



DesignNews

Constructing a Raspberry Pi RP2040 Low-Power Sensor Node

Day 1:

Assembling the RP2040 Linux Toolchain

Sponsored by



Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

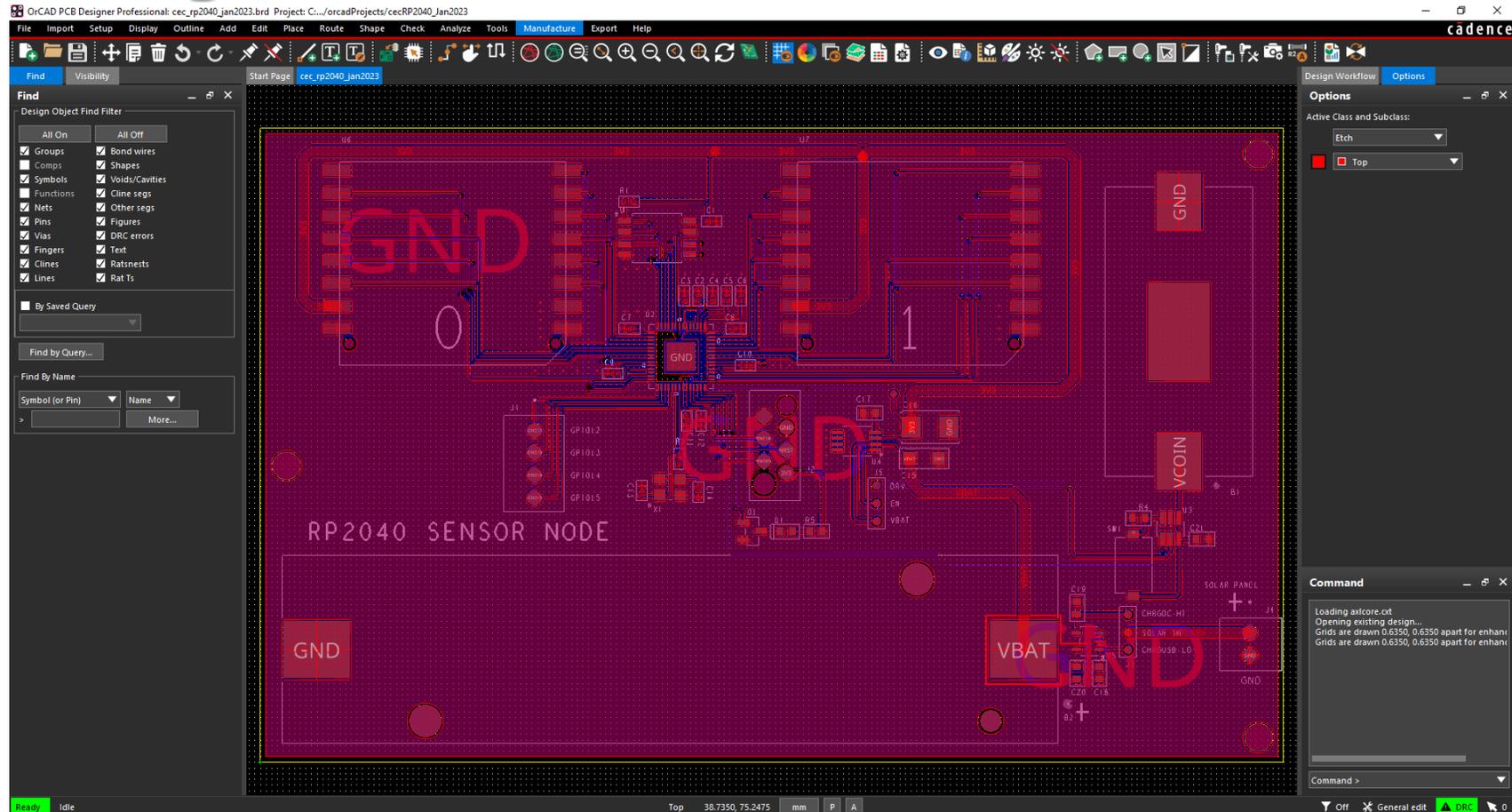


Fred Eady

Visit 'Lecturer Profile' in your console for more details.

AGENDA

- **Install the Raspberry Pi Pico C/C++ SDK**
- **Install the *gcc-arm-none-eabi* Toolchain**
- **Install and Configure Visual Studio Code**
- **Test Flight**



Create the *pico* Directory and Install git

```
fred@shoptc1660: ~/pico
File Edit View Search Terminal Help

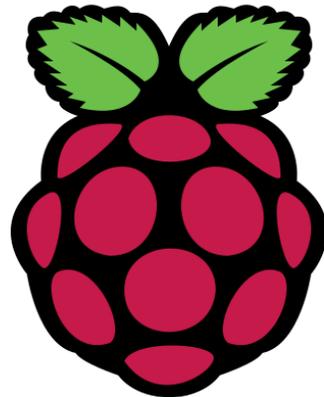
fred@shoptc1660:~$ mkdir pico
fred@shoptc1660:~$ cd pico
fred@shoptc1660:~/pico$

fred@shoptc1660:~/pico$ sudo apt install git
[sudo] password for fred:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,112 kB of archives.
After this operation, 20.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.5 [953 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.5 [3,132 kB]
Fetched 4,112 kB in 1s (4,085 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 198974 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../git-man_1%3a2.34.1-1ubuntu1.5_all.deb ...
Unpacking git-man (1:2.34.1-1ubuntu1.5) ...
Selecting previously unselected package git.
Preparing to unpack .../git_1%3a2.34.1-1ubuntu1.5_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.5) ...
Setting up liberror-perl (0.17029-1) ...
Setting up git-man (1:2.34.1-1ubuntu1.5) ...
Setting up git (1:2.34.1-1ubuntu1.5) ...
Processing triggers for man-db (2.10.2-1) ...
fred@shoptc1660:~/pico$
```

Install the Raspberry Pi Pico C/C++ SDK

```
fred@shoptc1660: ~/pico
File Edit View Search Terminal Help

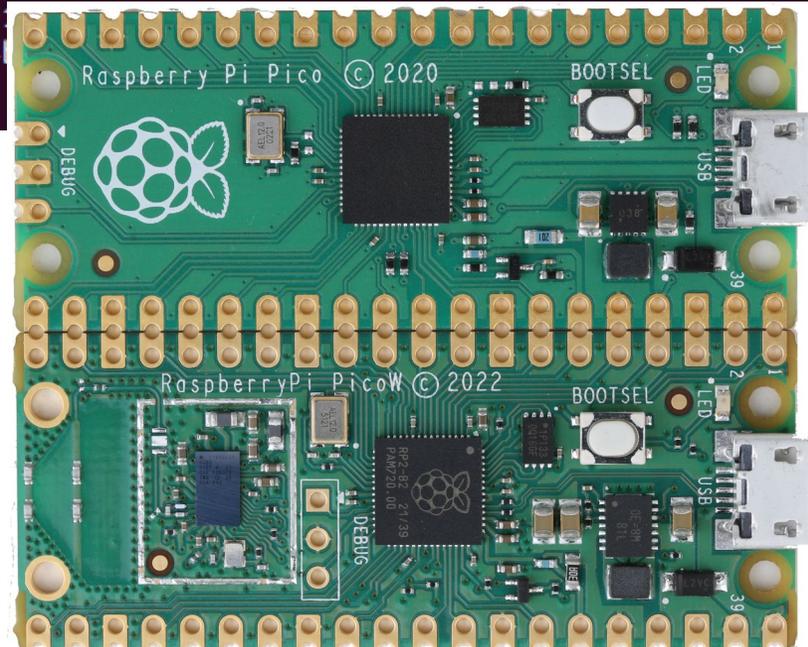
fred@shoptc1660:~/pico$ git clone https://github.com/raspberrypi/pico-sdk.git --branch master
Cloning into 'pico-sdk'...
remote: Enumerating objects: 5748, done.
remote: Counting objects: 100% (1306/1306), done.
remote: Compressing objects: 100% (291/291), done.
remote: Total 5748 (delta 1096), reused 1051 (delta 1011), pack-reused 4442
Receiving objects: 100% (5748/5748), 2.36 MiB | 4.09 MiB/s, done.
Resolving deltas: 100% (2992/2992), done.
fred@shoptc1660:~/pico$
```



Install *tinycusb* and the Pico W Drivers

```
fred@shoptc1660: ~/pico/pico-sdk
File Edit View Search Terminal Help

fred@shoptc1660:~/pico$ cd pico-sdk
fred@shoptc1660:~/pico/pico-sdk$ git submodule update --init
Submodule 'lib/cyw43-driver' (https://github.com/georgerobotics/cyw43-driver.git) registered for path 'lib/cyw43-driver'
Submodule 'lib/lwip' (https://github.com/lwip-tcpip/lwip.git) registered for path 'lib/lwip'
Submodule 'tinycusb' (https://github.com/hathach/tinycusb.git) registered for path 'lib/tinycusb'
Cloning into '/home/fred/pico/pico-sdk/lib/cyw43-driver'...
Cloning into '/home/fred/pico/pico-sdk/lib/lwip'...
Cloning into '/home/fred/pico/pico-sdk/lib/tinycusb'...
Submodule path 'lib/cyw43-driver': checked out '195dfcc10bb6f379e3dea45147590db2203d3c7b'
Submodule path 'lib/lwip': checked out '2399...'
Submodule path 'lib/tinycusb': checked out '4...'
fred@shoptc1660:~/pico/pico-sdk$
```



