

Raspberry Pi Pico/Pico2 Development Using Visual Studio Code

Day 1:

The Raspberry Pi Pico VS Code Extension

Sponsored by

DigiKey

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.

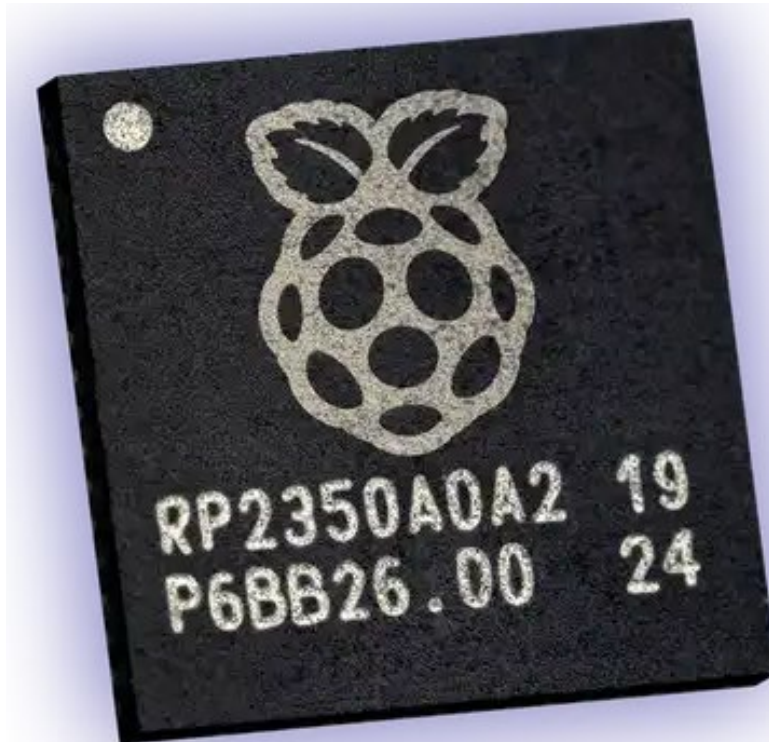


Fred Eady

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

AGENDA

- **Install Visual Studio Code**
- **Install the Raspberry Pi Pico VS Code Extension**
- **Equipment Check**



Download and Install VS Code

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

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

↓ Windows

Windows 10, 11

User Installer	x64	Arm64
System Installer	x64	Arm64
.zip	x64	Arm64
CLI	x64	Arm64



↓ .deb

Debian, Ubuntu

↓ .rpm

Red Hat, Fedora, SUSE

→ .deb	x64	Arm32	Arm64
.rpm	x64	Arm32	Arm64
.tar.gz	x64	Arm32	Arm64
Snap	Snap Store		
CLI	x64	Arm32	Arm64

