ECE 421 Lecture 15

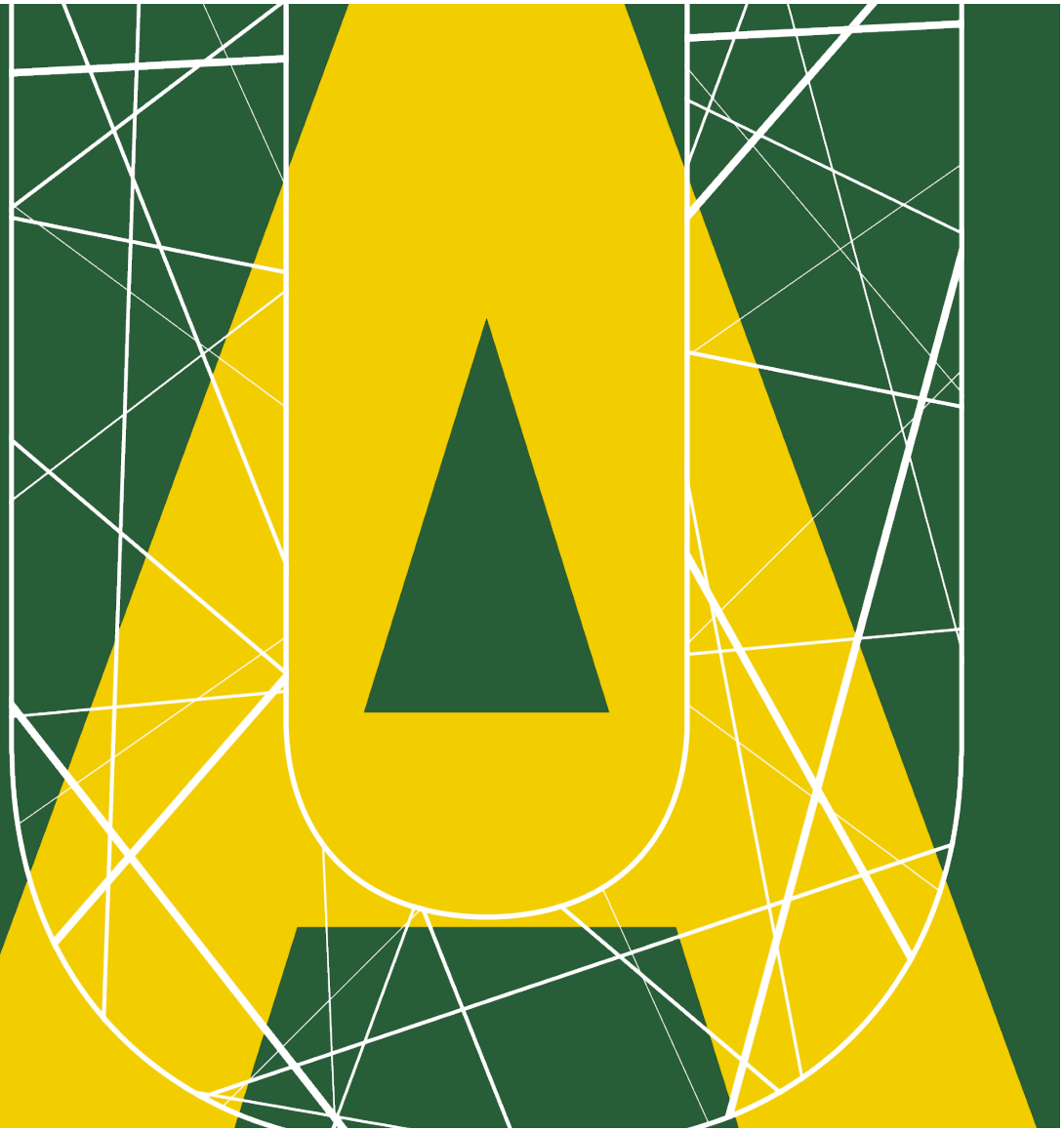
Embedded C/C++ on Raspberry Pi Pico

Authors

Steven Knudsen, PhD, PEng

Ron Unrau, PhD

UNIVERSITY OF ALBERTA



This Lecture — Embedded C/C++ on a Pico

- What is the Pico?
- Why do I care?
- Set up for development and test on Pico
- Embedded Hello World - ***blink***
- Debugging C/C++

Main References

https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf

https://github.com/StevenKnudsen

Outline

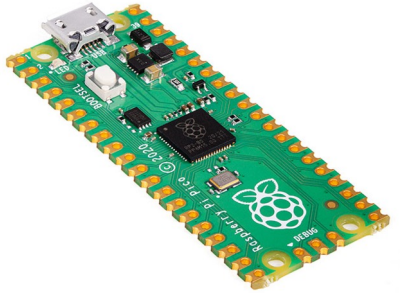
1. Install VSCode and set up
2. Setup a Pico as debug probe
3. Debug an example project (blink)
4. Create an ADC program and debug