Install the Pico Examples

```
fred@shoptc1660: ~/pico
File Edit View Search Terminal Help
fred@shoptc1660:~/pico/pico-sdk$ cd ..
fred@shoptc1660:~/pico$ git clone https://github.com/raspberrypi/pico-examples.git --branch master
Cloning into 'pico-examples'...
remote: Enumerating objects: 2126, done.
remote: Counting objects: 100% (52/52), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 2126 (delta 18), reused 25 (delta 11), pack-reused 2074
Receiving objects: 100% (2126/2126), 7.59 MiB | 6.56 MiB/s, done.
Resolving deltas: 100% (1085/1085), done.
fred@shoptc1660:~/pico$
```

Execute an Ubuntu Update

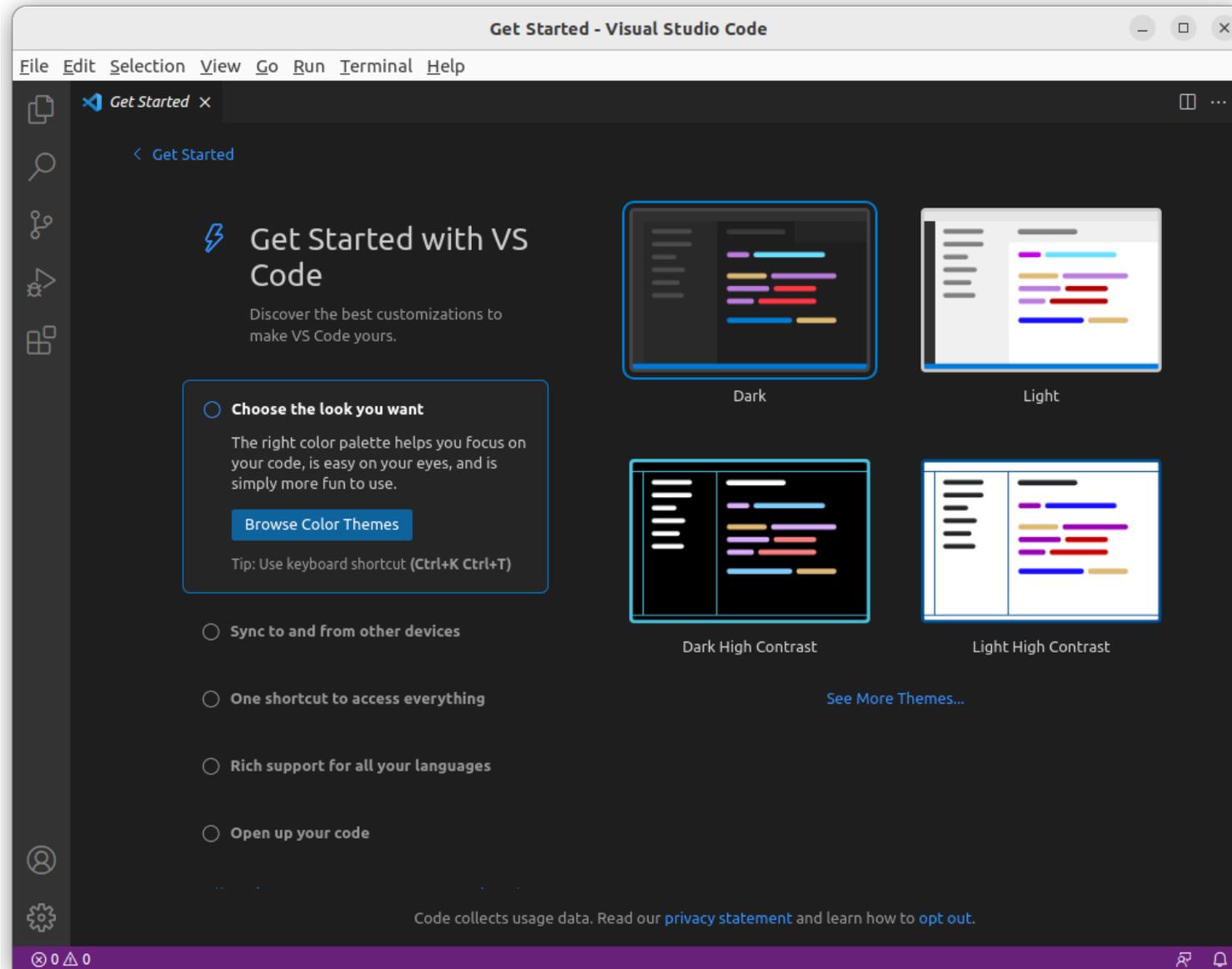
```
fred@shoptc1660: ~  
File Edit View Search Terminal Help  
fred@shoptc1660:~/pico$ cd  
fred@shoptc1660:~$ sudo apt update  
Hit:1 http://us.archive.ubuntu.com/ubuntu jammy InRelease  
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]  
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]  
Get:4 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]  
Get:5 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [94.8 kB]  
Get:6 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [257 kB]  
Get:7 http://us.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 DEP-11 Metadata [940 B]  
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [20.1 kB]  
Get:9 http://us.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 DEP-11 Metadata [11.7 kB]  
Get:10 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [13.3 kB]  
Fetched 722 kB in 1s (542 kB/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
All packages are up to date.  
fred@shoptc1660:~$
```

Install cmake – build-essential – libs and the GCC Toolchain

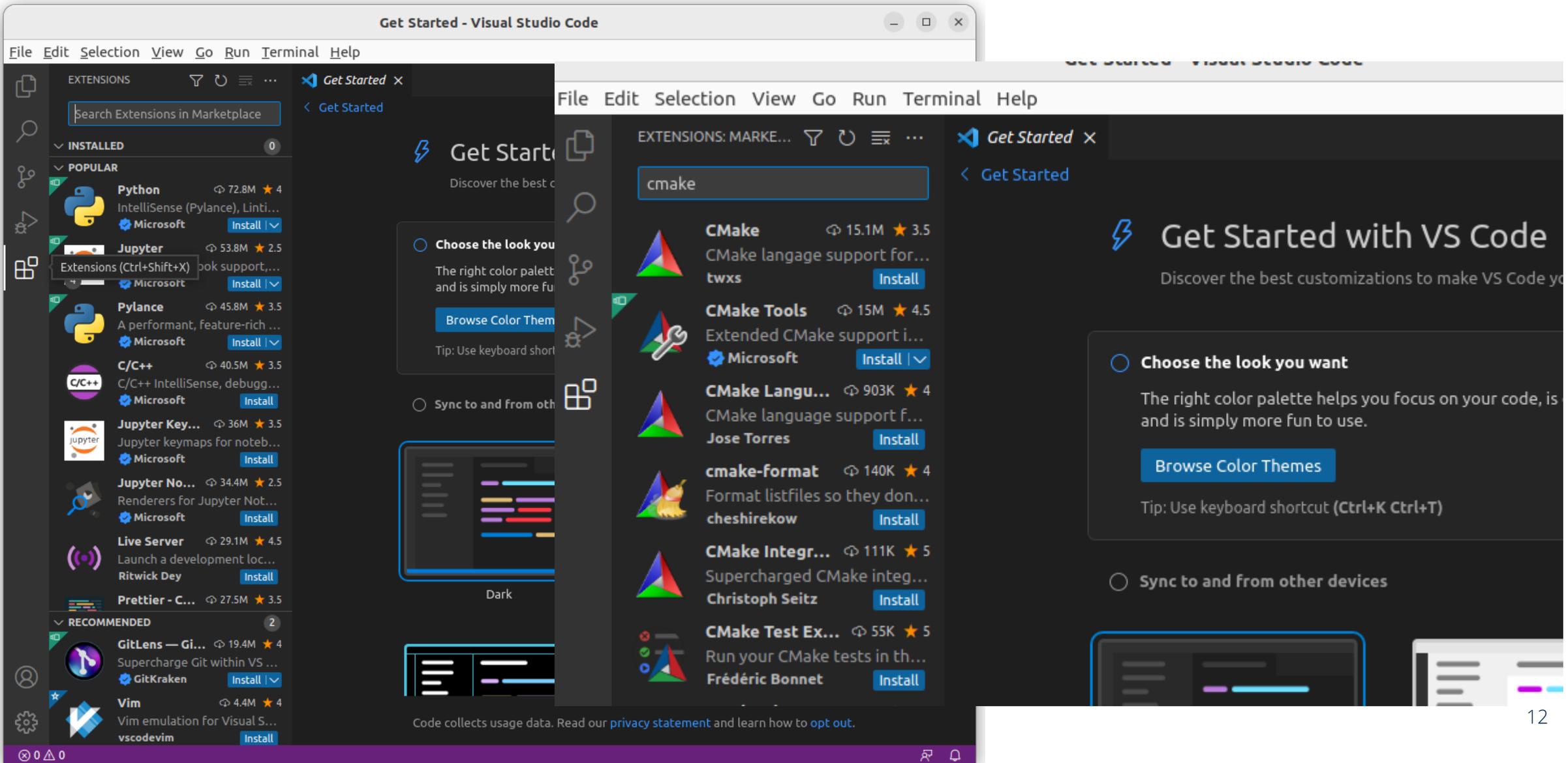
```
fred@shoptc1660: ~  
File Edit View Search Terminal Help  
fred@shoptc1660:~$ sudo apt install cmake gcc-arm-none-eabi libnewlib-arm-none-eabi build-essential libstdc++-arm-none-eabi-newlib
```



Download and Install Visual Studio Code



Install the CMake Tools Extension



The image shows two overlapping screenshots of the Visual Studio Code interface. The background screenshot shows the 'Get Started' page with a 'Choose the look you want' section. The foreground screenshot shows the 'EXTENSIONS: MARKETPLACE' view with 'cmake' searched in the search bar. The search results list several CMake-related extensions, with 'CMake Tools' by Microsoft highlighted.