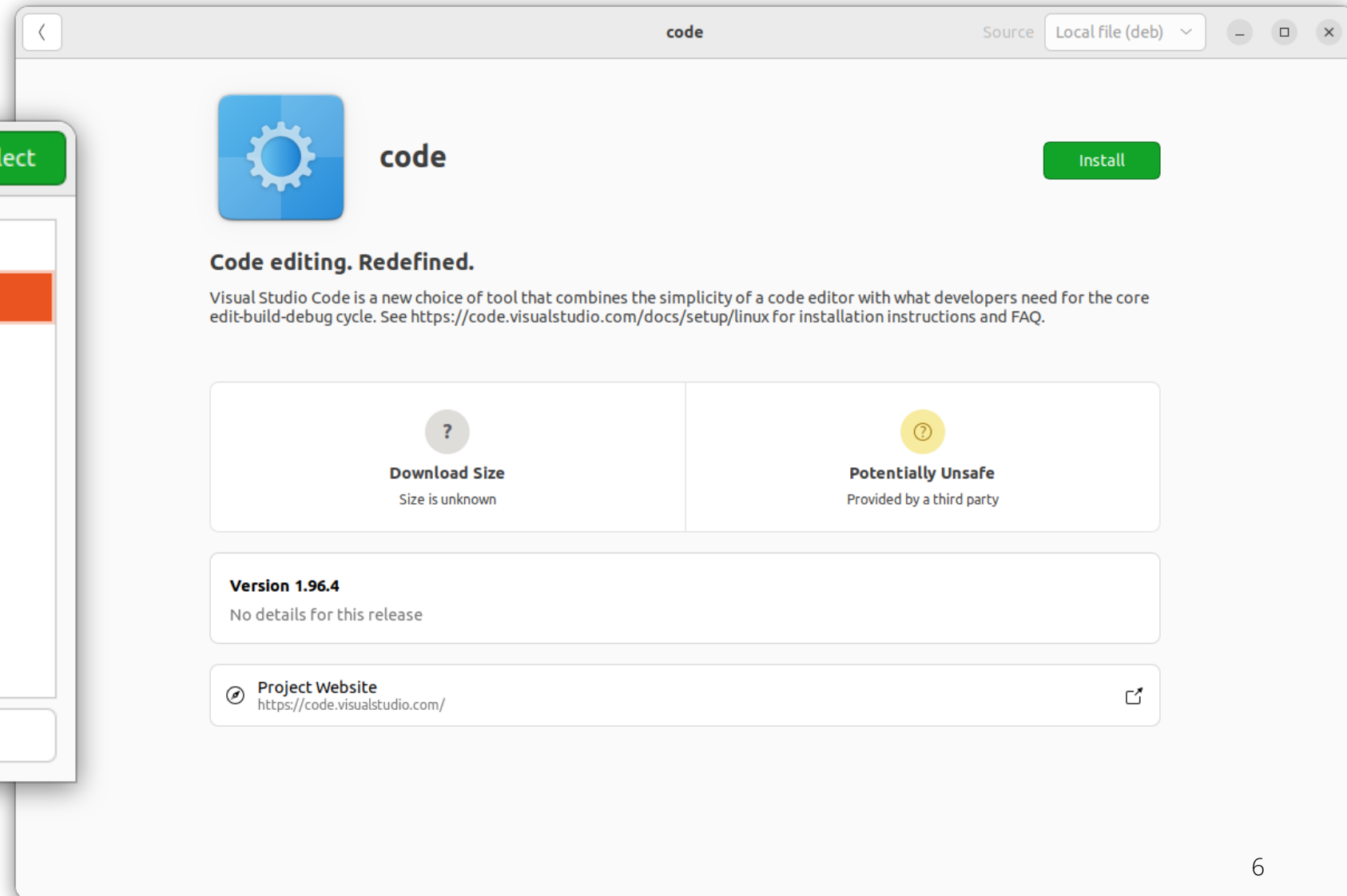
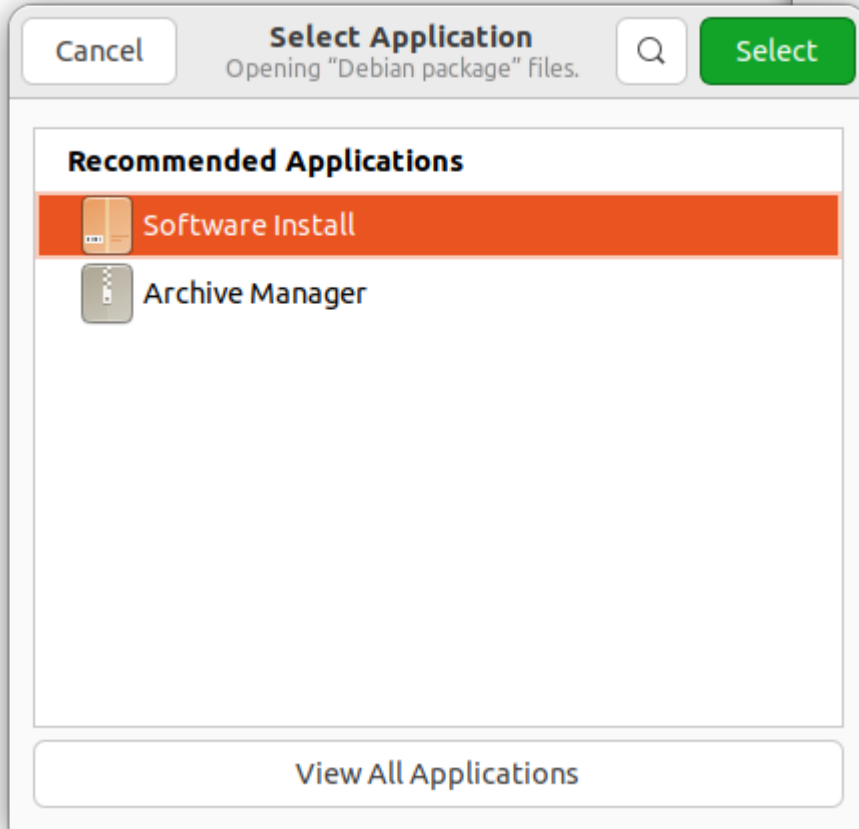


