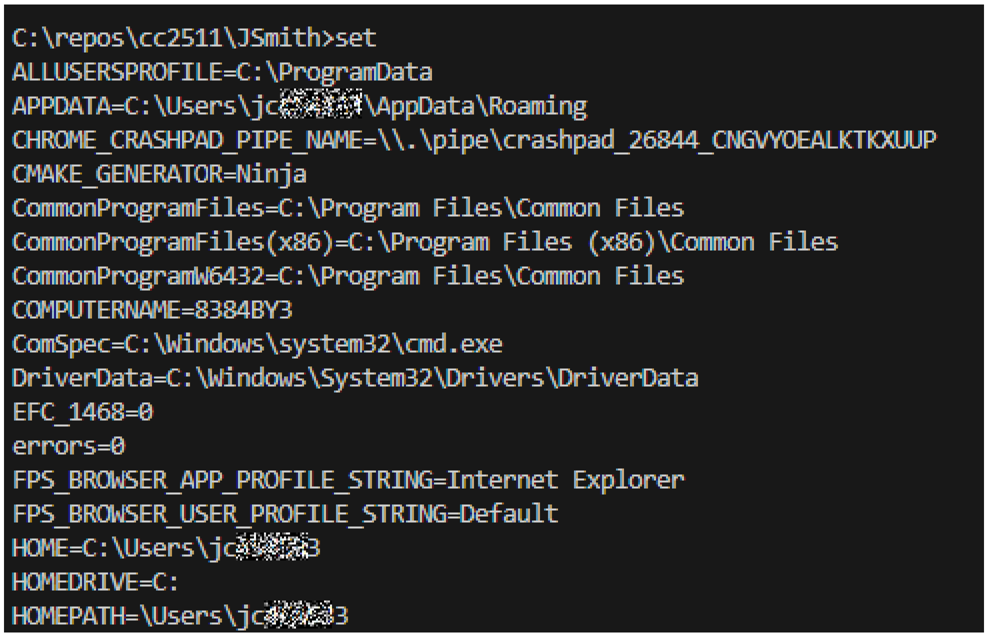
To navigate to a specific directory on the current disk, type the full path, starting with a back-slash, e.g:



```
C:\>cd \repos\cc2511\JSmith

C:\repos\cc2511\JSmith>
```

Environment variables

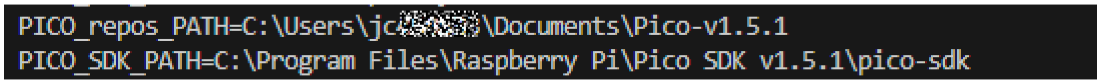
A command environment can include a number of variables, which are aliases for other values – often directory paths. You can see a list of all the active environment variables by typing `set` (followed by Enter).



```
C:\repos\cc2511\JSmith>set
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\jc[redacted]\AppData\Roaming
CHROME_CRASHPAD_PIPE_NAME=\\.\pipe\crashpad_26844_CNGVYOEALKTKXUUP
CMAKE_GENERATOR=Ninja
CommonProgramFiles=C:\Program Files\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=8384BY3
ComSpec=C:\Windows\system32\cmd.exe
DriverData=C:\Windows\System32\Drivers\DriverData
EFC_1468=0
errors=0
FPS_BROWSER_APP_PROFILE_STRING=Internet Explorer
FPS_BROWSER_USER_PROFILE_STRING=Default
HOME=C:\Users\jc[redacted]3
HOMEDRIVE=C:
HOMEPATH=\Users\jc[redacted]3
```

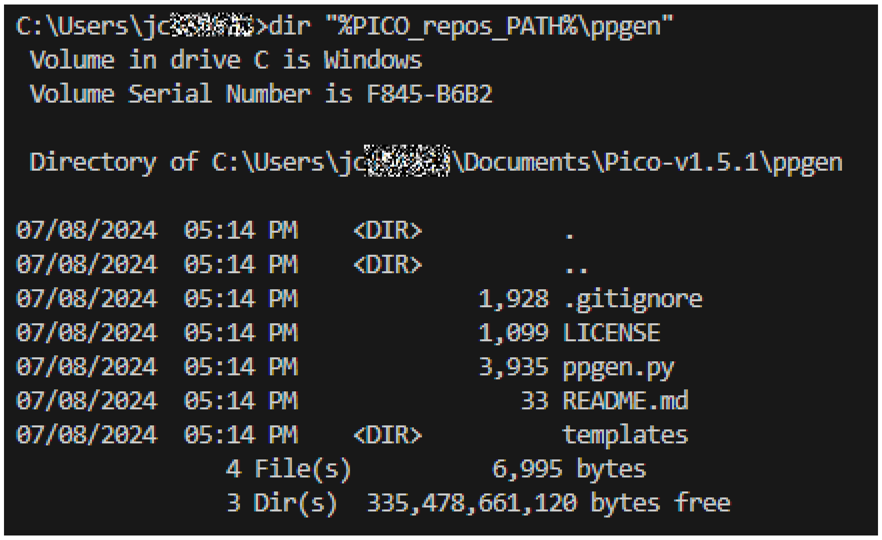
The variables are listed in alphabetical order. Scroll up and down to find the variable you are interested in.

In tis course we will be using the variables below extensively:



```
PICO_repos_PATH=C:\Users\jc[redacted]\Documents\Pico-v1.5.1
PICO_SDK_PATH=C:\Program Files\Raspberry Pi\Pico SDK v1.5.1\pico-sdk
```

You can substitute the value of an environment variable into a command by enclosing it in percent symbols, e.g.:



```
C:\Users\jc[redacted]>dir "%PICO_repos_PATH%\ppgen"
 Volume in drive C is Windows
 Volume Serial Number is F845-B6B2

 Directory of C:\Users\jc[redacted]\Documents\Pico-v1.5.1\ppgen

07/08/2024  05:14 PM    <DIR>          .
07/08/2024  05:14 PM    <DIR>          ..
07/08/2024  05:14 PM             1,928 .gitignore
07/08/2024  05:14 PM             1,099 LICENSE
07/08/2024  05:14 PM             3,935 ppgen.py
07/08/2024  05:14 PM                33 README.md
07/08/2024  05:14 PM    <DIR>          templates
               4 File(s)          6,995 bytes
               3 Dir(s)  335,478,661,120 bytes free
```

Keyboard shortcuts

TAB

Use the TAB key to complete the name of a folder or file in the current command. If there are multiple files that match the current input, they will be listed in alphabetical order: hit TAB once for the first match, press it again for the second, and so on.

Using the TAB key has 2 major advantages:

1) It saves on typing.
2) It reduces errors.

Up and Down cursors

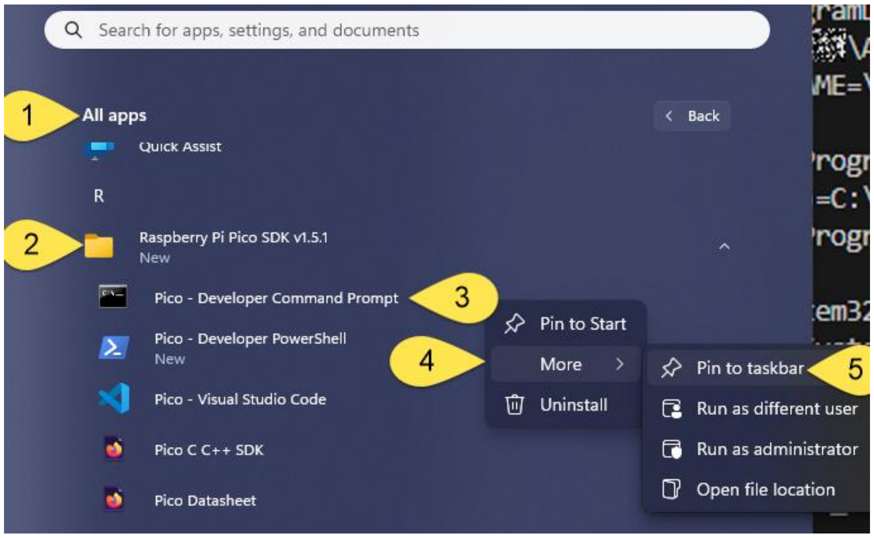
Use the UP cursor key to display the previous command entered at this command line. Keep pressing UP to scroll through a history of the commands. You can use DOWN to scroll forward through the list.

This saves time and mistakes during repetitious command entries.

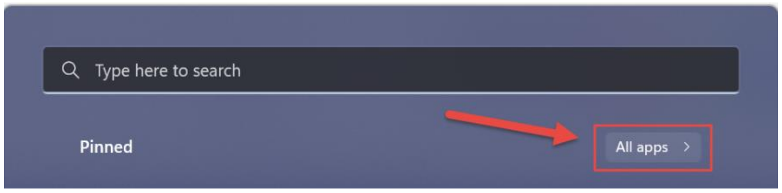
Opening the Pico - Developer Command Prompt

We will use a specific variant of the Windows command prompt in this course, which has the requires environment variables set up automatically. This is listed under **All apps**, in the **Raspberry Pi Pico SDK v1.5.1** folder, as **Pico - Developer Command Prompt**.

Save time by pinning it to the task bar, so that you won't need to find it every time you need it.



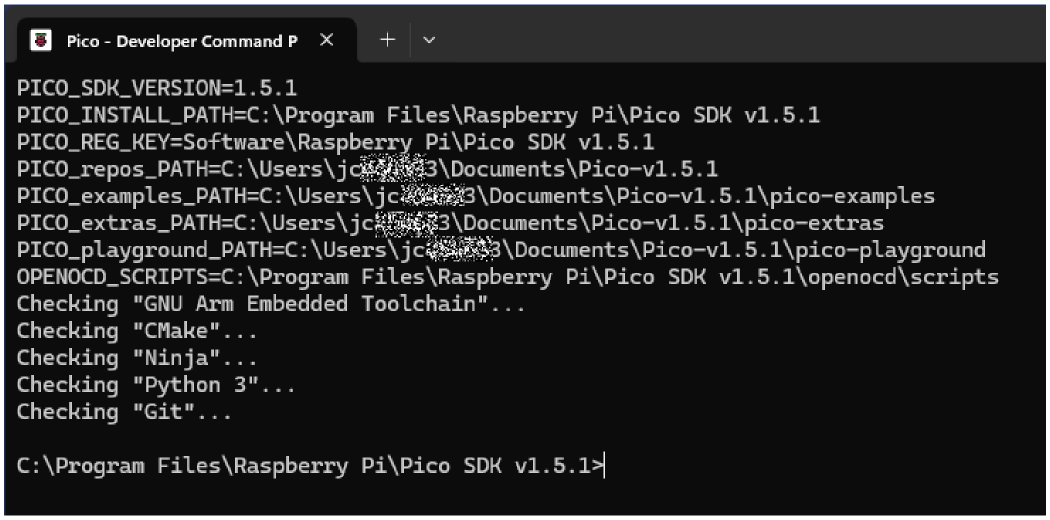
1. Press the Windows key (🪟) and click ***All apps***:



   Scroll down to the ***Raspberry Pi Pico SDK v1.5.1*** folder and click on it to open it.
2. Right-click on the ***Pico – Developer Command Prompt*** item.
3. Click ***More >***.
4. Then click ***Pin to taskbar***.

Checkpoint 2

1. Find the ***Pico – Developer Command Prompt*** icon on the task bar and click it. You should see a window resembling the following:



```
PICO_SDK_VERSION=1.5.1
PICO_INSTALL_PATH=C:\Program Files\Raspberry Pi\Pico SDK v1.5.1
PICO_REG_KEY=Software\Raspberry Pi\Pico SDK v1.5.1
PICO_repos_PATH=C:\Users\jc[redacted]3\Documents\Pico-v1.5.1
PICO_examples_PATH=C:\Users\jc[redacted]3\Documents\Pico-v1.5.1\pico-examples
PICO_extras_PATH=C:\Users\jc[redacted]3\Documents\Pico-v1.5.1\pico-extras
PICO_playground_PATH=C:\Users\jc[redacted]3\Documents\Pico-v1.5.1\pico-playground
OPENOCD_SCRIPTS=C:\Program Files\Raspberry Pi\Pico SDK v1.5.1\openocd\scripts
Checking "GNU Arm Embedded Toolchain"...
Checking "CMake"...
Checking "Ninja"...
Checking "Python 3"...
Checking "Git"...

C:\Program Files\Raspberry Pi\Pico SDK v1.5.1>
```

2. Note that the title bar contains “***Pico – Developer Command Prompt***”.
3. Note that a set of environment variables, including `PICO_repos_PATH`, are created at the start of the environment.