EXTENSIONS: MARKETPLACE

Search Extensions in Marketplace

cmake

Extension Name	Author	Downloads	Rating	Action
CMake	Microsoft	15.1M	3.5	Install
CMake Tools	Microsoft	15M	4.5	Install
CMake Language Support for Visual Studio Code	Jose Torres	903K	4	Install
cmake-format	cheshirekow	140K	4	Install
CMake Integrator	Christoph Seitz	111K	5	Install
CMake Test Explorer	Frédéric Bonnet	55K	5	Install

Get Started with VS Code

Discover the best customizations to make VS Code your own.

- Choose the look you want

The right color palette helps you focus on your code, is more consistent, and is simply more fun to use.

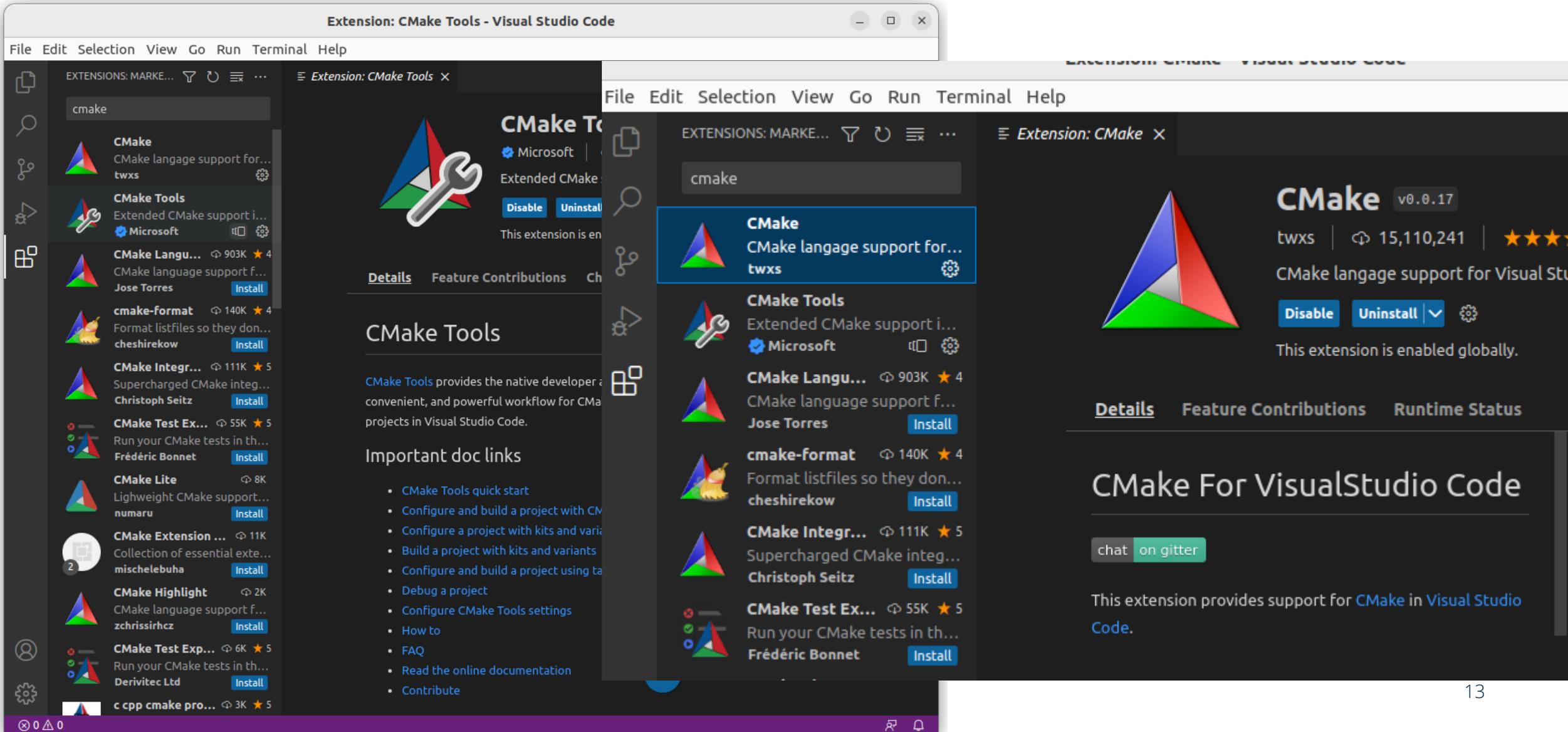
[Browse Color Themes](#)

Tip: Use keyboard shortcut (Ctrl+K Ctrl+T)

- Sync to and from other devices

Code collects usage data. Read our [privacy statement](#) and learn how to [opt out](#).

Install the CMake Tools Extension



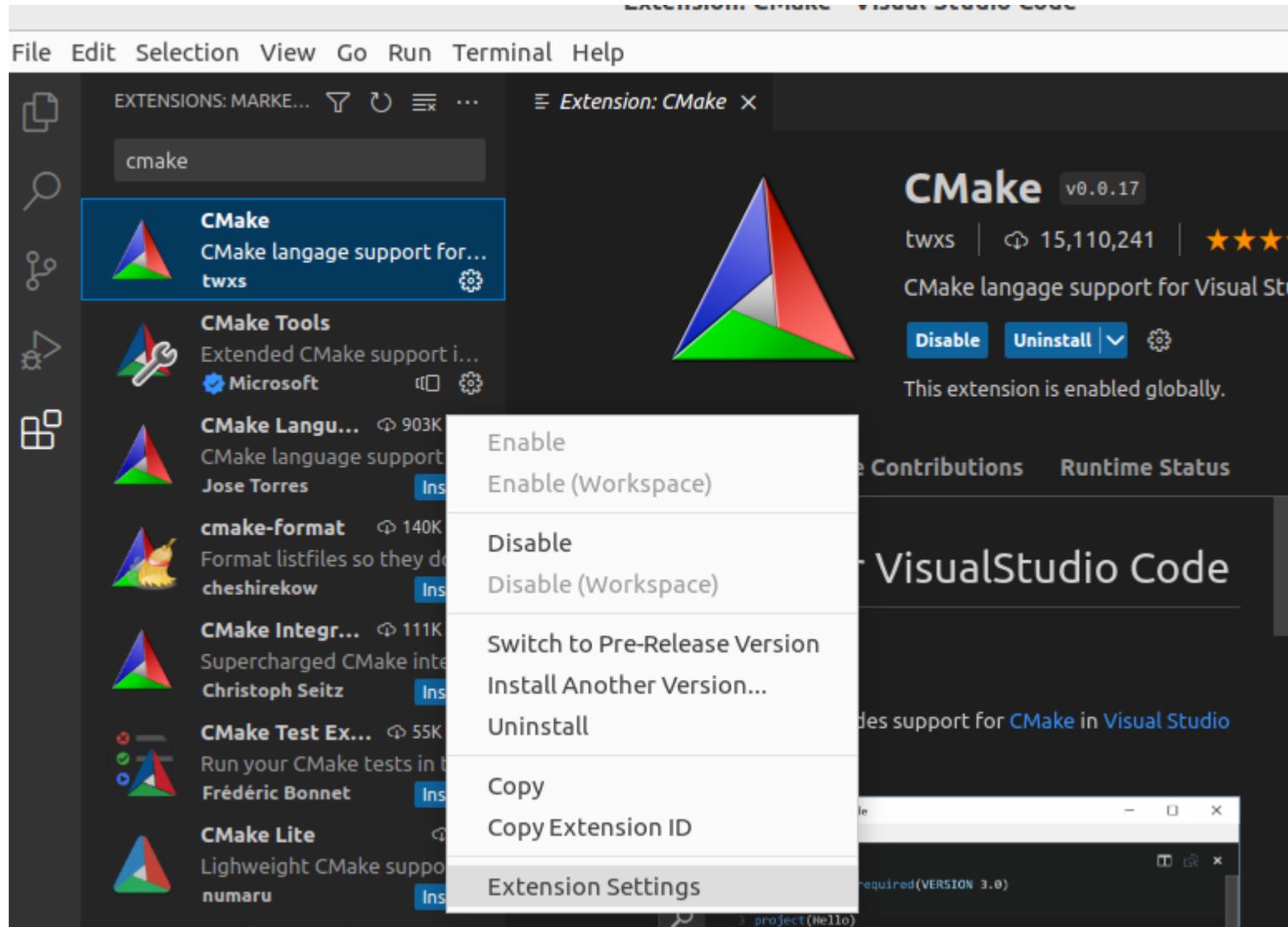
The image shows a sequence of three screenshots from the Visual Studio Code interface, illustrating the process of installing the CMake Tools extension.