↓ Mac

macOS 10.15+

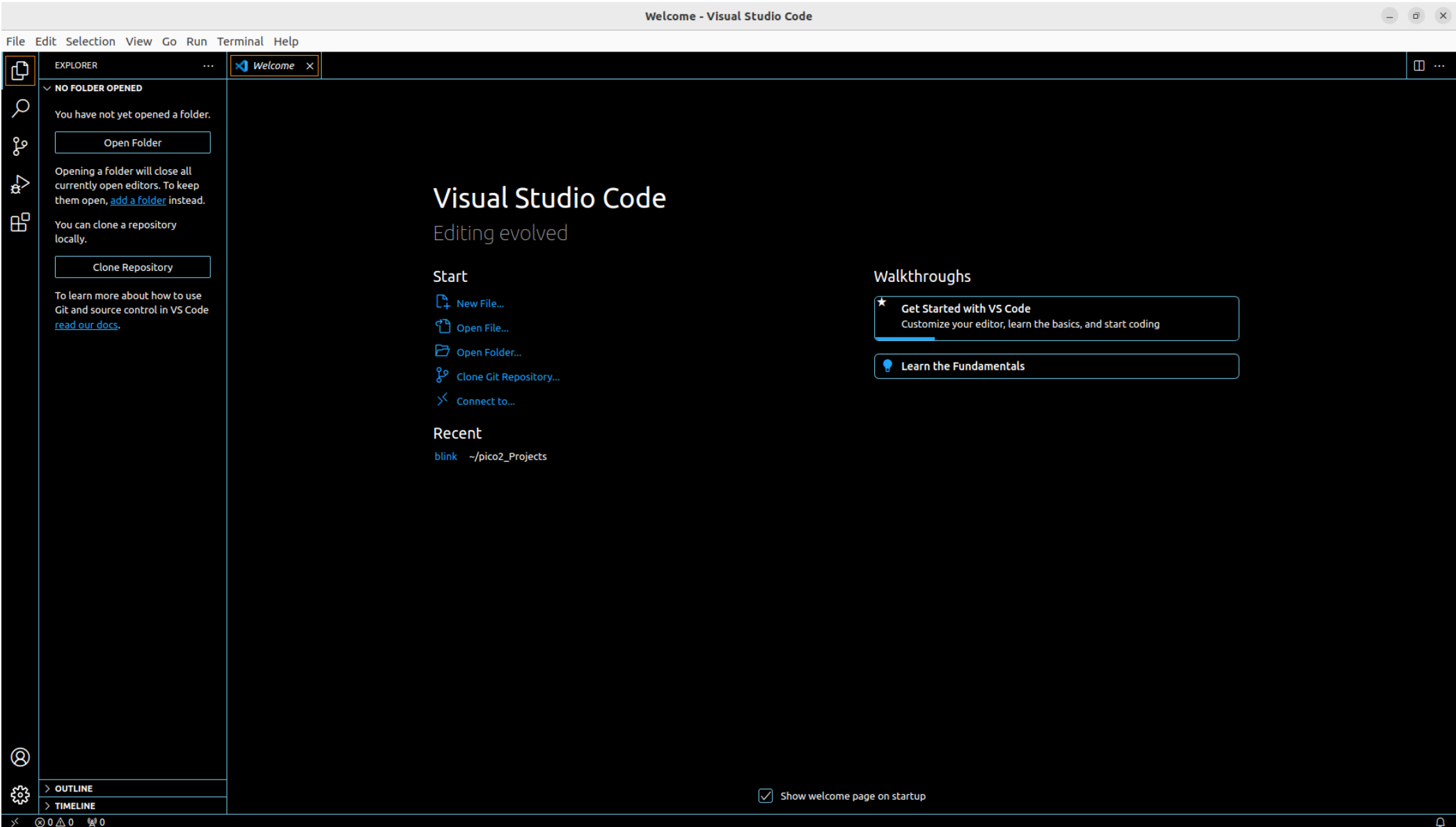
.zip	Intel chip	Apple silicon	Universal
CLI	Intel chip	Apple silicon	