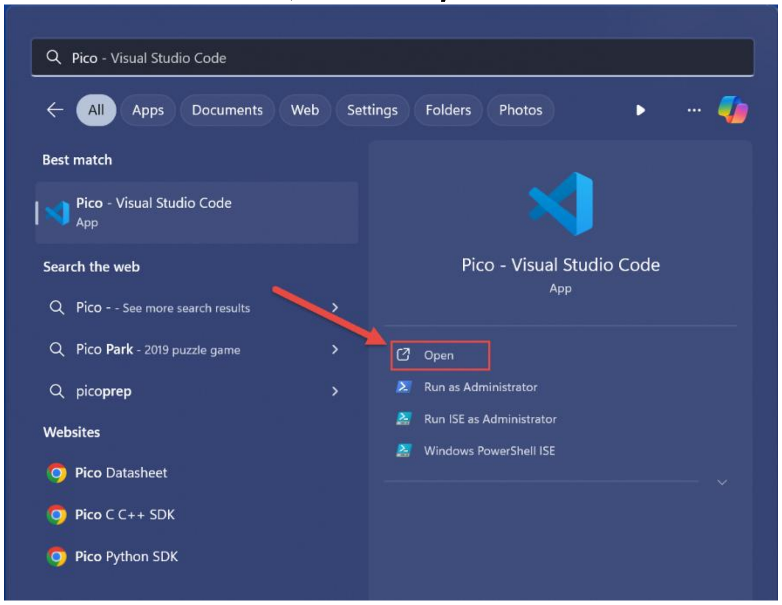
Starting Visual Studio Code

In your Start Menu, look for the "Pico – Visual Studio Code" shortcut, in the "Raspberry Pi" folder. The shortcut sets the required environment variables and then launches Visual Studio Code.

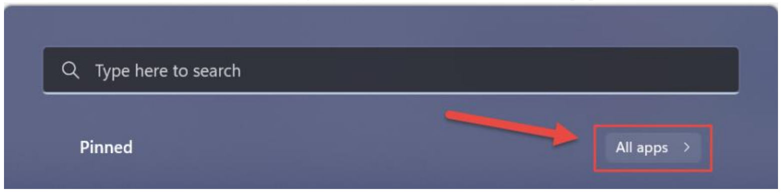
> ***Note – you may already have a copy of VS Code on your computer. Do not start the existing application: make sure that you use the shortcut "Pico – Visual Studio Code" as described above.***

On Windows 11, there are two ways you can start VS Code:

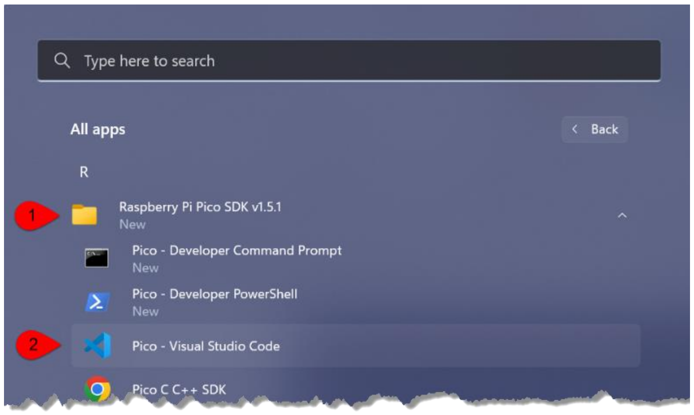
1) Press the Windows key (🪟) and type ***Pico –*** in the search bar. The application shown below will be offered, click on ***Open***:



2) Press the Windows key (🪟) and click ***All apps***:

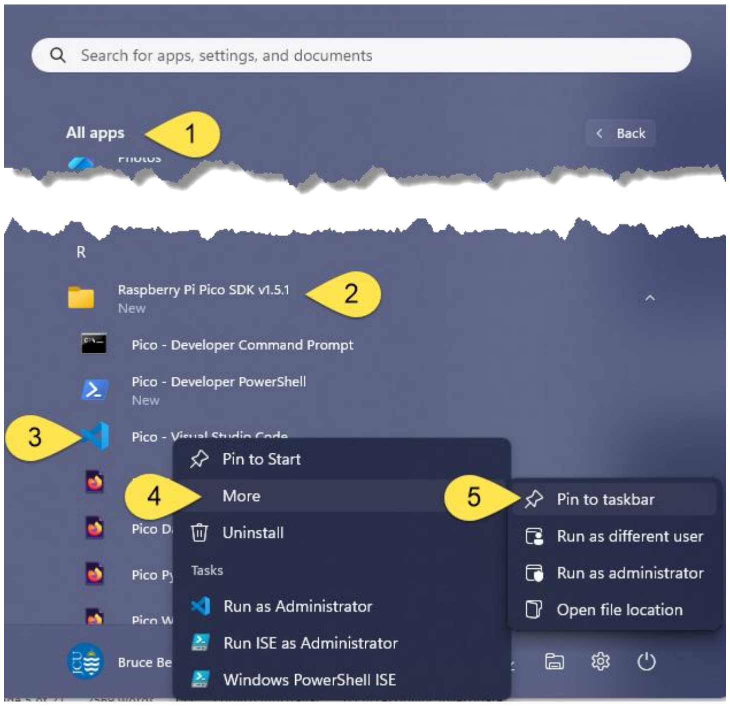


Then scroll down until you can see ***Raspberry Pi Pico SDK v1.5.1***. Click on this to expand it, and click on ***Pico – Visual Studio Code***:


Type here to search
All apps
Back
R
1
Raspberry Pi Pico SDK v1.5.1
New
Pico - Developer Command Prompt
New
Pico - Developer PowerShell
New
2
Pico - Visual Studio Code
Pico C C++ SDK

Making it easier

Tip: Before you click on the Pico – Visual Studio Code menu item, right-click on the icon and then select ***More > Pin to taskbar***.



Now you will be able to start the application from the task bar whenever you need it.

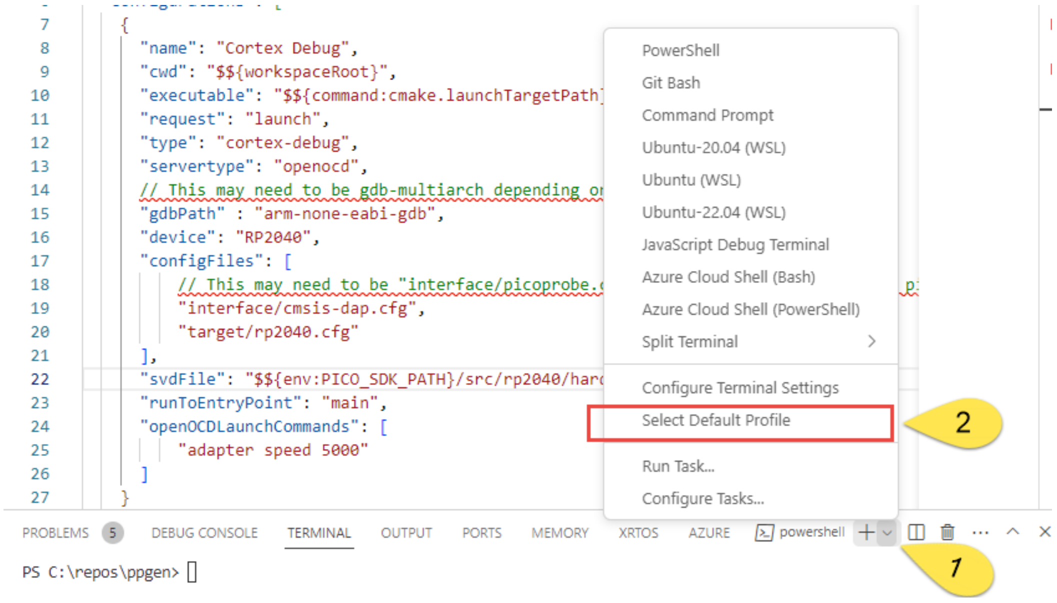
Checkpoint 3

1. Is there an icon on the Task Bar that looks like this?
2. If you float the mouse over the icon, does it show the text Pico – Visual Studio Code?
3. If you click on it, does Visual Studio Code start up?

Selecting the default VS Code Terminal

Once you have started VS Code, change the default Terminal window to be the CMD terminal instead of Powershell or Git Bash. You can set this terminal as the default with the following steps:

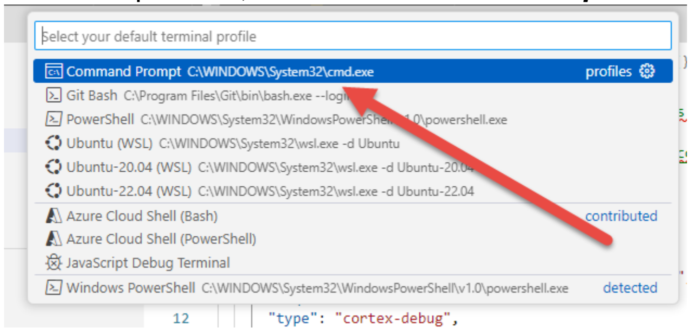
1) Start Pico - Visual Studio Code (as described above).
2) In VS Code use the menu command ***Terminal > New Terminal***. Note that this opens a PowerShell window, which is probably not what you want.
3) Click the down-arrow next to the + on the Terminal window toolbar.



```
{
    "name": "Cortex Debug",
    "cwd": "$${workspaceRoot}",
    "executable": "$${command:cmake.launchTargetPath
    "request": "launch",
    "type": "cortex-debug",
    "servertype": "openocd",
    // This may need to be gdb-multiarch depending o
    "gdbPath" : "arm-none-eabi-gdb",
    "device": "RP2040",
    "configFiles": [
        // This may need to be "interface/picoprobe.
        "interface/cmsis-dap.cfg",
        "target/rp2040.cfg"
    ],
    "svdFile": "$${env:PICO_SDK_PATH}/src/rp2040/hard
    "runToEntryPoint": "main",
    "openOCDLaunchCommands": [
        "adapter speed 5000"
    ]
}
```

```
PS C:\repos\ppgen>
```

4) Click ***Select Default Profile***.
5) In the drop-down, select ***Command Prompt***.



Select your default terminal profile
Command Prompt C:\WINDOWS\System32\cmd.exe profiles
Git Bash C:\Program Files\Git\bin\bash.exe --logi
PowerShell C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe
Ubuntu (WSL) C:\WINDOWS\System32\wsl.exe -d Ubuntu
Ubuntu-20.04 (WSL) C:\WINDOWS\System32\wsl.exe -d Ubuntu-20.04
Ubuntu-22.04 (WSL) C:\WINDOWS\System32\wsl.exe -d Ubuntu-22.04
Azure Cloud Shell (Bash) contributed
Azure Cloud Shell (PowerShell)
JavaScript Debug Terminal
Windows PowerShell C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe detected

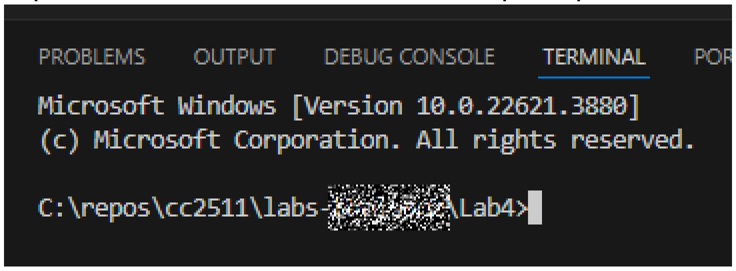
```
"type": "cortex-debug",
```

6) You can close the PowerShell window by clicking the trash icon on the Terminal window toolbar.
7) Next time you open a terminal window it will be a standard Command window.

Checkpoint 4

1. If VS Code is not open, start it, using the ***Pico – Visual Studio Code*** icon on the Taskbar.
2. Use the View > Terminal menu (or Ctrl+ `) to open a new Terminal window. Does it look something like the picture below, with a Windows version in the

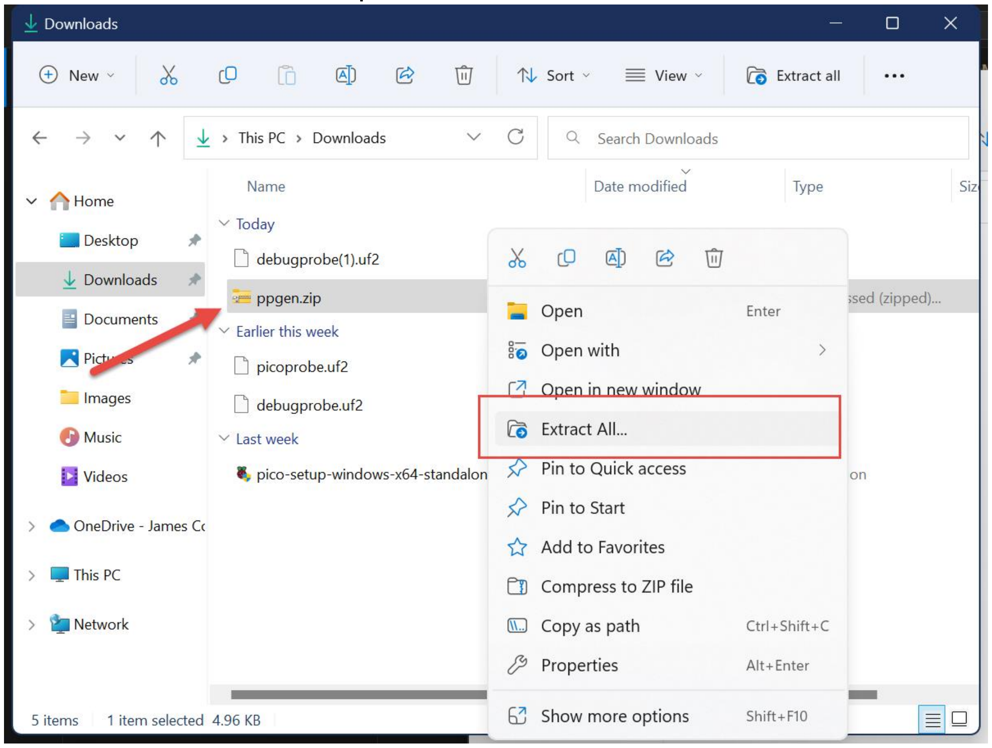
top line and no ***PS*** at the start of the prompt?