Top Screenshot: The 'Extension: CMake Tools - Visual Studio Code' window is open. The search bar contains 'cmake'. The extension list on the left shows several options, including 'CMake' by twxs, 'CMake Tools' by Microsoft, 'CMake Language Support' by Jose Torres, 'cmake-format' by cheshirekow, 'CMake Integr...' by Christoph Seitz, 'CMake Test Ex...' by Frédéric Bonnet, 'CMake Lite' by numaru, 'CMake Extension ...' by mischelebuha, 'CMake Highlight' by zchrissirhcz, and 'CMake Test Exp...' by Derivitec Ltd. The 'CMake Tools' extension by Microsoft is highlighted.

Middle Screenshot: The 'CMake Tools' extension details page is shown. It features the extension's logo (a triangle with a wrench) and the text 'CMake Tools provides the native developer a convenient, and powerful workflow for CMake projects in Visual Studio Code.' Below this, there are 'Important doc links' such as 'CMake Tools quick start', 'Configure and build a project with CMake', 'Configure a project with kits and variants', 'Build a project with kits and variants', 'Configure and build a project using tasks', 'Debug a project', 'Configure CMake Tools settings', 'How to', 'FAQ', 'Read the online documentation', and 'Contribute'.

Bottom Screenshot: The 'CMake' extension details page is shown. It features the extension's logo (a triangle) and the text 'CMake language support for Visual Studio Code'. Below this, there are 'Details', 'Feature Contributions', and 'Runtime Status' tabs. The 'CMake For VisualStudio Code' section includes a 'chat on gitter' button and the text 'This extension provides support for CMake in Visual Studio Code.'

Configure the CMake Tools Extension

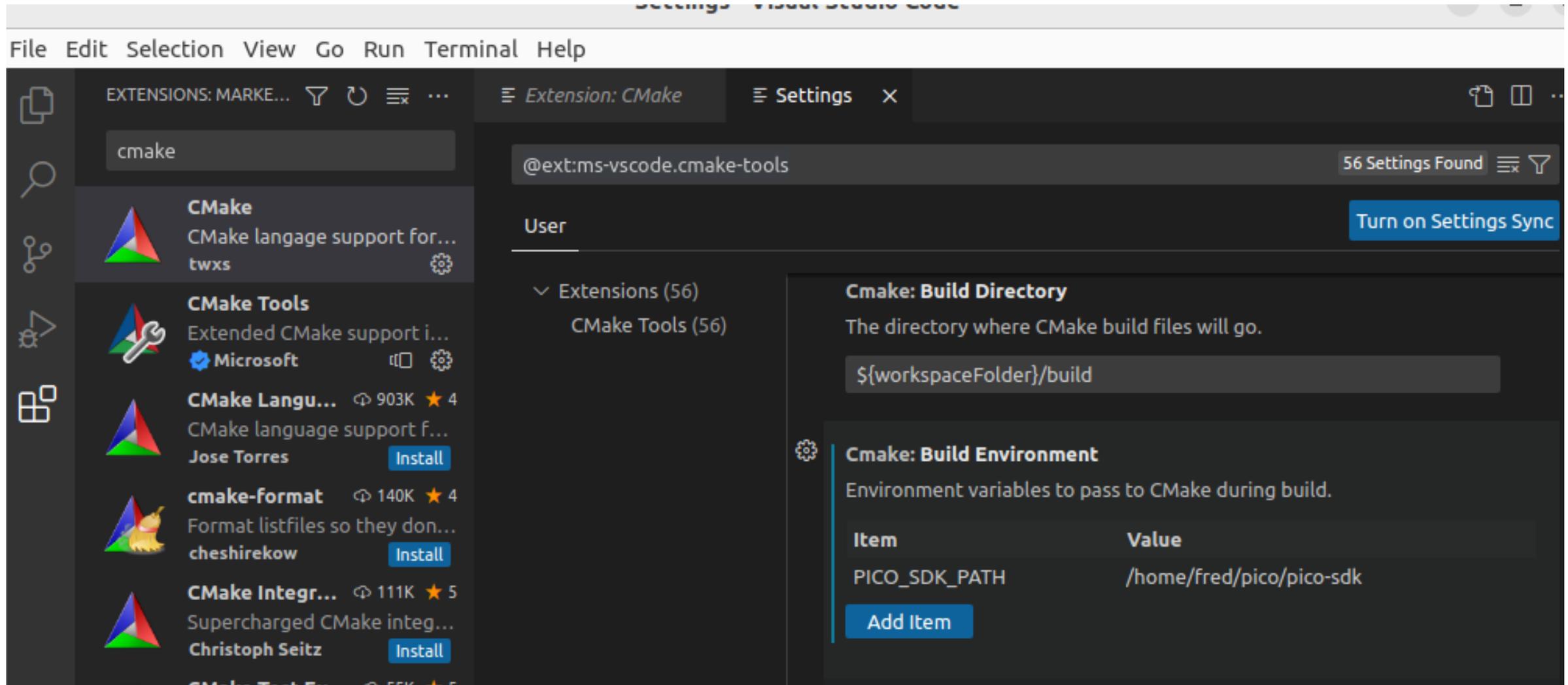


The screenshot shows the Visual Studio Code interface with the Extensions view open. The search bar contains "cmake". The "CMake" extension by twxs is selected, and its settings menu is open. The menu options are:

- Enable
- Enable (Workspace)
- Disable
- Disable (Workspace)
- Switch to Pre-Release Version
- Install Another Version...
- Uninstall
- Copy
- Copy Extension ID
- Extension Settings

The background shows the extension details for "CMake v0.0.17" by twxs, which is currently disabled. The extension description is "CMake language support for Visual Studio Code".

Configure the SDK Path

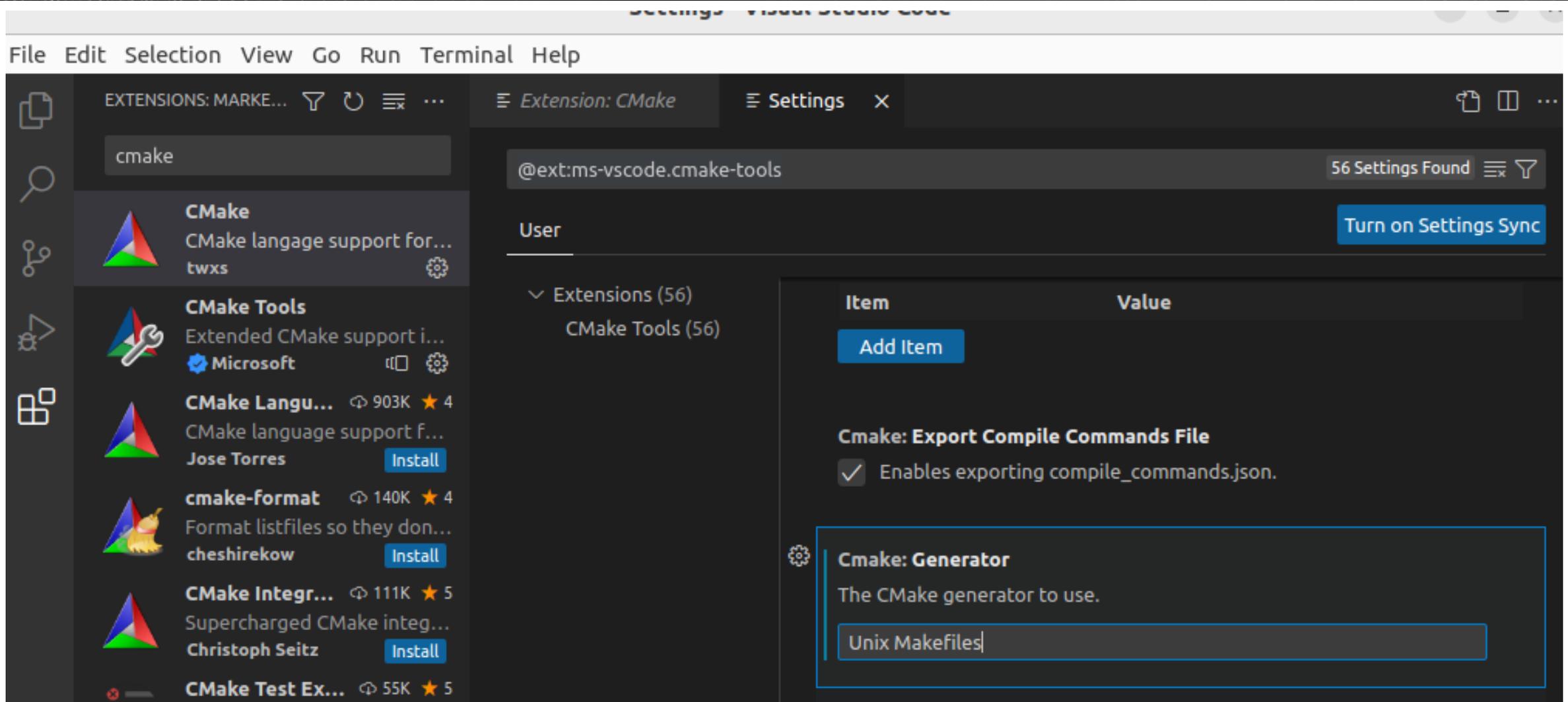


The screenshot shows the Visual Studio Code interface with the Settings window open for the CMake Tools extension. The search bar contains 'cmake' and 56 settings are found. The 'Cmake: Build Environment' setting is expanded, showing a table of environment variables.