Download and Install VS Code

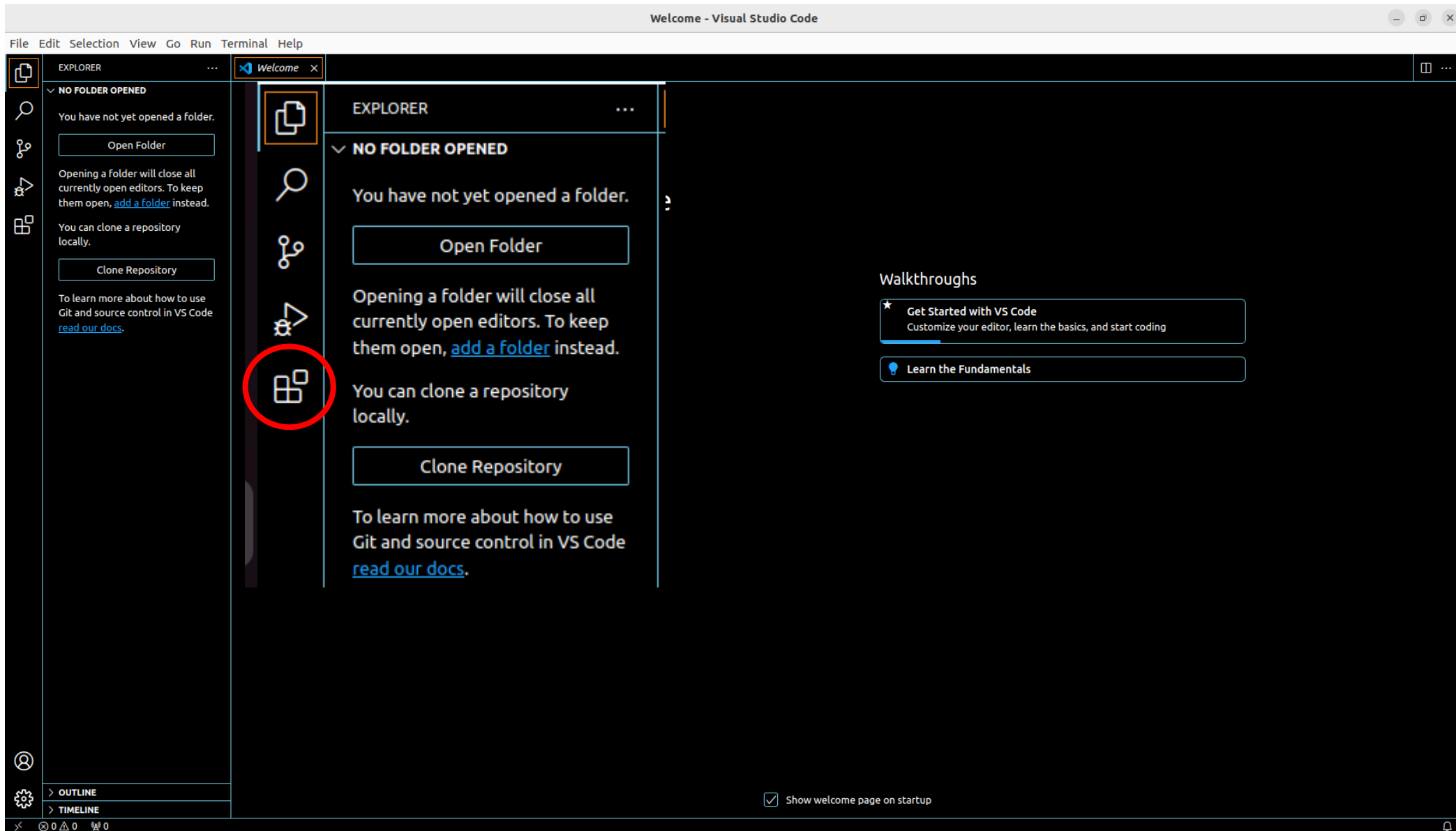




Download and Install VS Code



Install the Pico VS Code Extension



Install the Pico VS Code Extension

The screenshot shows the Visual Studio Code interface with the Extensions Marketplace open. The search bar contains the text "pico". The "Raspberry Pi Pico" extension is highlighted with a red arrow. The "Install" button for this extension is circled in red. The extension details for "Raspberry Pi Pico" are shown in the main panel, including the description: "The official VS Code extension for Raspbe". The interface also shows other extensions like "Pico-8 Theme", "Picopilot", "pico8-ls", "Picoblaze assembly", "AHS PicoRun", "MicroPico", "pico8vscodeeditor", "picoG Firmware Development", "pico8-theme", "pico8-lang", and "pico8-vscode".

Install the Pico VS Code Extension

The screenshot shows the Visual Studio Code interface with the 'Extension: Raspberry Pi Pico - Visual Studio Code' window open. The interface is divided into several sections:

- Left Panel (EXTENSIONS):** A list of installed and available extensions. The 'Raspberry Pi Pico' extension is highlighted with a red circle. Other extensions include C/C++, C/C++ Extension Pack, C/C++ Themes, CMake, CMake Tools, Cortex-Debug, debug-tracker-vscode, MemoryView, MicroPico, Peripheral Viewer, and Pylance.
- Center Panel (Raspberry Pi Pico Visual Studio Code):** The details page for the extension, showing a note that it is currently under development. It lists features such as Project Setup and Management, Configuration and Tool Management, and Build, Debug, and Documentation.