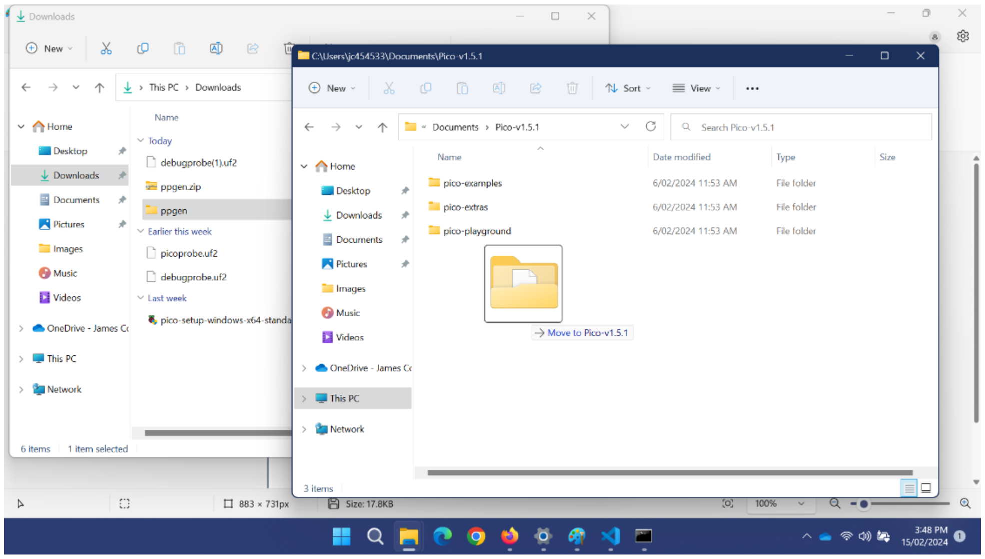
```
Microsoft Windows [Version 10.0.22621.3880]
(c) Microsoft Corporation. All rights reserved.

C:\repos\cc2511\labs-[redacted]\Lab4>
```

Install the project generator

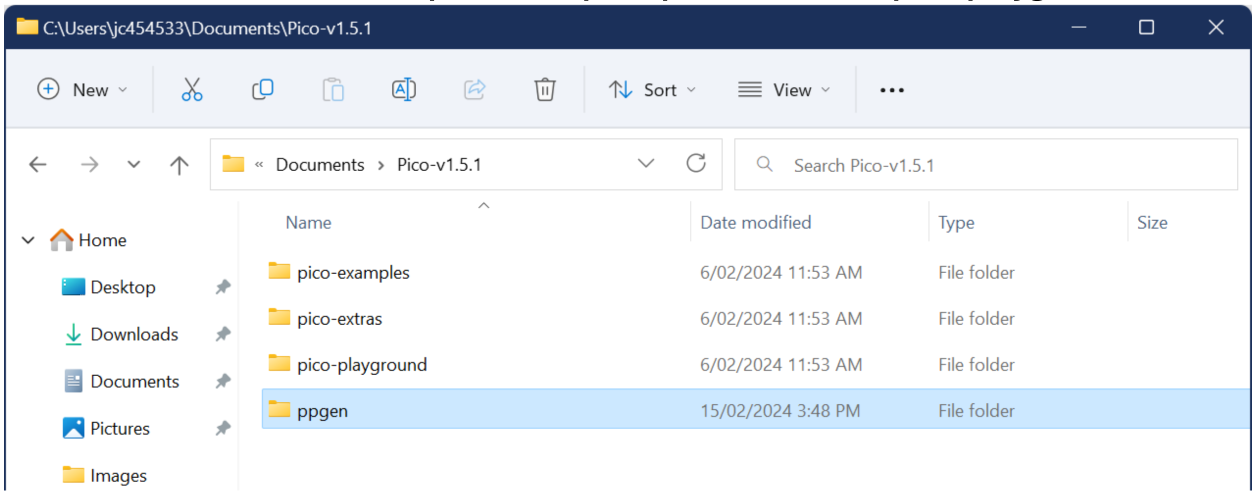
1) Download the project generator zip file – ***ppgen.zip*** - from ***LearnJCU > Computer Setup > Pico Development***.
2) Extract all files from the zip file.



3) The expanded files will be saved to a folder named ***ppgen***. Use Windows Explorer to move this folder to ***Documents\Pico-v1.5.1***.


Downloads
C:\Users\jc454533\Documents\Pico-v1.5.1
Documents > Pico-v1.5.1
Search Pico-v1.5.1
pico-examples 6/02/2024 11:53 AM File folder
pico-extras 6/02/2024 11:53 AM File folder
pico-playground 6/02/2024 11:53 AM File folder
Move to Pico-v1.5.1
debugprobe(1).uf2
ppgen.zip
ppgen
picoprobe.uf2
debugprobe.uf2
pico-setup-windows-x64-standa
6 items 1 item selected
3 items

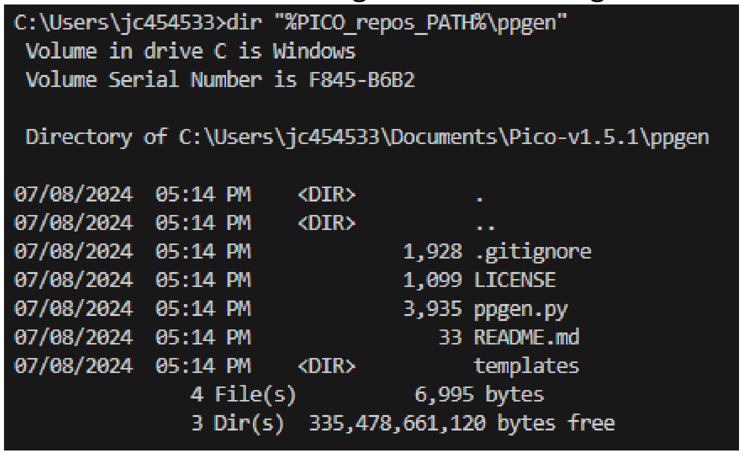
4) It will be next to folders named ***pico-examples***, ***pico-extras*** and ***pico-playground***.



Name	Date modified	Type	Size
pico-examples	6/02/2024 11:53 AM	File folder	
pico-extras	6/02/2024 11:53 AM	File folder	
pico-playground	6/02/2024 11:53 AM	File folder	
ppgen	15/02/2024 3:48 PM	File folder	

Checkpoint 5

1. If VS Code is not open, start it, using the ***Pico – Visual Studio Code*** icon on the Taskbar.
2. Use the View > Terminal menu (or Ctrl+`) to open a new Terminal window.
3. Type (or copy) the command below and press enter:
   `dir "%PICO_repos_PATH%\ppgen"`
4. You should see something like the following:



```
C:\Users\jc454533>dir "%PICO_repos_PATH%\ppgen"
 Volume in drive C is Windows
 Volume Serial Number is F845-B6B2

 Directory of C:\Users\jc454533\Documents\Pico-v1.5.1\ppgen

07/08/2024  05:14 PM    <DIR>          .
07/08/2024  05:14 PM    <DIR>          ..
07/08/2024  05:14 PM             1,928 .gitignore
07/08/2024  05:14 PM             1,099 LICENSE
07/08/2024  05:14 PM             3,935 ppgen.py
07/08/2024  05:14 PM                33 README.md
07/08/2024  05:14 PM    <DIR>          templates
               4 File(s)          6,995 bytes
               3 Dir(s)  335,478,661,120 bytes free
```

It may not be identical, but it will list (among other items) a file named `ppgen.py` and a folder named `templates`.

Setting up your work area

You need to select a folder on your drive where you will store your coding projects for this course. You will of course want to be sure that they are backed up.

Often, a good practise is to store data you wish to be backed up in your Documents folder or under OneDrive. However, you will be using GitHub to back up all the important parts your work in this course, so selecting a folder which is entirely backed to the could is wasteful and can slow you down.

As you will see later, creating and building VS Code projects causes many temporary files to be created. The file sizes of a typical new project are shown below:

Category	Number of files	Size of files
Source files	4	4 kb
Temporary files	255	3.2 MB

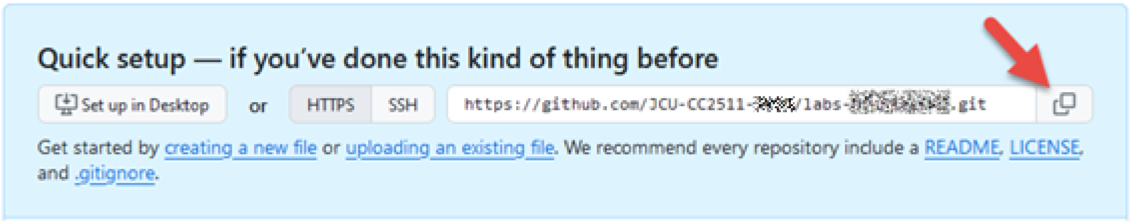
It is therefore important not to waste time and energy saving these temporary files. A popular choice is to have a folder called repos in your C: drive's root folder, e.g. C:\repos.

You can do this in Windows Explorer (also known as File Manager). Alternatively, use the ***Pico – Developer Command Prompt*** as follows:

```
C:
cd \
mkdir repos
cd repos
```

Now create a local copy – a clone – of your CC2511 GitHub repo.

1. If you have not already done so, click on the link in ***LearnJCU* > *Computer setup* > *Set up your CC2511 GitHub repository***.
2. Open your repo page in a browser. If you have not already added anything to your repo, you will see a page like the following. Click on the Copy button.

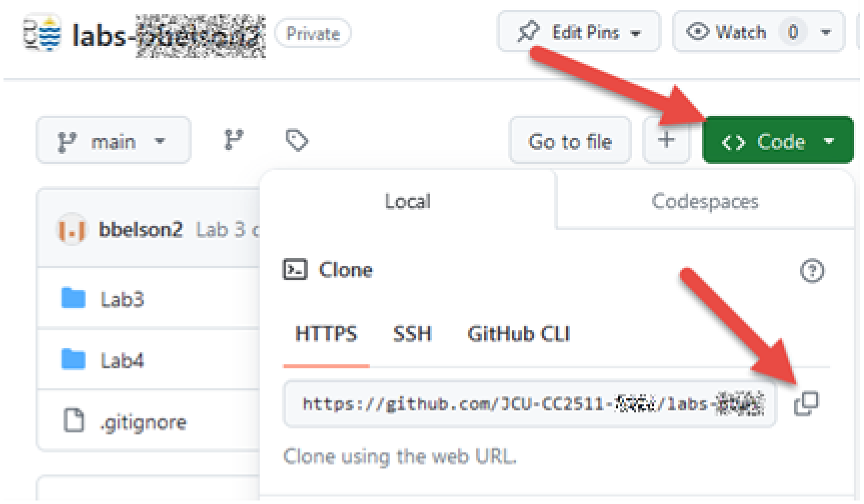


Quick setup — if you've done this kind of thing before

https://github.com/JCU-CC2511-[redacted]/labs-[redacted].git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

3. If you have already added content to your repo, you will see a page as below. Click on Code to open the drop-down, then click on the Copy button.