Item	Value
PICO_SDK_PATH	/home/fred/pico/pico-sdk

Other visible settings include 'Cmake: Build Directory' set to `${workspaceFolder}/build`.

Specify Unix Makefiles



The screenshot shows the Visual Studio Code interface with the CMake Tools extension settings open. The left sidebar shows the extension search results for 'cmake'. The main settings pane shows the 'Cmake: Generator' setting, which is currently set to 'Unix Makefiles'.

File Edit Selection View Go Run Terminal Help

EXTENSIONS: MARKE... 🔍 ↻ ☰ ...

cmake

CMake
CMake language support for...
twxs

CMake Tools
Extended CMake support i...
Microsoft

CMake Langu... 903K ★ 4
CMake language support f...
Jose Torres **Install**

cmake-format 140K ★ 4
Format listfiles so they don...
cheshirekow **Install**

CMake Integr... 111K ★ 5
Supercharged CMake integ...
Christoph Seitz **Install**

CMake Test Ex... 55K ★ 5

@ext:ms-vscode.cmake-tools 56 Settings Found

Turn on Settings Sync

Item	Value
Add Item	
Cmake: Export Compile Commands File	<input checked="" type="checkbox"/> Enables exporting compile_commands.json.
Cmake: Generator	The CMake generator to use. <input type="text" value="Unix Makefiles"/>

Discover and Select the Desired GCC Toolchain

● main.c - blink - Visual Studio Code

Select a Kit for blink

[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 = /usr/bin/arm-none-eabi-gcc, CXX = /usr/bin/arm-non...

GCC 11.3.0 x86_64-linux-gnu Using compilers: C = /usr/bin/gcc, CXX = /usr/bin/g++



> OUTLINE

> TIMELINE



CMake: [Debug]: Ready



No Kit Selected



Build

[all]



Run CTest



> OUTLINE

> TIMELINE



CMake: [Debug]: Ready



[GCC 10.3.1 arm-none-eabi]



Build

[all]