- Right Panel (EXTENSIONS):** A search bar and a list of installed extensions. The 'Raspberry Pi Pico' extension is highlighted with a red circle. Other installed extensions include C/C++, C/C++ IntelliSense, debuggin..., C/C++ Extension Pack, and C/C++ Themes.

The search bar in the right panel is highlighted with a yellow box, and the search results are also highlighted with a yellow box. The 'Raspberry Pi Pico' extension is highlighted with a red circle in both the left and right panels.

Load the *blink* Example

Raspberry Pi **Pico**

Basic Settings

Debugger

RASPBERY PI PICO PROJECT: QUICK ACCESS

General

- New C/C++ Project
- New MicroPython Project
- Import Project
- New Project From Example

Project

- Debug Project
- Compile Project
- Run Project (USB)
- Flash Project (SWD)
- Configure CMake
- Clean CMake
- Switch SDK Current: N/A
- Switch Board
- Debug Layout

Documentation

- Hardware APIs
- High Level APIs
- Networking Libraries
- Runtime Infrastructure

Basic Settings

Name

blink

Board type

Pico

Pico 2

Pico

Pico W

Pico 2 W

Other

Debugger

DebugProbe (CMSIS-DAP)
[Default]

SWD (Pi host, on Pi 5 it requires Linux Kernel >= 6.6.47)

Show Advanced Options

Cancel

Create

Compile the *blink* Example

The screenshot shows the Visual Studio Code interface with the `blink.c` file open in the editor. The Explorer sidebar shows the project structure, including `.vscode`, `build`, `.gitignore`, `blink.c`, `CMakeLists.txt`, and `pico_sdk_import.cmake`. The terminal window displays the following output:

```
Executing task: /home/fred/.pico-sdk/ninja/v1.12.1/ninja -C /home/fred/pico2_Projects/blink/build
ninja: Entering directory `/home/fred/pico2_Projects/blink/build'
[2/2] Linking CXX executable blink.elf
* Terminal will be reused by tasks, press any key to close it.
```