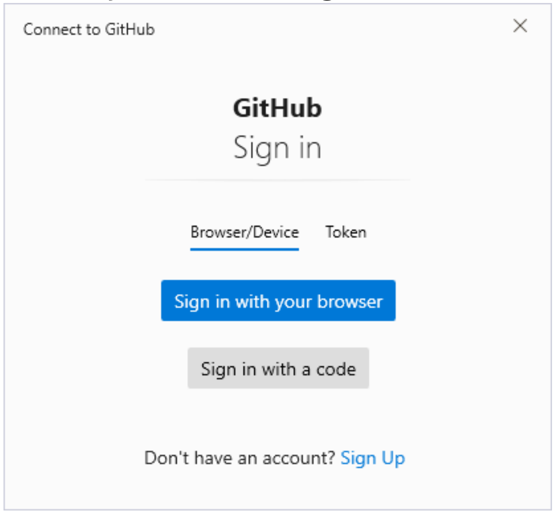
4. Whichever way you did it, you now have the URL of your repo in the clipboard. Return to the Pico – Developer Command Prompt (or open a new one) and navigate to your source folder, e.g.:

```
C:
cd \repos
```

5. Make a clone of your repo by typing git<space>clone<space> and then Ctrl+V to paste in the URL, e.g.:

```
git clone https://github.com/JCU-CC2511-2025/labs-jc123456.git
```

6. You may be asked to log in to GitHub using your Browser, e.g.:



7. If all succeeds, you now have a local copy of your repo in e.g. C:\repos\labs-jc1234546

Creating a new project

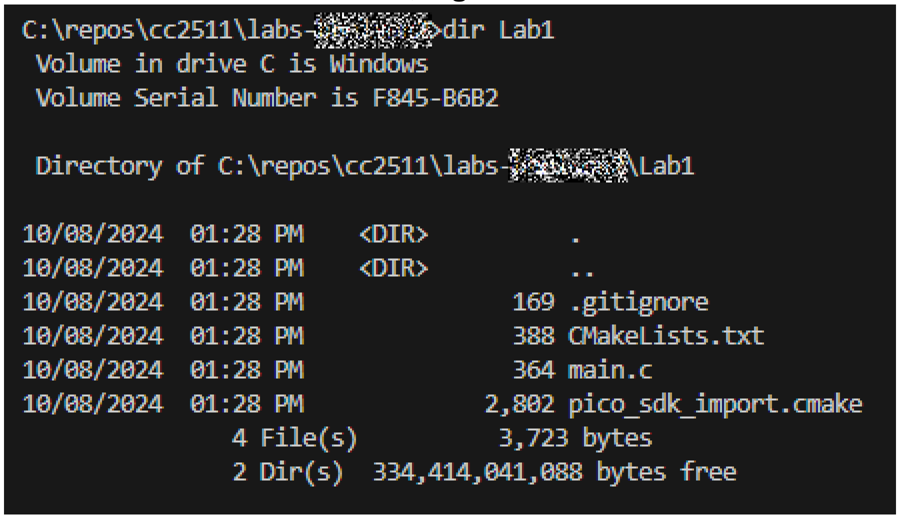
Create a project

1) It is important that you create your projects in the correct location. See ***Setting up your work area*** above. If you are in doubt, talk to your lecturer or tutor about which folder you should be creating your projects in.
2) In VS Code, open the terminal window using the menu command: ***Terminal > New Terminal***.
3) Navigate to your repo folder. For example, if you are storing projects in C:\repos\labs-jc123456, then type:
   ```
   cd \repos\labs-jc123456
   ```
   and press Enter.
4) Now run the Python script ***ppgen.py*** and specify the new project name. For example, to create a project named Lab1, type the following:
   ```
   python %pico_repos_path%\ppgen\ppgen.py Lab1
   ```
5) If the script runs successfully, you will see a report on the Terminal, similar to this:
   ```
   ppgen generated 4 files to C:\repos\labs-[your name]\Lab1
   ```

   Tip: You can also specify other command line arguments. Type the following to get a full list:
   ```
   python %pico_repos_path%\ppgen\ppgen.py -help
   ```

Checkpoint 6

1. Check that the project creation succeeded by typing as follows at the command line:
   ```
   dir Lab1
   ```
2. You should see at least 4 files, e.g.:



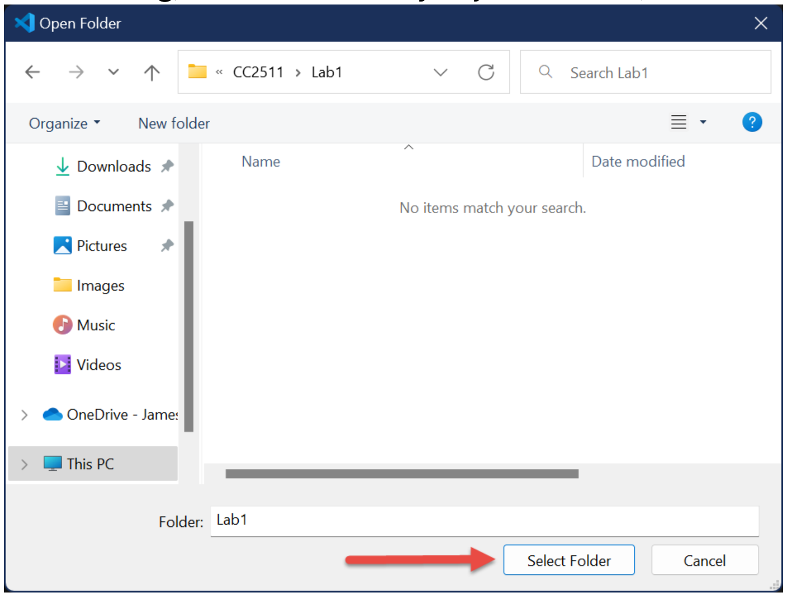
```
C:\repos\cc2511\labs-[redacted]>dir Lab1
 Volume in drive C is Windows
 Volume Serial Number is F845-B6B2

 Directory of C:\repos\cc2511\labs-[redacted]\Lab1

10/08/2024  01:28 PM    <DIR>          .
10/08/2024  01:28 PM    <DIR>          ..
10/08/2024  01:28 PM               169 .gitignore
10/08/2024  01:28 PM               388 CMakeLists.txt
10/08/2024  01:28 PM               364 main.c
10/08/2024  01:28 PM             2,802 pico_sdk_import.cmake
               4 File(s)          3,723 bytes
               2 Dir(s)  334,414,041,088 bytes free
```

3. If there is no file named `.gitignore`, make sure that you have downloaded the latest version of ppgen.zip from LearnJCU. If necessary, repeat the section ***Install the project generator*** above.

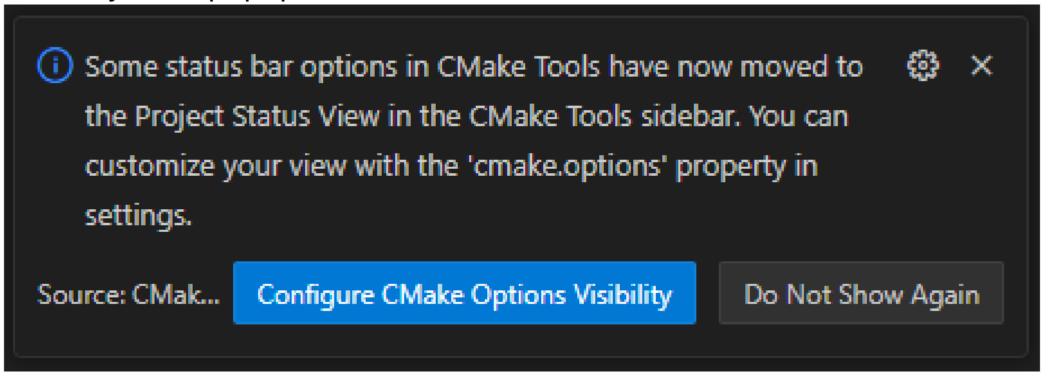
Open the project

Now that the project has been created, open it in VS Code.

1) Use ***File > Open Folder…*** (or Ctrl+K Ctrl+O) to show the Open Folder dialog.
2) In the dialog, select the folder you just created, and click Select Folder:



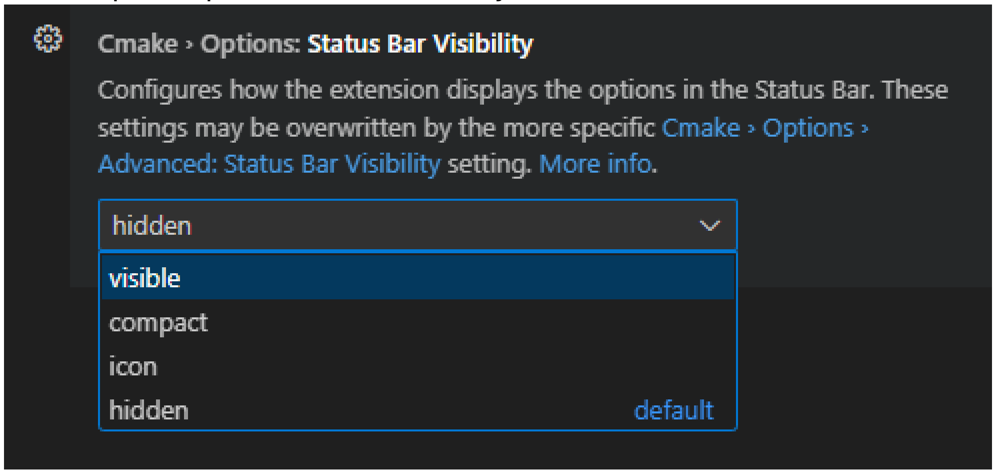
3) You may be asked “Do you trust the authors of the files in this folder?”. Answer “Yes, I trust the authors”.
4) You may see a popup as follows:



Some status bar options in CMake Tools have now moved to the Project Status View in the CMake Tools sidebar. You can customize your view with the 'cmake.options' property in settings.

5) Click ***Configure CMake Options Visibility***.

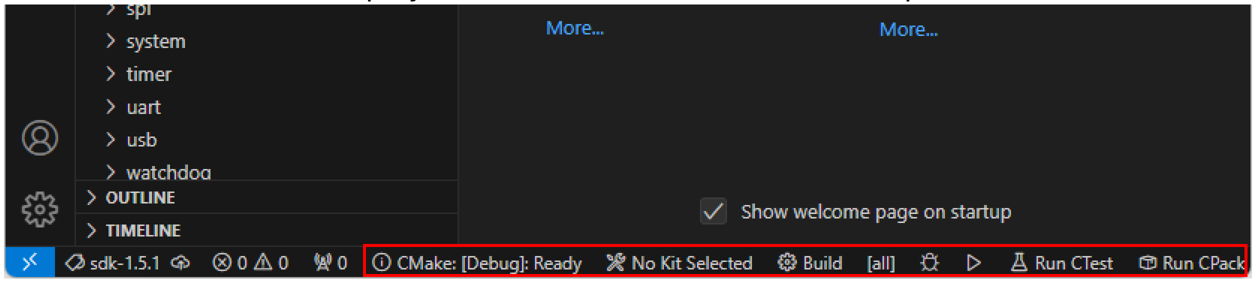
6) In the Options panel, set the visibility to Visible:



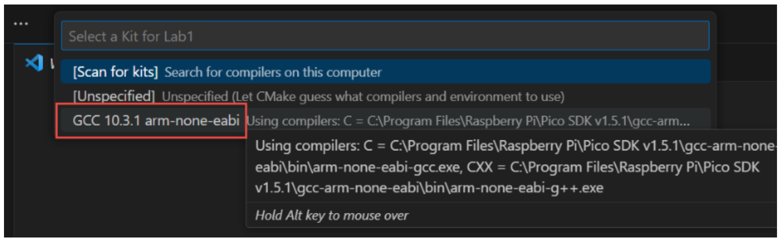
Cmake › Options: Status Bar Visibility

Configures how the extension displays the options in the Status Bar. These settings may be overwritten by the more specific Cmake › Options › Advanced: Status Bar Visibility setting. More info.

7) The status bar will now display useful information about the build process.



8) Next time this popup appears, you can click ***Do Not Show Again***.
9) You may see a popup asking “Would you like to configure project Lab1?”. Answer “Yes”. In the drop-down which asks “Select a kit for Lab1”, select the kit that begins with “Pico” or “GCC”:



Select a Kit for Lab1

[Scan for kits] Search for compilers on this computer

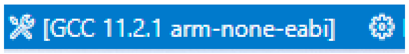
[Unspecified] Unspecified (Let CMake guess what compilers and environment to use)

GCC 10.3.1 arm-none-eabi Using compilers: C = C:\Program Files\Raspberry Pi\Pico SDK v1.5.1\gcc-arm...