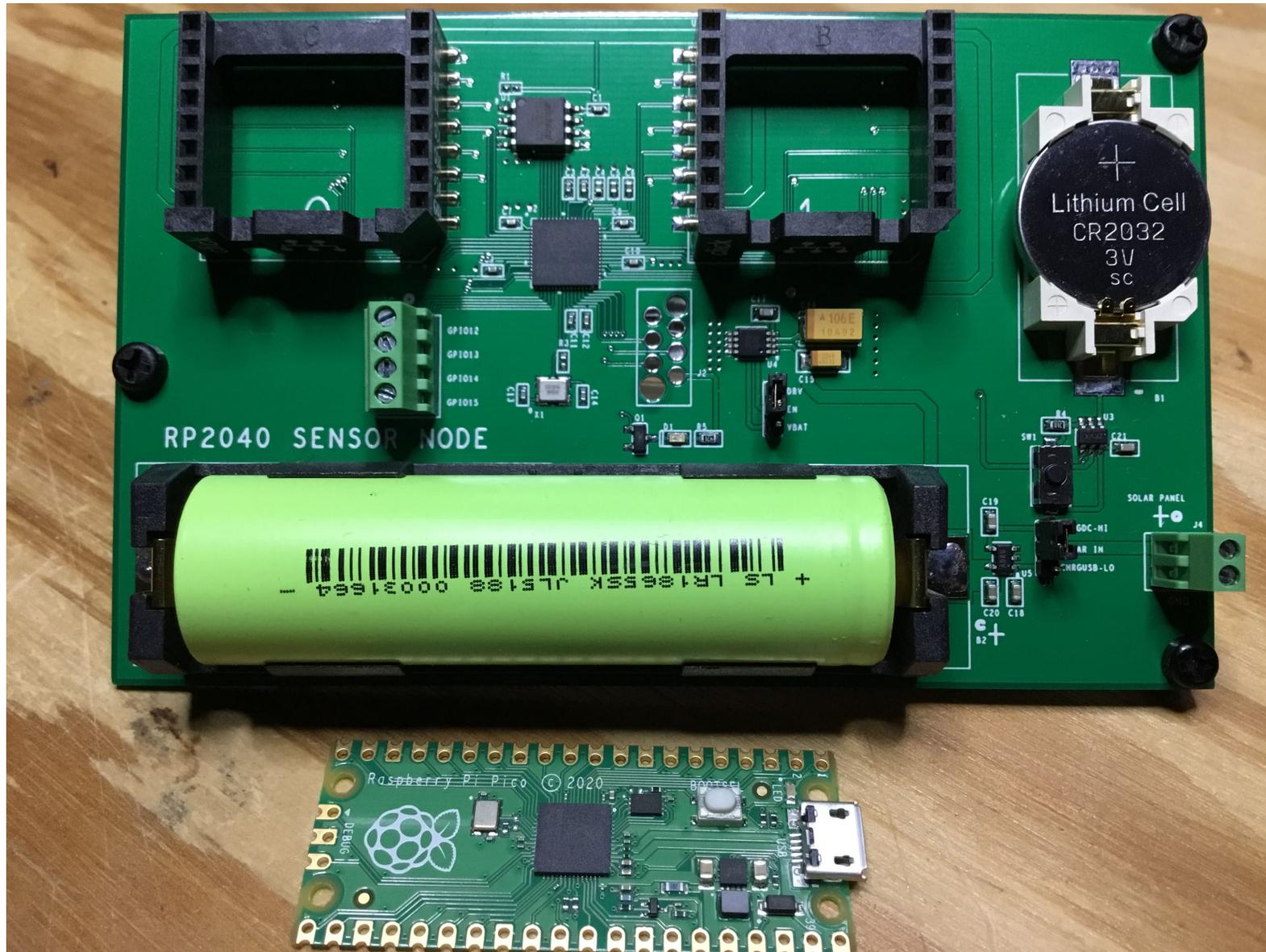
Run CTest

Designate the SDK Paths

```
Open  [icon] .bashrc Save [icon] [icon] [icon] [icon]
86
87 # colored GCC warnings and errors
88 #export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:quote=01'
89
90 # some more ls aliases
91 alias ll='ls -alF'
92 alias la='ls -A'
93 alias l='ls -CF'
94
95 # Add an "alert" alias for long running commands.  Use like so:
96 #   sleep 10; alert
97 alias alert='notify-send --urgency=low -i "${[ $? = 0 ]} && echo terminal || echo error)" "$
    (history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[\;]&]\s*alert$//'\`)"'
98
99 # Alias definitions.
100 # You may want to put all your additions into a separate file like
101 # ~/.bash_aliases, instead of adding them here directly.
102 # See /usr/share/doc/bash-doc/examples in the bash-doc package.
103
104 if [ -f ~/.bash_aliases ]; then
105     . ~/.bash_aliases
106 fi
107
108 # enable programmable completion features (you don't need to enable
109 # this, if it's already enabled in /etc/bash.bashrc and /etc/profile
110 # sources /etc/bash.bashrc).
111 if ! shopt -oq posix; then
112     if [ -f /usr/share/bash-completion/bash_completion ]; then
113         . /usr/share/bash-completion/bash_completion
114     elif [ -f /etc/bash_completion ]; then
115         . /etc/bash_completion
116     fi
117 fi
118
119 export PICO_SDK_PATH=/home/fred/pico/pico-sdk
120 export PICO_EXAMPLES_PATH=/home/fred/pico/pico-examples
121 export PICO_TOOLCHAIN_PATH=/home/fred/usr/bin

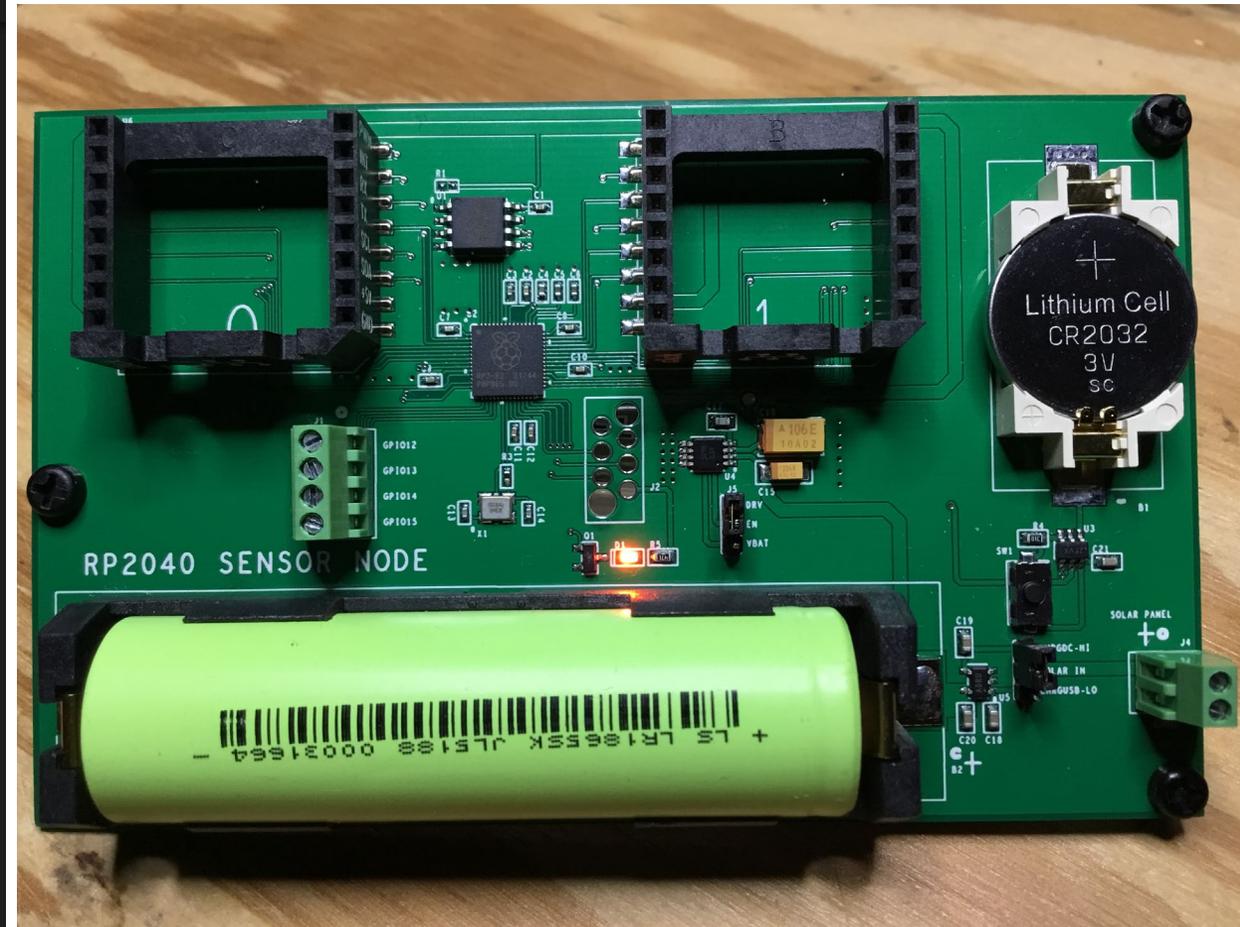
sh  Tab Width: 8  Ln 1, Col 1  INS
```

Known-Good Hardware



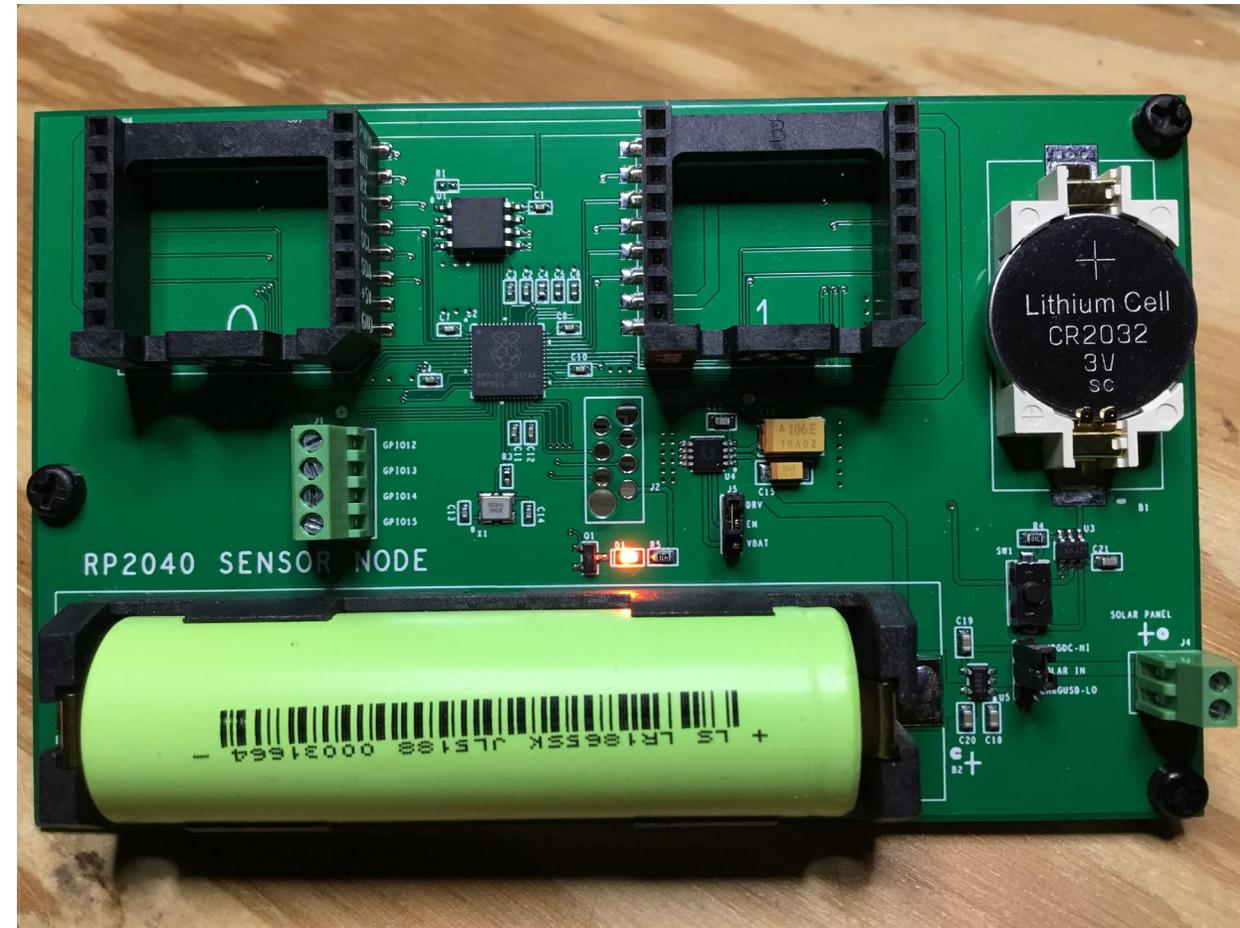
Test Flight Code – CMakeLists.txt

```
M CMakeLists.txt x C main.c
M CMakeLists.txt
4 # Include build functions from Pico SDK
5 include($ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake)
6
7 # Set name of project (as PROJECT_NAME) and C/C++ standards
8 project(blink C CXX ASM)
9 set(CMAKE_C_STANDARD 11)
10 set(CMAKE_CXX_STANDARD 17)
11
12 # Creates a pico-sdk subdirectory in our project for the libraries
13 pico_sdk_init()
14
15 # Tell CMake where to find the executable source file
16 add_executable(${PROJECT_NAME}
17 |   main.c
18 | )
19
20 # Create map/bin/hex/uf2 files
21 pico_add_extra_outputs(${PROJECT_NAME})
22
23 # Link to pico_stdlib (gpio, time, etc. functions)
24 target_link_libraries(${PROJECT_NAME}
25 |   pico_stdlib
26 | )
27
28 # Enable usb output, enable uart output
29 pico_enable_stdio_usb(${PROJECT_NAME} 0)
30 pico_enable_stdio_uart(${PROJECT_NAME} 1)
```