The screenshot shows the Visual Studio Code interface with the `blink.c` file open in the editor. The terminal window displays the following output:

```
Executing task: /home/fred/.pico-sdk/ninja/v1.12.1/ninja -C /home/fred/pico2_Projects/blink/build
ninja: Entering directory `/home/fred/pico2_Projects/blink/build'
[2/2] Linking CXX executable blink.elf
* Terminal will be reused by tasks, press any key to close it.
```

The `Compile` button in the bottom right corner of the editor is highlighted with a red box.



Run the *blink* Example

The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER:** Shows the project structure with files like `.vscode`, `build`, `.gitignore`, `blink.c`, and `CMakeLists.txt`.
- EDITOR:** Displays the `blink.c` file with a copyright notice: `Copyright (c) 2020 Raspberry Pi (Trading) Ltd.`
- TERMINAL:** Shows the execution of the command `picotool load /home/fred/pico2_Projects/blink/build/blink.elf -fx`. The output indicates that no RP-series devices were found in BOOTSEL mode. A yellow box highlights the error message: "No accessible RP-series devices in BOOTSEL mode were found. but: RP2040 device at bus 1, address 43 appears to be in BOOTSEL mode, but picotool was unable to connect. Maybe try 'sudo' or check your permissions." Below this, a message states: "The terminal process '/home/fred/.pico-sdk/picotool/2.1.0/picotool/picotool 'load', '/home/fred/pico2_Projects/blink/build/blink.elf', '-fx' failed to launch (exit code: 249)." A red box highlights the **Run** button in the terminal toolbar.
- STATUS BAR:** Shows "Ln 9, Col 1 Spaces: 4 UTF-8 LF () C Compile Run Pico SDK: 2.1.0 Board: pico Pico".
- OUTPUT:** Shows the output of the `ninja` build process: `ninja: Entering directory '/home/fred/pico2_Projects/blink/build'` and `[2/2] Linking CXX executable blink.elf`.

Sense and Identify the Picos

```
fred@shop-ubuntu-1770: ~  
fred@shop-ubuntu-1770:~$ lsusb  
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub  
Bus 001 Device 005: ID 0403:6001 Future Technology Devices International, Ltd FT232 Serial (UART) IC  
Bus 001 Device 040: ID 2e8a:000f Raspberry Pi RP2350 Boot  
Bus 001 Device 041: ID 2e8a:000f Raspberry Pi RP2350 Boot  
Bus 001 Device 004: ID 14cd:8601 Super Top 4-Port hub  
Bus 001 Device 038: ID 2e8a:0003 Raspberry Pi RP2 Boot  
Bus 001 Device 037: ID 1a40:0101 Terminus Technology Inc. Hub  
Bus 001 Device 003: ID 046d:c52b Logitech, Inc. Unifying Receiver  
Bus 001 Device 039: ID 2e8a:0003 Raspberry Pi RP2 Boot  
Bus 001 Device 002: ID 05e3:0746 Genesys Logic, Inc. USB Storage  
Bus 001 Device 006: ID 8087:0033 Intel Corp.  
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub  
fred@shop-ubuntu-1770:~$
```