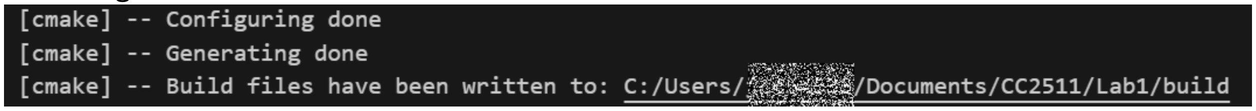
Using compilers: C = C:\Program Files\Raspberry Pi\Pico SDK v1.5.1\gcc-arm-none-eabi\bin\arm-none-eabi-gcc.exe, CXX = C:\Program Files\Raspberry Pi\Pico SDK v1.5.1\gcc-arm-none-eabi\bin\arm-none-eabi-g++.exe

Hold Alt key to mouse over

10) If neither appears, then click on “[Scan for kits]”, then select it when it does appear. If it does not appear at all, check that you started the VS Code instance from the ***Pico – Visual Studio Code*** icon. If this does not fix the problem, talk to your tutor.
11) The kit on the status bar will be updated, e.g.:



12) You will see a lot of output in the ***Output*** window of VS Code, ending with the following:



```
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: C:/Users/[redacted]/Documents/CC2511/Lab1/build
```

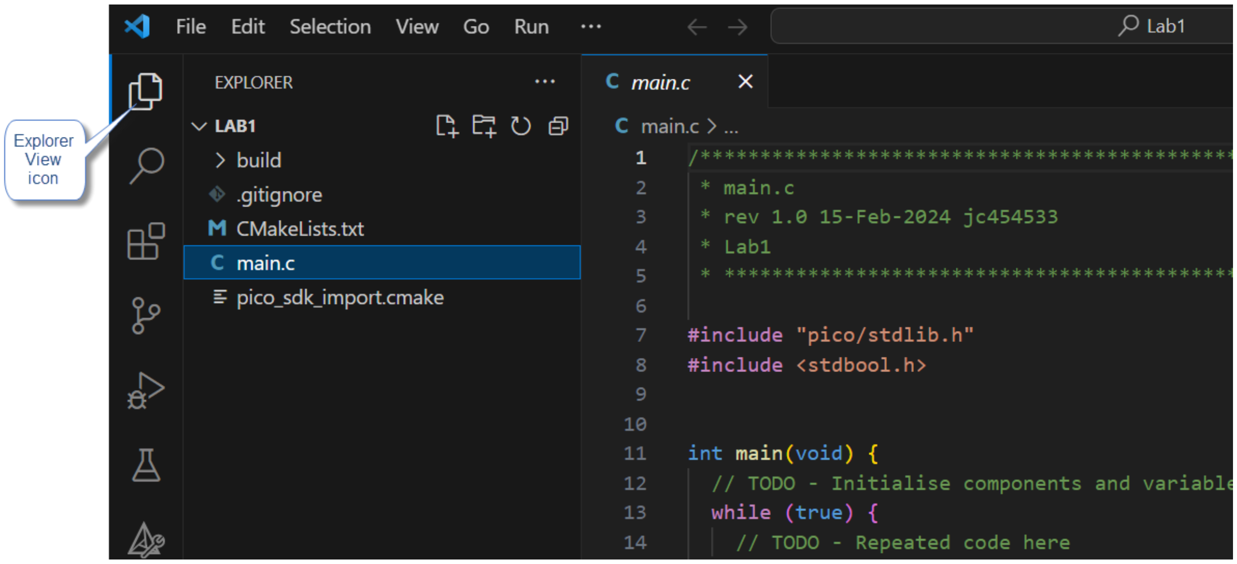
Your new project is now created, open & configured. It is ready for you to work with.

Working with a project

This section assumes that you have created a project and configured it as discussed above. If you have not yet configured your project for building, then refer to the instructions in ***Open the project*** above, some of which may still be needed.

Exploring the project

Your project contains several source files. You can use the ***Explorer View*** in VS Code to inspect them. Press ***Ctrl+Shift+E*** or click on the Explorer View icon to see the view. You can click on a file in the list to view and edit it:



```
/*********************************************
 * main.c
 * rev 1.0 15-Feb-2024 jc454533
 * Lab1
 * *******************************************

#include "pico/stdlib.h"
#include <stdbool.h>


int main(void) {
  // TODO - Initialise components and variable
  while (true) {
    // TODO - Repeated code here
```

The following files are important:

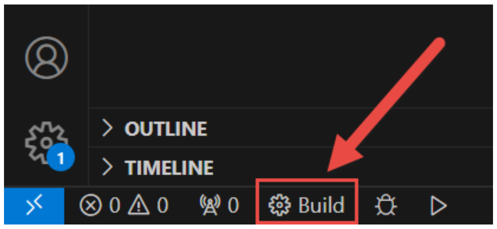
File	Description
CMakeLists.txt	This is the project make file. It contains instructions to the build tools, telling them how to convert your source code to an executable binary program.
main.c	This is a 'C' language source file. It contains the source code for your new program.

Edit your code

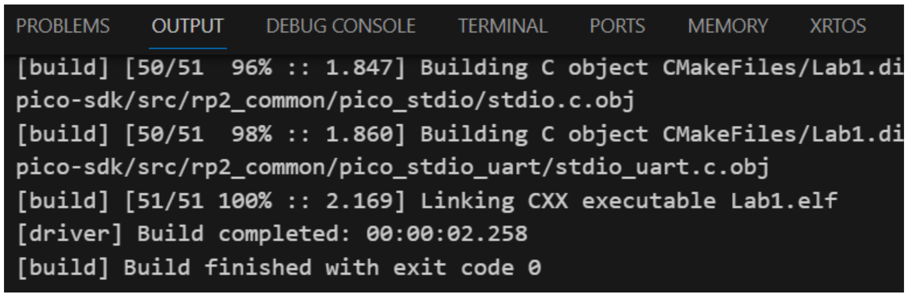
- Edit your source code in the right-hand-side pane of VS Code.
- Save your changes by pressing ***Ctrl+S*** to save the current file or ***Ctrl+K, S*** to save all changed files. (Alternatively use the ***File > Save*** or ***File > Save All*** menus.)

Building your program

To convert your source code to an executable program, you need to build it. Press the function key ***F7*** or click the ***Build*** tool in the status bar at the bottom of the VS Code window.



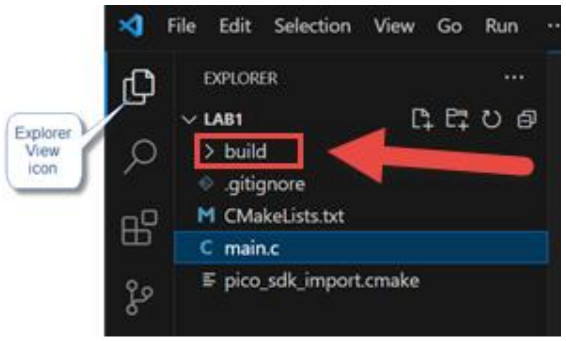
If the compilation is successful, you will see a line containing "Build finished with exit code 0" in the ***Output*** pane:



```
[build] [50/51  96% :: 1.847] Building C object CMakeFiles/Lab1.di
pico-sdk/src/rp2_common/pico_stdio/stdio.c.obj
[build] [50/51  98% :: 1.860] Building C object CMakeFiles/Lab1.di
pico-sdk/src/rp2_common/pico_stdio_uart/stdio_uart.c.obj
[build] [51/51 100% :: 2.169] Linking CXX executable Lab1.elf
[driver] Build completed: 00:00:02.258
[build] Build finished with exit code 0
```

Checkpoint 7

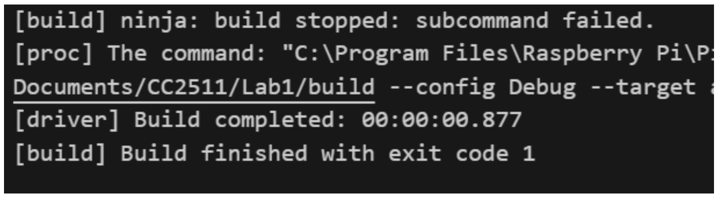
1. Did the build succeed? i.e. Was the exit code 0?
2. Is there a new folder named build visible in the Project Explorer?



3. If you expand the folder can you see a file named `Lab1.uf2`?

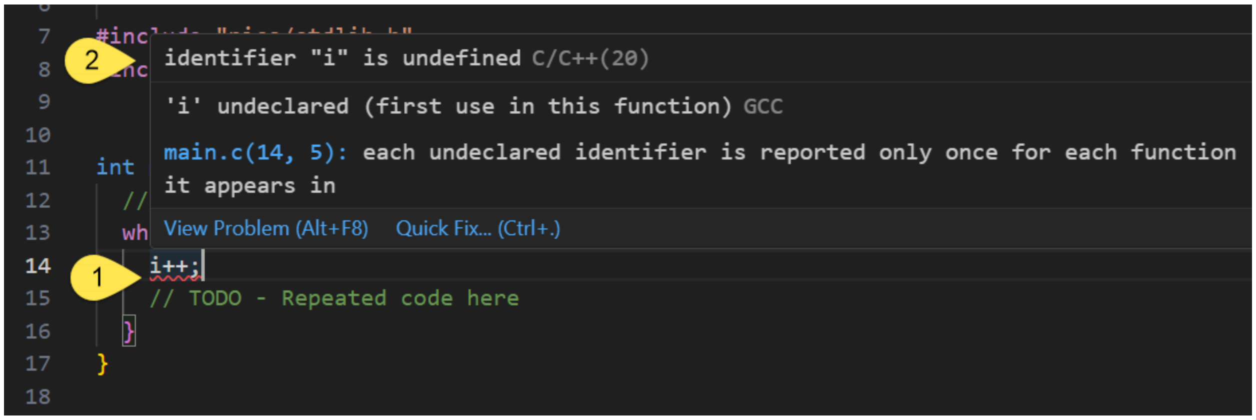
Errors

If there are any errors, they will appear in the Output window, followed by an exit code 1:



```
[build] ninja: build stopped: subcommand failed.
[proc] The command: "C:\Program Files\Raspberry Pi\P
Documents/CC2511/Lab1/build --config Debug --target
[driver] Build completed: 00:00:00.877
[build] Build finished with exit code 1
```

You can inspect your errors in the source code. An error will have a red squiggly line under it (1); if you float the cursor above the error location, you will see a description of the error (2), e.g.:



```
#incl
 nc
int
    //
    wh
        i++;
        // TODO - Repeated code here
    }
}
```

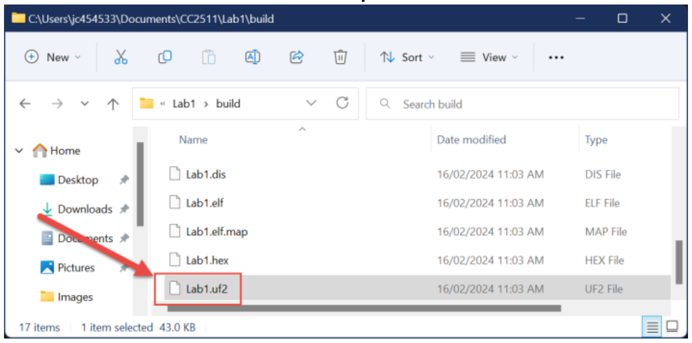
```
identifier "i" is undefined C/C++(20)
'i' undeclared (first use in this function) GCC
main.c(14, 5): each undeclared identifier is reported only once for each function it appears in
```

You should correct each compilation error and then try to build again, until you are successful.

Delivering a program to the Pico

When your program has been successfully built, you need to deliver it to the Pico where it will run.

1) In Windows Explorer, locate the folder containing your project.
   Tip – Press ***Shift+Alt+R*** in VS Code to open the correct folder.
2) In the ***Build*** sub-folder, locate the file with the extension ".uf2", e.g. `Lab1.uf2`.
   Tip – You can make file extensions visible by clicking View > Show > File name extensions in Windows Explorer.

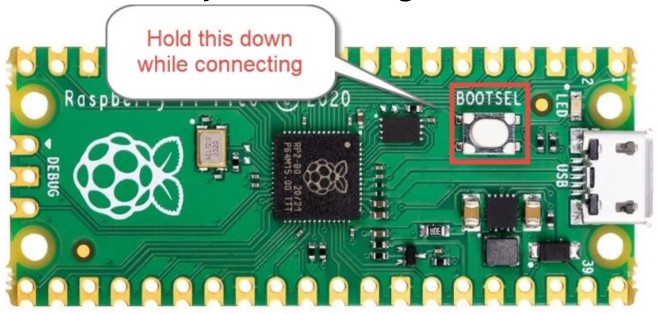


C:\Users\jc454533\Documents\CC2511\Lab1\build

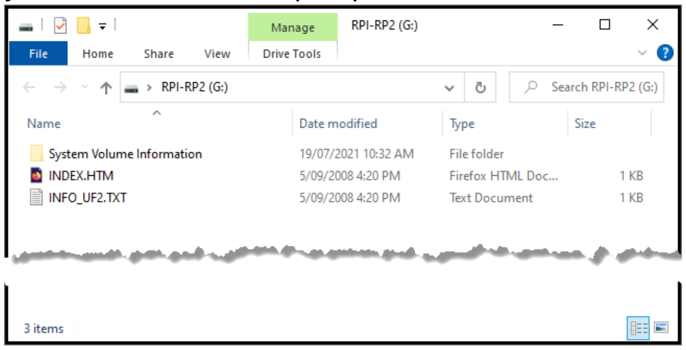
Name	Date modified	Type
Lab1.dis	16/02/2024 11:03 AM	DIS File
Lab1.elf	16/02/2024 11:03 AM	ELF File
Lab1.elf.map	16/02/2024 11:03 AM	MAP File
Lab1.hex	16/02/2024 11:03 AM	HEX File
Lab1.uf2	16/02/2024 11:03 AM	UF2 File

17 items 1 item selected 43.0 KB

3) Connect a Pico board to your computer using a USB cable. As you plug in the cable, make sure that you are holding down the BOOTSEL button.