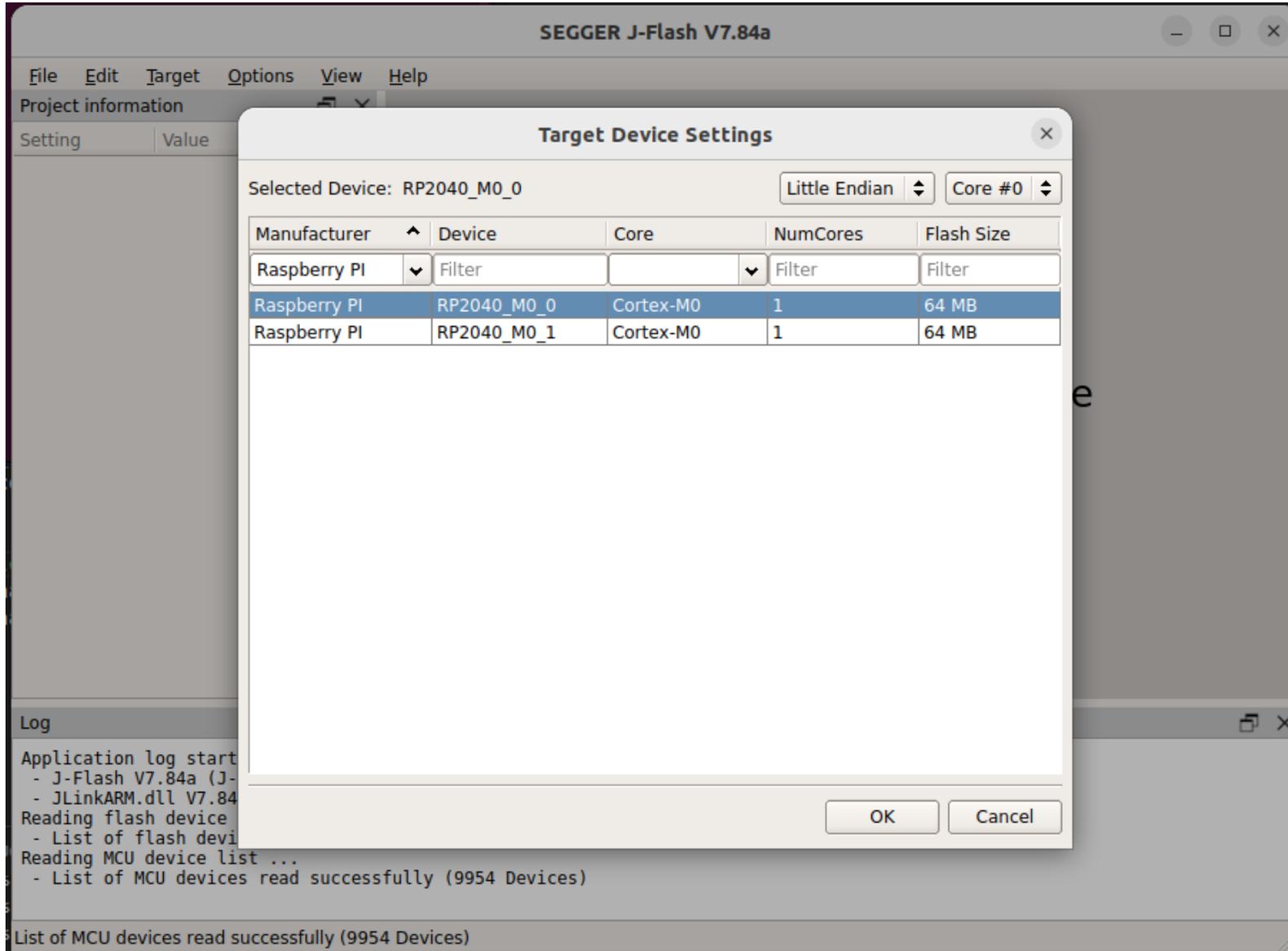


Test Flight Code - main.c

```
CMakeLists.txt  C main.c  X
C main.c
1  #include <stdio.h>
2  #include "pico/stdlib.h"
3
4  int main() {
5
6      const uint led_pin = 17;
7      const uint tx0_pin = 0;
8      const uint rx0_pin = 1;
9
10     // Initialize LED pin
11     gpio_init(led_pin);
12     gpio_set_dir(led_pin, GPIO_OUT);
13     // Initialize chosen serial port
14     stdio_uart_init_full(uart0, 115200, tx0_pin, rx0_pin);
15
16     // Loop forever
17     while (true)
18     {
19
20         // Blink LED
21         printf("RP2040 Low-Power Sensor Node Test Flight\r\n");
22         gpio_put(led_pin, true);
23         sleep_ms(500);
24         gpio_put(led_pin, false);
25         sleep_ms(500);
26     }
27 }
```



Fuel Up

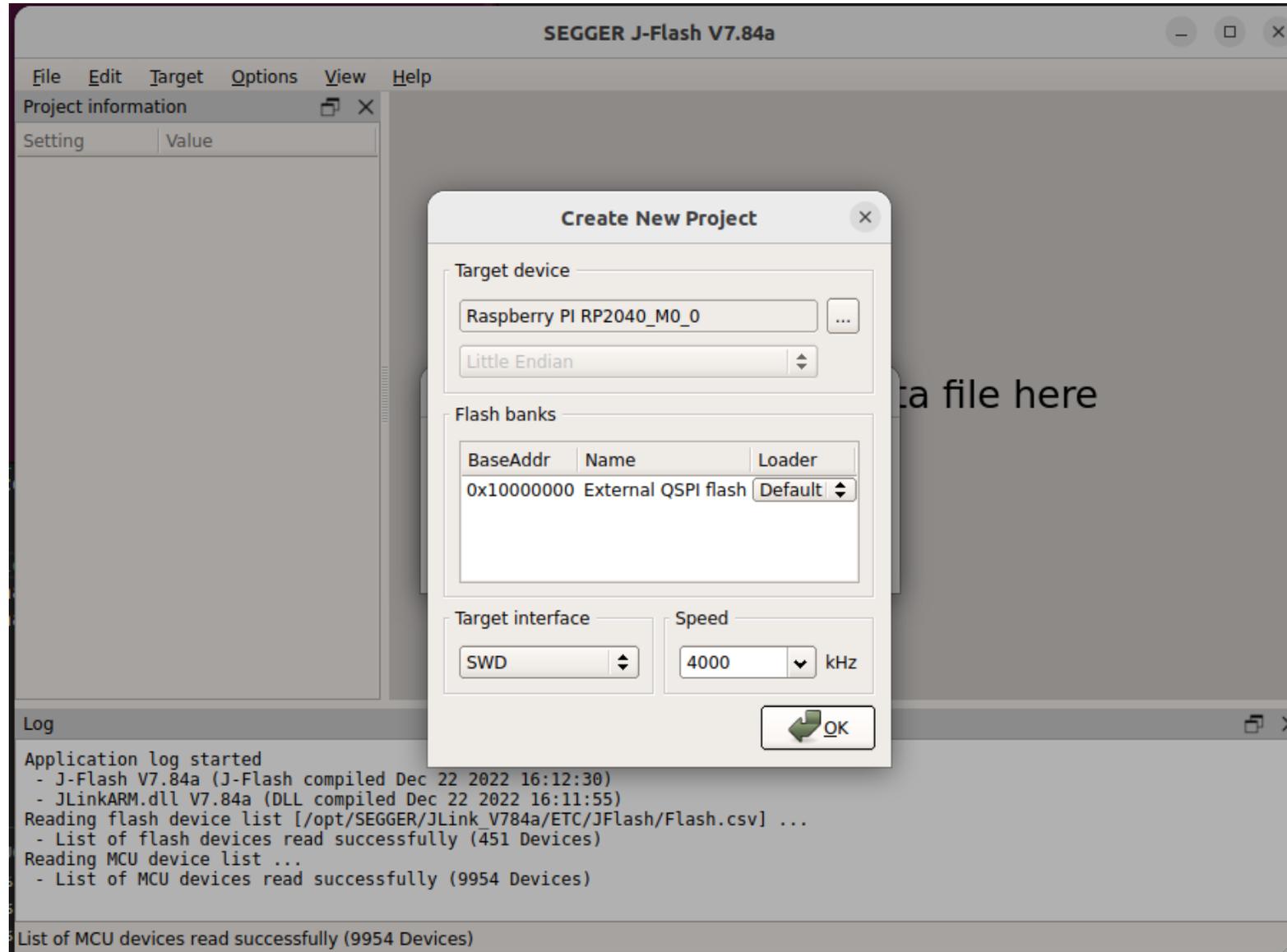


The screenshot shows the SEGGER J-Flash V7.84a application window. A 'Target Device Settings' dialog box is open, displaying a table of available devices. The 'Selected Device' is set to 'RP2040_M0_0'. The table lists two Raspberry Pi RP2040 devices, both with Cortex-M0 cores and 64 MB of flash memory. The background shows the main application interface with a menu bar (File, Edit, Target, Options, View, Help) and a log window at the bottom.