Download and Place *99-picotool.rules*

The screenshot shows the GitHub interface for the 'picotool' repository. The repository is public and has 13 issues, 11 pull requests, and 24 actions. The file browser on the left shows the directory structure, with '99-picotool.rules' selected under the 'udev' folder. The main content area displays the code for '99-picotool.rules', which is a USB device descriptor file. The code is as follows:

```
1 SUBSYSTEM=="usb", \  
2     ATTRS{idVendor}=="2e8a", \  
3     ATTRS{idProduct}=="0003", \  
4     TAG+="uaccess" \  
5     MODE="660", \  
6     GROUP="plugdev" \  
7 SUBSYSTEM=="usb", \  
8     ATTRS{idVendor}=="2e8a", \  
9     ATTRS{idProduct}=="0009", \  
10    TAG+="uaccess" \  
11    MODE="660", \  
12    GROUP="plugdev" \  
13 SUBSYSTEM=="usb", \  
14    ATTRS{idVendor}=="2e8a", \  
15    ATTRS{idProduct}=="000a", \  
16    TAG+="uaccess" \  
17    MODE="660", \  
18    GROUP="plugdev" \  
19 SUBSYSTEM=="usb", \  
20    ATTRS{idVendor}=="2e8a", \  
21    ATTRS{idProduct}=="000f", \  
22    TAG+="uaccess" \  
23    MODE="660", \  
24    GROUP="plugdev"
```

Download and Place *99-picotool.rules*

```
fred@shop-ubuntu-1770: /etc/udev/rules.d
File Edit View Search Terminal Help

fred@shop-ubuntu-1770:/etc/udev/rules.d$ ls
49-stlinkv1.rules      50-usb-conf.rules      70-snap.snap-store.rules  Readme.txt
49-stlinkv2-1.rules   70-snap.firefox.rules  70-uuu.rules              version.txt
49-stlinkv2.rules     70-snap.snapd-desktop-integration.rules  90-cyusb.rules
49-stlinkv3.rules     70-snap.snapd.rules    99-jlink.rules
fred@shop-ubuntu-1770:/etc/udev/rules.d$ sudo cp ../99-picotool.rules 99-picotool.rules
fred@shop-ubuntu-1770:/etc/udev/rules.d$ ls
49-stlinkv1.rules      70-snap.firefox.rules  90-cyusb.rules
49-stlinkv2-1.rules   70-snap.snapd-desktop-integration.rules  99-jlink.rules
49-stlinkv2.rules     70-snap.snapd.rules    99-picotool.rules ←
49-stlinkv3.rules     70-snap.snap-store.rules  Readme.txt
50-usb-conf.rules     70-uuu.rules           version.txt
fred@shop-ubuntu-1770:/etc/udev/rules.d$ sudo udevadm trigger
[sudo] password for fred:
fred@shop-ubuntu-1770:/etc/udev/rules.d$
```



Run the *blink* Example.. Again

The screenshot shows Visual Studio Code with the `blink.c` file open. The terminal displays the following output:

```
C blink.c > ...
1  /**
2   * Copyright (c) 2020 Raspberry Pi (Trading) Ltd.
3   *
4   * SPDX-License-Identifier: BSD-3-Clause
5   */
```

The terminal shows a failed attempt to load the application:

```
* The terminal process "/home/fred/.pico-sdk/picotool/2.1.0/picotool/picotool 'load', '/home/fred/pico2_Projects/blink/build/blink.elf', '-fx'" failed to launch (exit code: 249).
* Terminal will be reused by tasks, press any key to close it.
```

Then, the application is successfully loaded and executed:

```
* Executing task: /home/fred/.pico-sdk/picotool/2.1.0/picotool/picotool load /home/fred/pico2_Projects/blink/build/blink.elf -fx
Loading into Flash: [=====] 100%
The device was rebooted to start the application.
* Terminal will be reused by tasks, press any key to close it.
```