4) When you release the BOOTSEL button, a new drive (named RPI-RP2) will appear on your windows desktop. Open the new drive in Windows Explorer:



Name	Date modified	Type	Size
System Volume Information	19/07/2021 10:32 AM	File folder	
INDEX.HTM	5/09/2008 4:20 PM	Firefox HTML Doc...	1 KB
INFO_UF2.TXT	5/09/2008 4:20 PM	Text Document	1 KB

3 items

5) Copy the uf2 file (identified in step 2 above) to the new drive.
6) The new drive window will close and your program will now start running on the Pico board.

Note – there is a simpler* ‘hands-free’ *method to deliver a program to the Pico board, which you will use when you start debugging.

Connecting the Pico to a Terminal

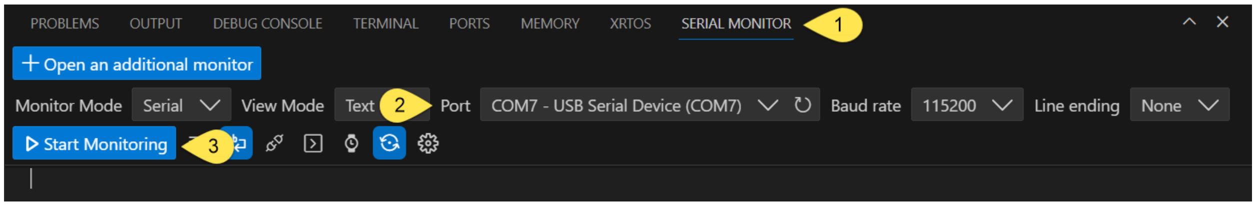
You can connect your Pico to a terminal to send text between the Pico and your PC, using eth serial port.

Source code

You may need to add code to your project to enable the use of the serial port.

1) In `CMakeLists.txt`, add `pico_stdio_usb` to the list of libraries in the `target_link_libraries` statement, e.g.:
```
target_link_libraries(${projname} pico_stdlib pico_stdio_usb)
```
Note - If you built your project with the -d picoprobe flag, you can skip this step.
2) In `main.c`, initialise the stdio (standard input/output) subsystem by adding the following line at near the start of the main() function:
```
stdio_init_all();
```

Serial Monitor



Monitor Mode Serial View Mode Text Port COM7 - USB Serial Device (COM7) Baud rate 115200 Line ending None

1) In Visual Studio Code, alongside the ***Problems***, ***Output*** and ***Terminal*** windows, there is a ***Serial Monitor*** window which you can use as a terminal.
2) Your Pico will appear in the list of Ports. (Select the Port if there are multiple choices available.)
3) Click on ***Start Monitoring*** and output will appear in the lower part of the window.

You can stop it by pressing the ***Stop Monitoring*** button.

Note that this terminal emulator is not ideal for sending text from the PC to the Pico – you need to first type into the field at the bottom of the panel and then press the enter key to send any data from the input field to the Pico.

Putty is an application that provides a more complete and useful terminal emulation (see below).

GitHub Desktop

There are many ways to work with git – the source control system – and with GitHub – the cloud based storage system for git code repositories (or 'repos').

Three are easily available to you in this class.

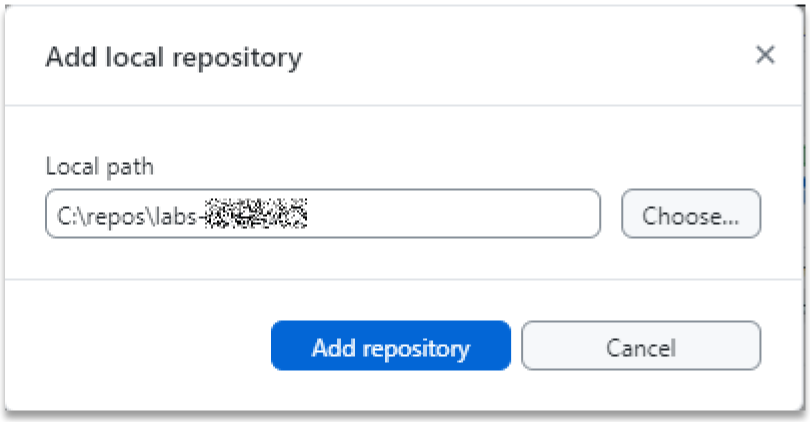
a) **Command line**. You can use the git command line tool from the ***Pico – Developer Command Prompt***. This was used to clone your repo in ***Setting up your work area*** above. However, this is not a very user-friendly environment to start with.

b) **GitHub Desktop**. This is a user-friendly tool which can be downloaded from the GitHub website. For this course, it is probably the most useful option to begin with.

c) **VS Code Extension**. There is an extension available for use within VS Code, which is installed by the Pico installation script. After a while, you may find this a more efficient interface.

To install and setup GitHub Desktop follow these steps:

1. Open a browser to https://github.com/apps/desktop and download GitHub Desktop for Windows (64bit).
2. Install the program using the default settings. You will be asked to log into your GitHub account.
3. Use File > Add local repository (or press Ctrl+O) and select the repository that you cloned earlier:

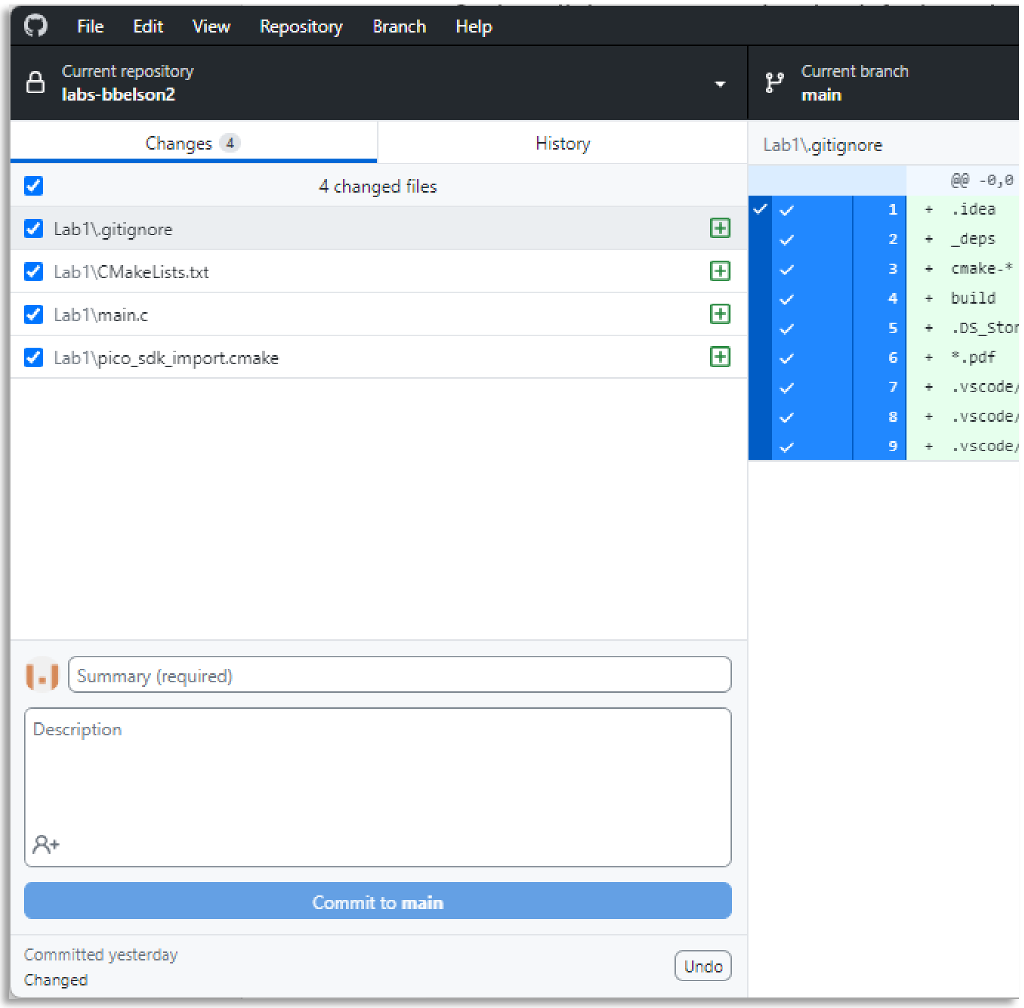


Add local repository

Local path

C:\repos\labs-[redacted]

4. The repository will be displayed in the application:



```
@@ -0,0
+ .idea
+ _deps
+ cmake-*
+ build
+ .DS_Stor
+ *.pdf
+ .vscode/
+ .vscode/
+ .vscode/
```

In the Changes tab, you can see a list of changed files, and you can select each to see details of recent changes.

Commit

You can save a snapshot of your work by creating a ***commit***.

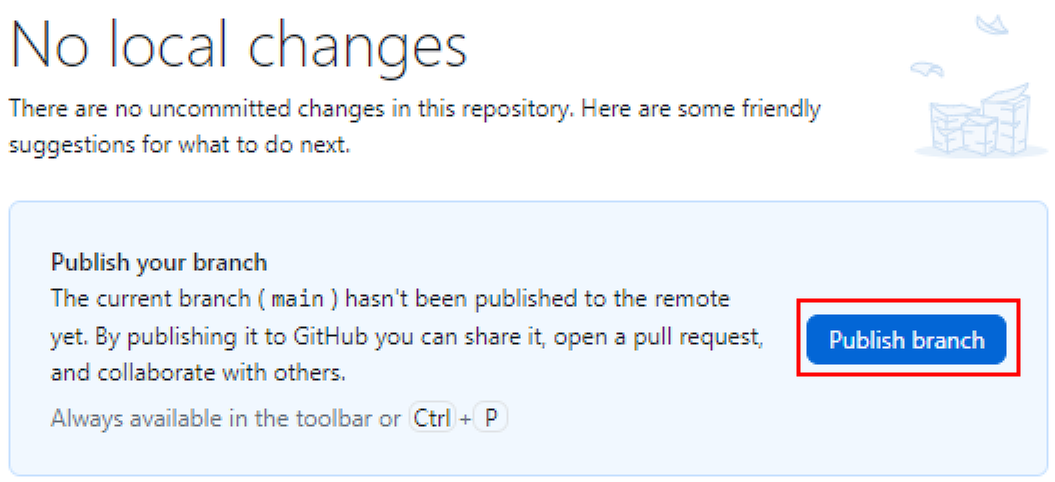
i) Type a short, useful description of the changes into the ***Summary*** field.
ii) Click on ***Commit to main***.

The changes are now committed to the local repo. However they have not yet been copied up to the GitHub cloud.

Push

A number of new commands are now available on the right-hand side. You can now publish your changes to the GitHub cloud by ***pushing*** them.

If this is the first content you are adding to the repo you will see a panel like this:


No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

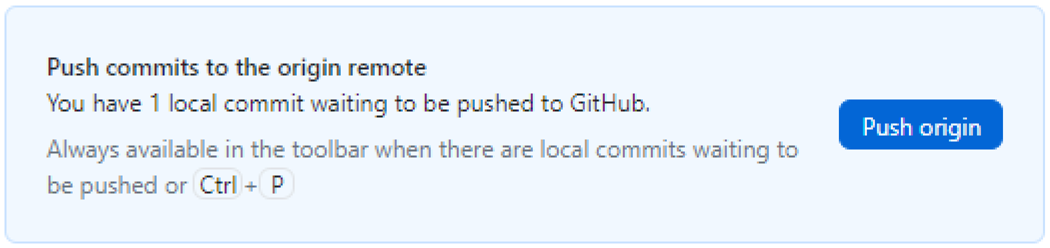
Publish your branch

The current branch (`main`) hasn't been published to the remote yet. By publishing it to GitHub you can share it, open a pull request, and collaborate with others.

Always available in the toolbar or Ctrl + P

Click on ***Publish branch*** to push the committed changes up to the cloud.

If this is not your first push, you will see something like this:


Push commits to the origin remote

You have 1 local commit waiting to be pushed to GitHub.