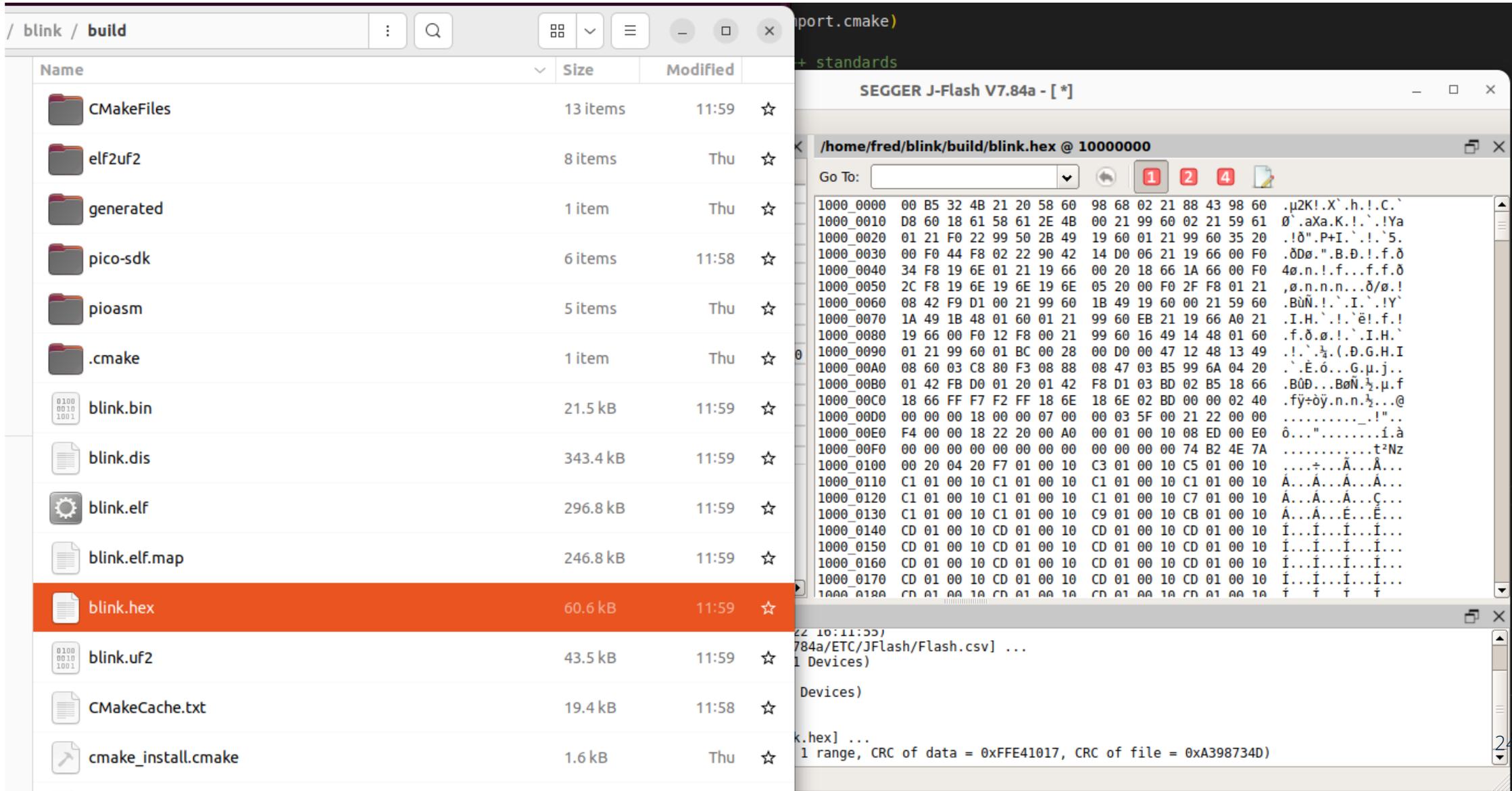
Manufacturer	Device	Core	NumCores	Flash Size
Raspberry PI	Filter		Filter	Filter
Raspberry PI	RP2040_M0_0	Cortex-M0	1	64 MB
Raspberry PI	RP2040_M0_1	Cortex-M0	1	64 MB

Log
Application log start
- J-Flash V7.84a (J-
- JLinkARM.dll V7.84
Reading flash device
- List of flash devi
Reading MCU device list ...
- List of MCU devices read successfully (9954 Devices)
List of MCU devices read successfully (9954 Devices)

Engine Start



Takeoff Roll



The image shows a file manager window displaying the contents of the `/blink/build` directory. The file `blink.hex` is selected and highlighted in orange. The file manager shows the following files and folders:

Name	Size	Modified
CMakeFiles	13 items	11:59
elf2uf2	8 items	Thu
generated	1 item	Thu
pico-sdk	6 items	11:58
picoasm	5 items	Thu
.cmake	1 item	Thu
blink.bin	21.5 kB	11:59
blink.dis	343.4 kB	11:59
blink.elf	296.8 kB	11:59
blink.elf.map	246.8 kB	11:59
blink.hex	60.6 kB	11:59
blink.uf2	43.5 kB	11:59
CMakeCache.txt	19.4 kB	11:58
cmake_install.cmake	1.6 kB	Thu

The hex editor window, titled "SEGGER J-Flash V7.84a - [*]", shows the contents of `/home/fred/blink/build/blink.hex @ 10000000`. The editor displays hexadecimal data in columns, with the first column showing the address (e.g., `1000_0000`), the second column showing the hexadecimal value (e.g., `00 B5 32 4B 21 20 58 60`), and the third column showing the corresponding ASCII characters (e.g., `.µ2K!.X`.h.!.C.``). The editor also includes a "Go To:" field and several numbered buttons (1, 2, 4).

The bottom of the hex editor window shows a status bar with the following text:

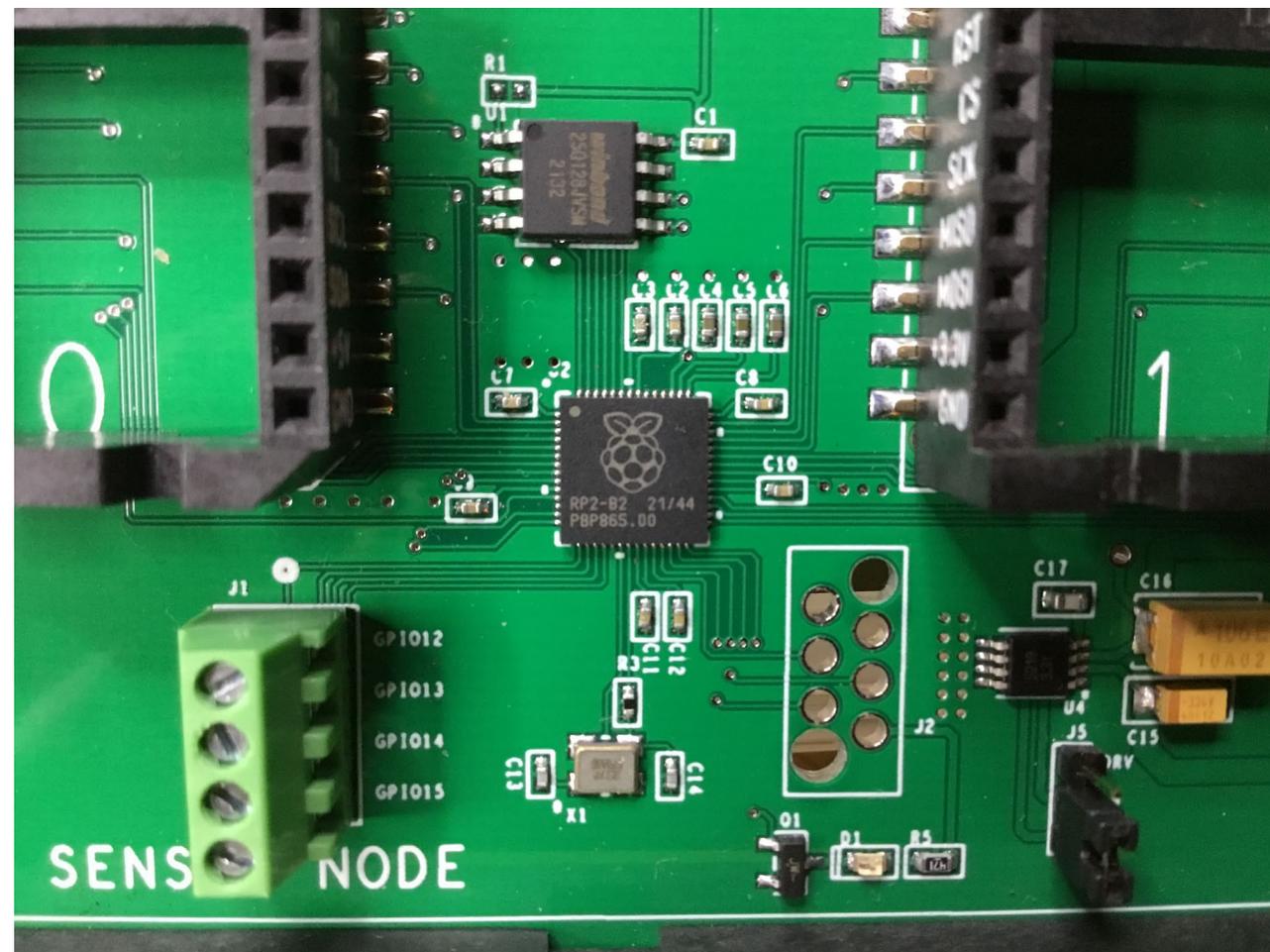
```
22:16:11:55) ...  
784a/ETC/JFlash/Flash.csv] ...  
1 Devices)  
Devices)  
k.hex] ...  
1 range, CRC of data = 0xFFE41017, CRC of file = 0xA398734D)
```


Thank you for attending!!!

Please consider the resources below:

- raspberrypi.org
- [RP2040 Data Sheet](#)
- [Raspberry Pi Pico C/C++ SDK](#)

MORE TO COME..





DesignNews

Thank You

Sponsored by