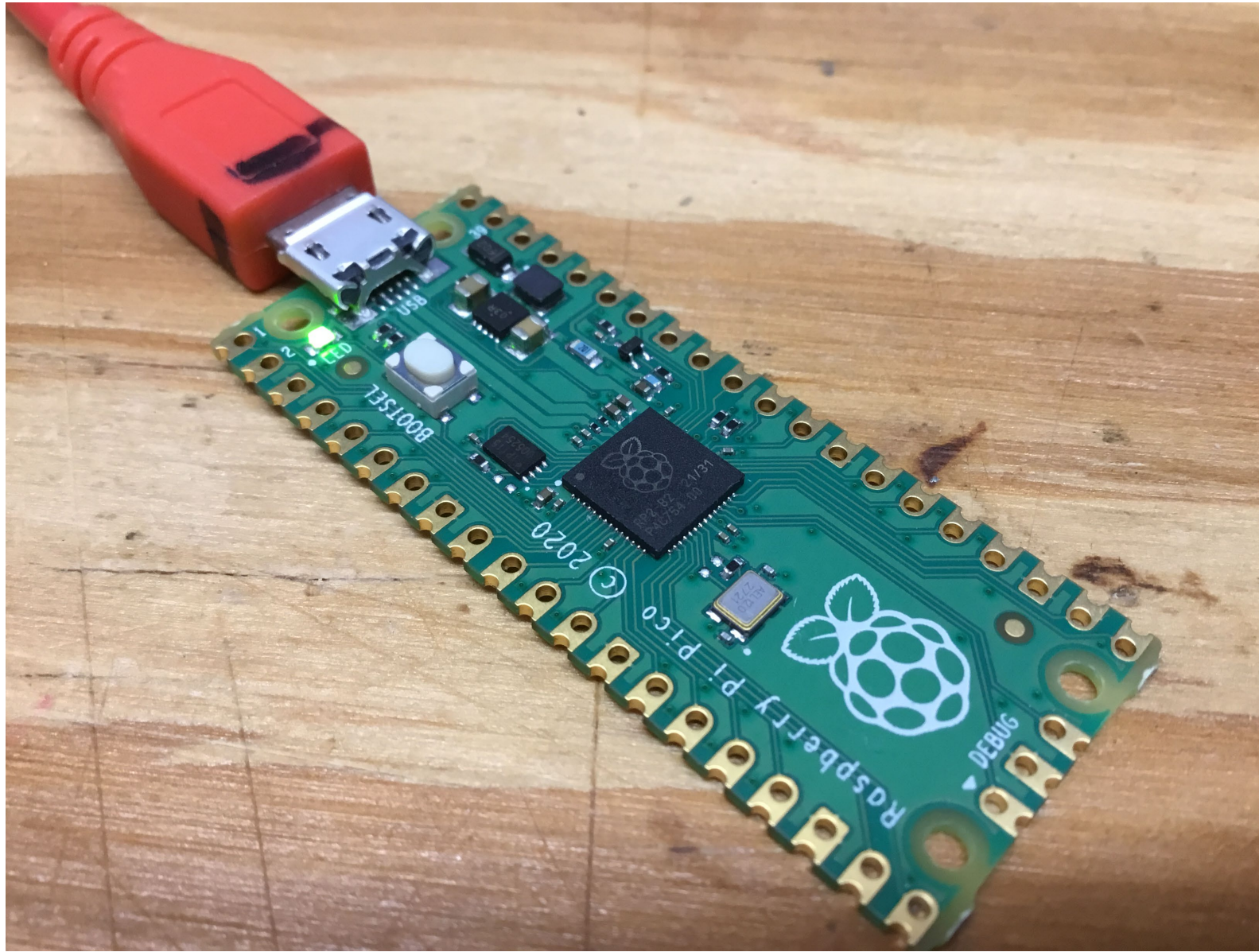
The code editor shows the following code snippet:

```
27 #elif defined(CYW43_WL_GPIO_LED_PIN)
28     // For Pico W devices we need to initialise the driver etc
29     return cyw43_arch_init();
30 #endif
31 }
32
33 // Turn the led on or off
34 void pico_set_led(bool led_on) {
35     #if defined(PICO_DEFAULT_LED_PIN)
```

The terminal shows the successful compilation and linking of the application:

```
* Executing task: /home/fred/.pico-sdk/ninja/v1.12.1/ninja -C /home/fred/pico2_Projects/blink/build
ninja: Entering directory `/home/fred/pico2_Projects/blink/build'
[2/2] Linking CXX executable blink.elf
* Terminal will be reused by tasks, press any key to close it.
```

Trust Me.. It is blinking...



Next Time...

MORE TO COME..

Thank you for attending!!!

Please consider the resources below:

- [Today's Download Package](#)
- raspberrypi.org





DesignNews

Thank You

Sponsored by

DigiKey