Always available in the toolbar when there are local commits waiting to be pushed or Ctrl + P

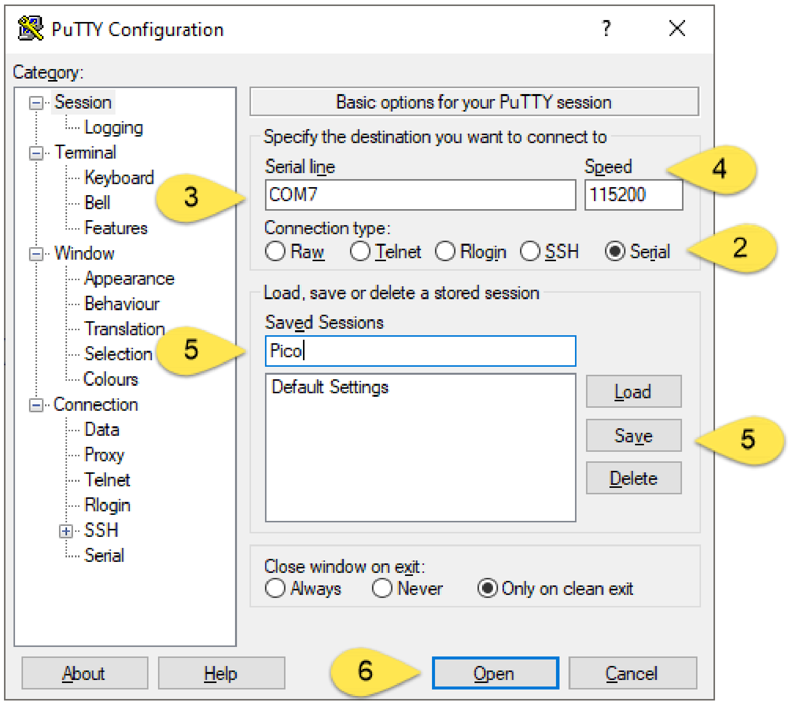
Click on ***Push origin*** to push the committed changes up to the cloud.

Cloud synchronisation

Open your repo in GitHub, and you will be able to see your new code, and a history of the changes you have made.

Putty

For some labs and for Assignment 2, you will need to use Putty as a terminal emulator. You can download it from https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html. This will allow you to use more powerful terminal facilities, including cursor control and colours.

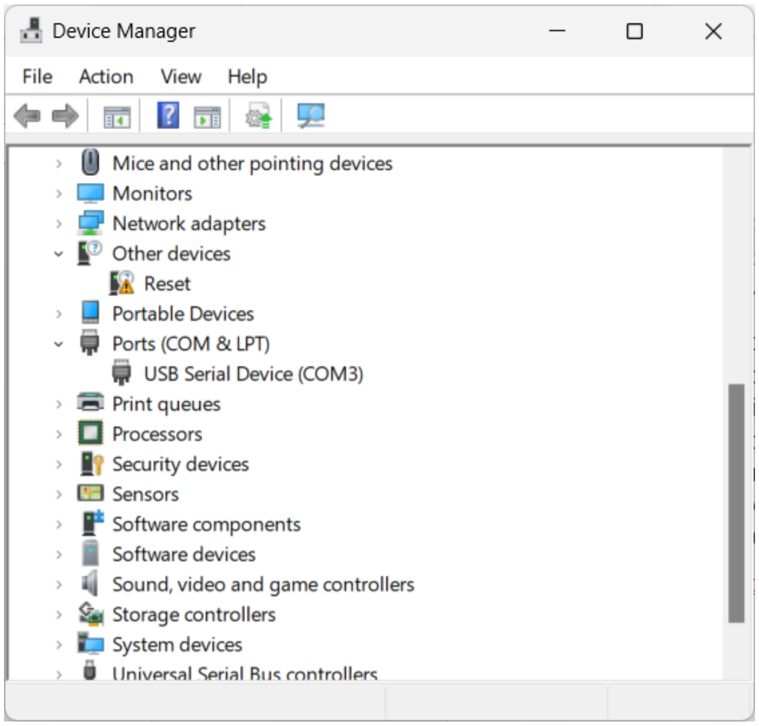


Serial line: COM7
Speed: 115200
Connection type: Raw Telnet Rlogin SSH [✓ Serial]
Saved Sessions: Pico
Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

After installing the application, follow the steps below.

1) Start the Putty application.
2) Set Connection type to Serial.
3) In Serial line enter the COM port name.
4) Set Speed to 115200.
5) Optionally, set the Saved Sessions name to Pico and Save the Session. This will save you time next time you run Putty: you'll be able to load these settings.
6) Click Open to start the session.

Note:- To determine which COM port the Pico is attached to, you can use the Device Manager tool – look in the Ports (COM & LPT) folder.



Mice and other pointing devices
Monitors
Network adapters
Other devices
Reset
Portable Devices
Ports (COM & LPT)
USB Serial Device (COM3)
Print queues
Processors
Security devices
Sensors
Software components
Software devices
Sound, video and game controllers
Storage controllers
System devices
Universal Serial Bus controllers

Debugging your program

Debugging your program can hugely improve your productivity and improve program quality and reliability. The Raspberry Pi Pico offers very powerful and convenient debugging facilities.

You will need to use either the specialised unit - the Raspberry Pi Debug Probe – or a second Pico board (known as a Picoprobe).

Enable debugging

You can enable debugging in your project by using the `-d picoprobe2` command-line option when you create the new project, e.g.:

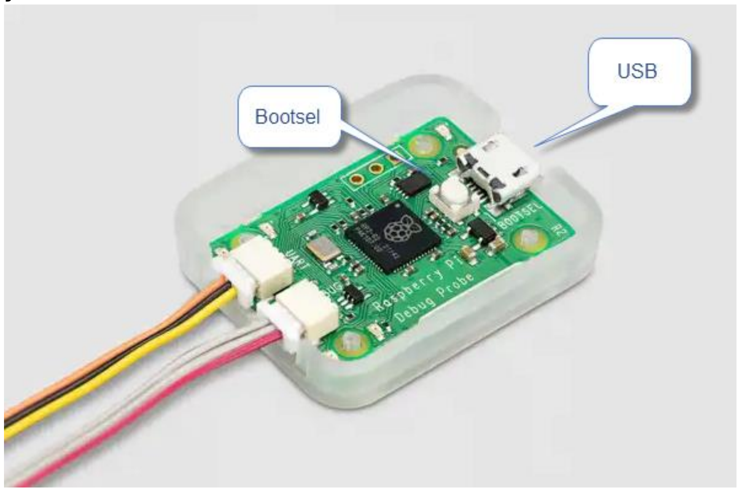
```
python %pico_repos_path%\ppgen\ppgen.py -d picoprobe2 Lab1
```

This will add a new file, `launch.json`, to your project, in the `.vscode` subfolder.

Set up the Debug Probe

First, make sure that the Debug Probe has the correct firmware.

1) Download the file ***debugprobe.uf2***.
2) Remove the plastic lid of the Debug Probe.
3) Holding down the BOOTSEL button, connect the USB port of the debug probe to your PC.



4) Release the BOOTSEL button.
5) In Windows Explorer, open the new temporary drive that appears in your windows system, named RPI_RP2.
6) Copy the file ***debugprobe.uf2*** on to the new drive. The drive window will close.

Next, connect the Debug Probe to your Pico as follows:

- The Debug Probe "D" connector to Pico H SWD JST-SH connector
- The Debug Probe "U" connector has the three-pin JST-SH connector to 0.1-inch header (male)
  - Debug Probe RX connected to Pico H TX pin

- Debug Probe TX connected to Pico H RX pin
- Debug Probe GND connected to Pico H GND pin

Note: Make sure you connect your Pico to a power source. (Normally you will connect to a PC using the Pico's USB port; alternatively, you could provide 5V and 0V to the Pico's VBUS and GND pins.)

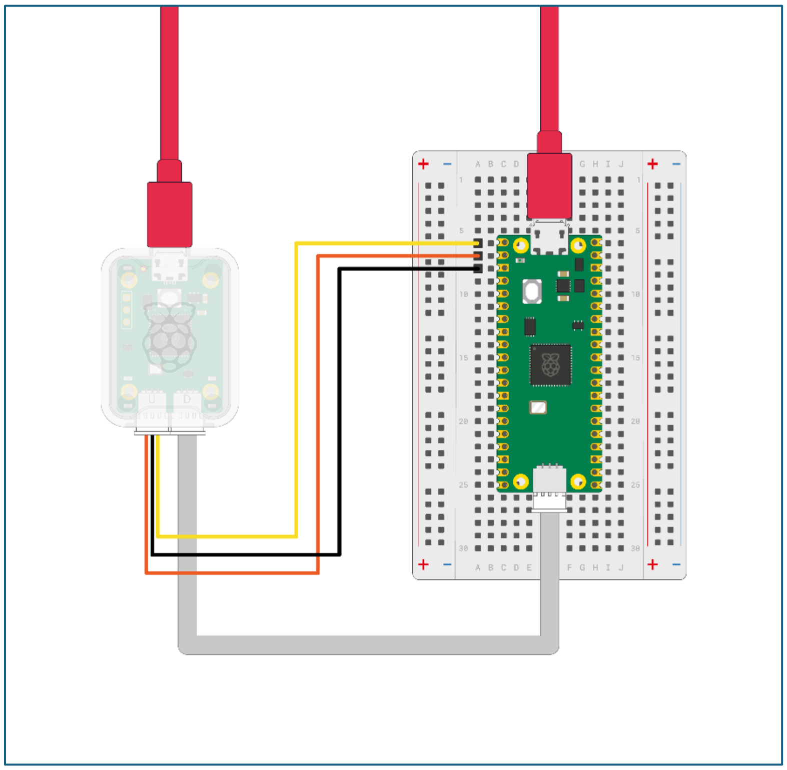


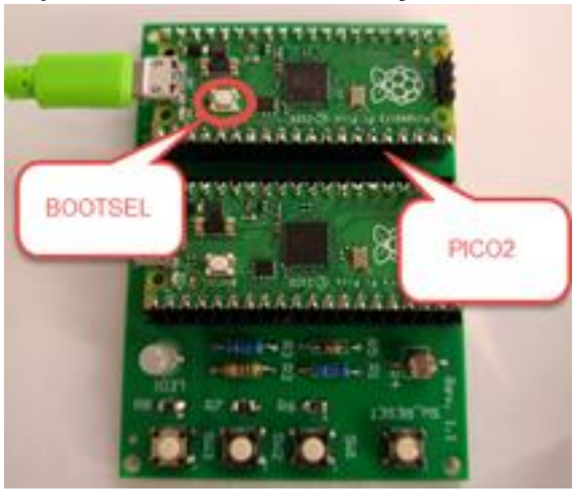
Figure: Raspberry Pi Foundation, 2024

Set up Picoprobe

If you have a development board with two Pico boards, follow the steps below. First, make sure that the Debug Probe has the correct firmware.

1) Download the file ***debugprobe_on_pico.uf2*** from LearnJCU, in the same folder as this guide.
2) Holding down the BOOTSEL button, connect the USB port of the debug Pico (PICO2) to your PC.

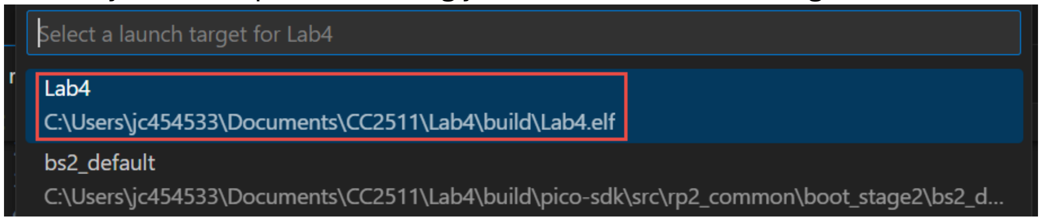
Important – make sure that you are connected to PICO2 not PICO1.



3) Release the BOOTSEL button.
4) A new Windows Explorer will open, containing a virtual drive for the PICO2 device. If it does not open automatically, open a new window for the drive, which will be named RPI_RP2.)
5) Copy the file picoprobe.uf2 (located in step 1 above) on to the Pico virtual drive.

Debugging a Project

1) To debug a project, make sure that the project has debugging enabled (see ***Enable debugging*** above).
2) Once you have successfully built your project, with no errors, press function key F5 to start the program in the debugger.
3) VS Code will deliver an executable file to your debugged Pico, via the debugging device (either the Debug Probe or a Picoprobe).
4) You may see a drop-down asking you to select a launch target:



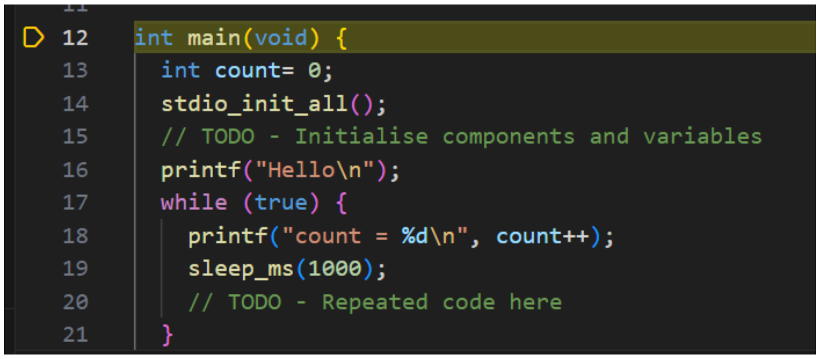
Select a launch target for Lab4

Lab4
C:\Users\jc454533\Documents\CC2511\Lab4\build\Lab4.elf

bs2_default
C:\Users\jc454533\Documents\CC2511\Lab4\build\pico-sdk\src\rp2_common\boot_stage2\bs2_d...

Select the main target, which has the same name as your project.

The program will start on the Pico, and will then automatically pause at the first line of the `main()` function:



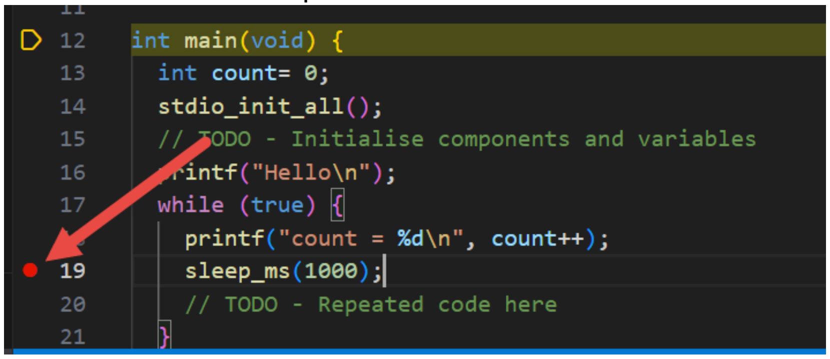
```
12  int main(void) {
13      int count= 0;
14      stdio_init_all();
15      // TODO - Initialise components and variables
16      printf("Hello\n");
17      while (true) {
18          printf("count = %d\n", count++);
19          sleep_ms(1000);
20          // TODO - Repeated code here
21      }
```

You can now use the debugger to control the execution of your program:

- Press ***F10*** to execute one line of code.
- Press ***F11*** to debug inside a function called from the current line of code.
- Use ***F5*** to let the debugger continue to execute, until it reaches a break-point.
- Press ***Shift+F5*** to exit the debugger and stop execution.

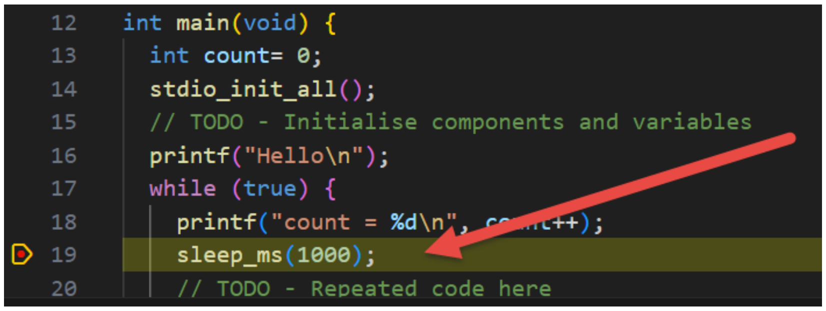
Breakpoints

Use ***F9*** to set a break-point on a selected line of code:



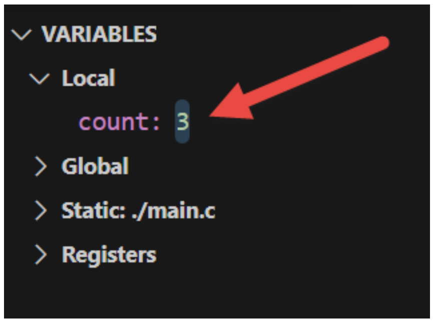
```
12  int main(void) {
13      int count= 0;
14      stdio_init_all();
15      // TODO - Initialise components and variables
16      printf("Hello\n");
17      while (true) {
        printf("count = %d\n", count++);
19          sleep_ms(1000);
20          // TODO - Repeated code here
21      }
```

A red dot will appear on the line. When the code reaches line 19, execution will pause and you will be able to inspect variable values.



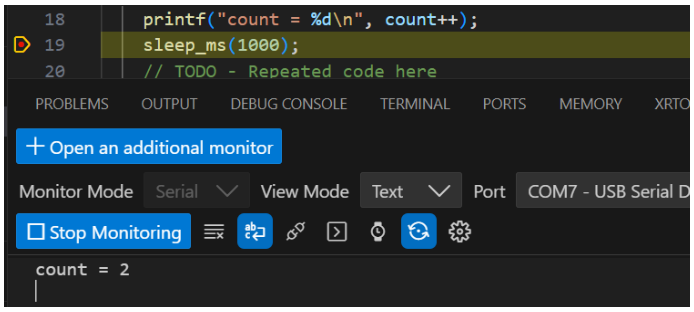
```
12  int main(void) {
13      int count= 0;
14      stdio_init_all();
15      // TODO - Initialise components and variables
16      printf("Hello\n");
17      while (true) {
18          printf("count = %d\n", count++);
19          sleep_ms(1000);
20          // TODO - Repeated code here
```

On the left-hand side of the screen, the Debugging pane contains state information for your program, including the values of all variables:



```
count: 3
```

If you have made a call to `stdio_init_all()` in your program, then you will be able to see serial output in a terminal emulator (see ***Connecting the Pico to a Terminal*** above).



```
printf("count = %d\n", count++);
sleep_ms(1000);
// TODO - Repeated code here
```

Monitor Mode Serial View Mode Text Port COM7 - USB Serial D

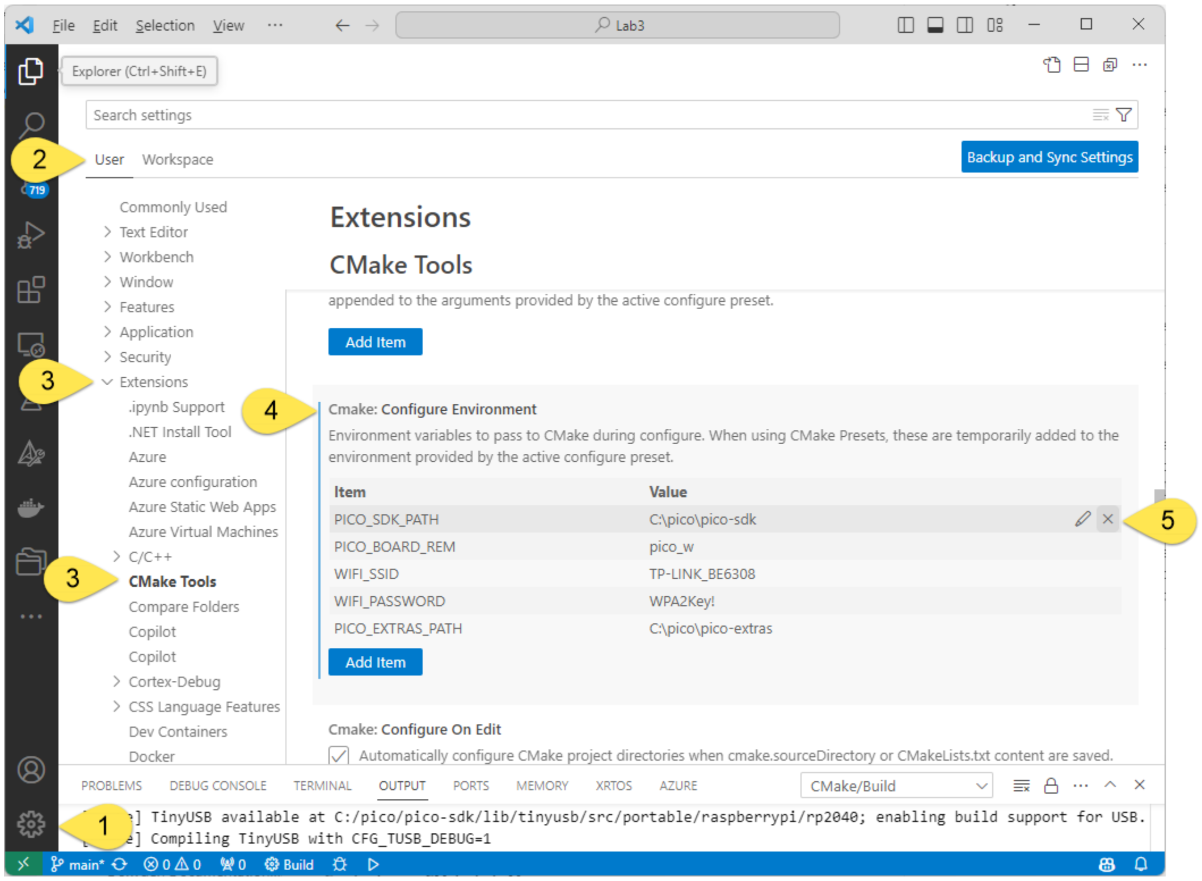
```
count = 2
```

Troubleshooting

Previous installation

If you have had a previous installation of the Pico SDK on your computer you may encounter problems while building and/or debugging within VS Code.

Follow these steps to fix the build process:



Extensions

CMake Tools

appended to the arguments provided by the active configure preset.

Add Item

Cmake: Configure Environment

Environment variables to pass to CMake during configure. When using CMake Presets, these are temporarily added to the environment provided by the active configure preset.

Item	Value
PICO_SDK_PATH	C:\pico\pico-sdk
PICO_BOARD_REM	pico_w
WIFI_SSID	TP-LINK_BE6308
WIFI_PASSWORD	WPA2Key!
PICO_EXTRAS_PATH	C:\pico\pico-extras

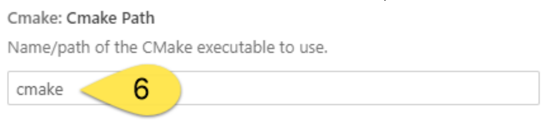
Add Item

Cmake: Configure On Edit

✓ Automatically configure CMake project directories when cmake.sourceDirectory or CMakeLists.txt content are saved.

```
] TinyUSB available at C:/pico/pico-sdk/lib/tinyusb/src/portable/raspberrypi/rp2040; enabling build support for USB.
] Compiling TinyUSB with CFG_TUSB_DEBUG=1
```

1. Open the Settings panel - either:
   a. Click on the gear symbol at the bottom-left of VS Code and then click ***Settings***; or
   b. Press ***Ctrl+,*** (comma)
2. Select ***User*** at the top of the panel.
3. Select ***Extensions > CMake Tools*** in the list.
4. Locate ***<u>Cmake: Configure Environment</u>***
5. If `PICO_SDK_PATH` is present, delete it.
6. Locate ***Cmake: Cmake Path***, and set it to `cmake`.

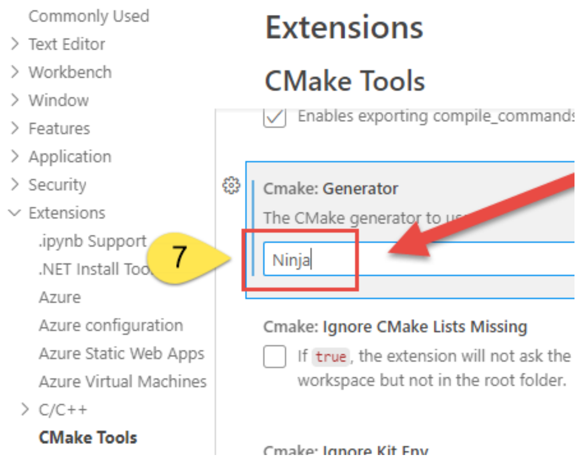


Cmake: Cmake Path

Name/path of the CMake executable to use.

cmake

7. Locate ***Cmake: Generator***, and set it to `Ninja`.



Enables exporting compile_commands

Cmake: Generator
The CMake generator to u

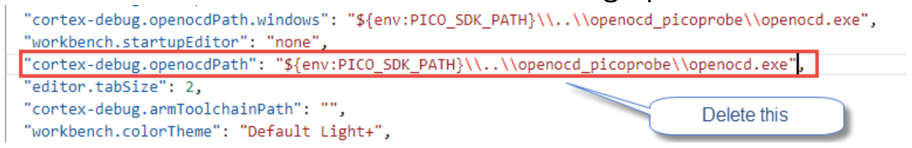
Ninja

Cmake: Ignore CMake Lists Missing
If true, the extension will not ask the
workspace but not in the root folder.

Cmake: Ignore Kit Env

Follow these steps to fix the debugging process:

1. Open the Settings panel.
2. Select ***User*** at the top of the panel.
3. Select ***Extensions > Cortex-Debug*** in the list.
4. Locate ***Cmake: Openocd Path*** and click ***Edit in settings.json***.
5. Delete the line that starts with "cortex-debug.openocdPath"



```
"cortex-debug.openocdPath.windows": "${env:PICO_SDK_PATH}\\..\\openocd_picoprobe\\openocd.exe",
"workbench.startupEditor": "none",
"cortex-debug.openocdPath": "${env:PICO_SDK_PATH}\\..\\openocd_picoprobe\\openocd.exe",
"editor.tabSize": 2,
"cortex-debug.armToolchainPath": "",
"workbench.colorTheme": "Default Light+",
```

6. Ensure that you have copied the latest version of **debugprobe_on_pico.uf2** (provided along with this document) to your debugging Pico device.