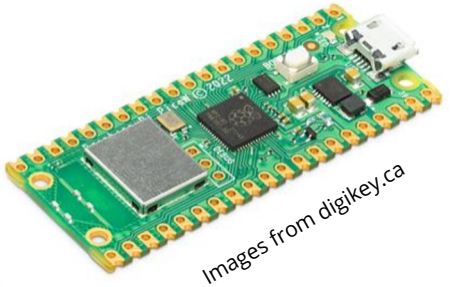
Will only highlight key points from the tutorials and any deviations

Raspberry Pi Pico and Pico W

Raspberry Pi Pico is a low-cost, high-performance microcontroller board with flexible digital interfaces. Key features include:

- RP2040 microcontroller chip designed by Raspberry Pi in the United Kingdom
- Dual-core Arm Cortex M0+ processor, flexible clock running up to 133 MHz
- 264 kB of SRAM, and 2MB of on-board flash memory
- USB 1.1 with device and host support
- Low-power sleep and dormant modes
- Drag-and-drop programming using mass storage over USB
- 26 × multi-function GPIO pins
- 2 × SPI, 2 × I2C, 2 × UART, 3 × 12-bit ADC, 16 × controllable PWM channels
- Accurate clock and timer on-chip
- Temperature sensor
- Accelerated floating-point libraries on-chip
- 8 × Programmable I/O (PIO) state machines for custom peripheral support



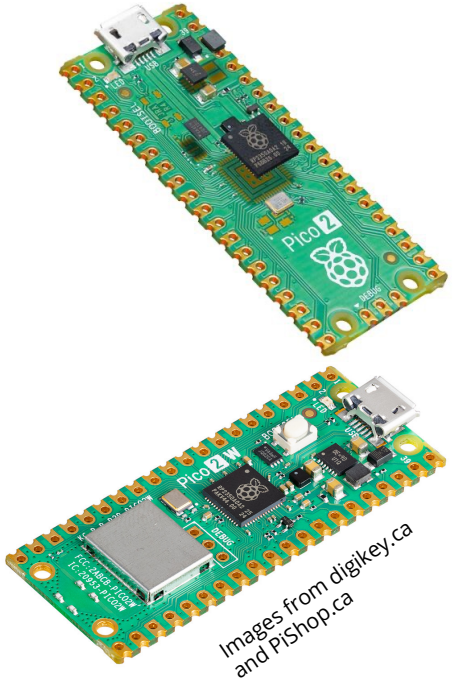


Images from digikey.ca

Raspberry Pi Pico 2 and Pico 2 W

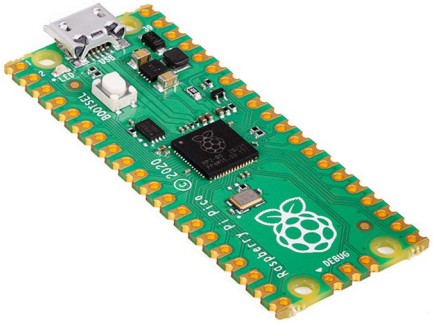
Raspberry Pi Pico 2 updates the Pico with:

- Dual Arm Cortex-M33 or dual Hazard3 RISC-V processors @ 150MHz
- 520 KB on-chip SRAM
- 2.4GHz 802.11n wireless LAN and Bluetooth 5.2 (Raspberry Pi Pico 2 W only)
- Software- and hardware-compatible with Raspberry Pi Pico 1
- Robust and fully documented security features:
  - ARM TrustZone for Cortex-M
  - Optional boot signing, enforced by on-chip mask ROM, with key fingerprint in OTP
  - Protected OTP storage for optional boot decryption key
  - Global bus filtering based on Arm or RISC-V security/privilege levels
  - Peripherals, GPIOs, and DMA channels individually assignable to security domains
  - Hardware mitigations for fault injection attacks
  - Hardware SHA-256 accelerator
- 2 × SPI, 2 × I2C, 2 × UART, 24 × PWM, 3 x 12-bit ADC
- 1 × USB 1.1 controller and PHY, with host and device support
- 12 × PIO state machines
- Open-source C/C++ SDK, MicroPython support



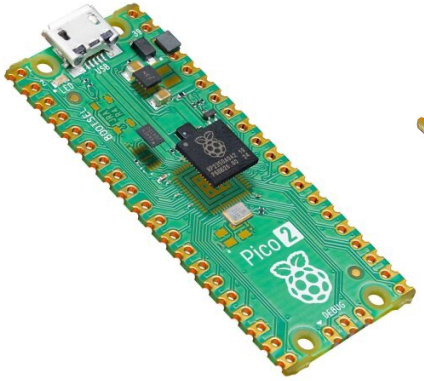
Images from digikey.ca and PiShop.ca

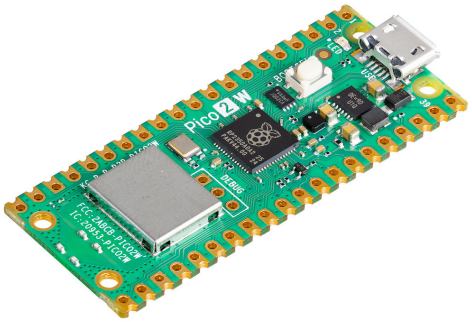
Pico & Pico W



$CA6.10 & $CA9.15

Pico 2 & Pico 2 W

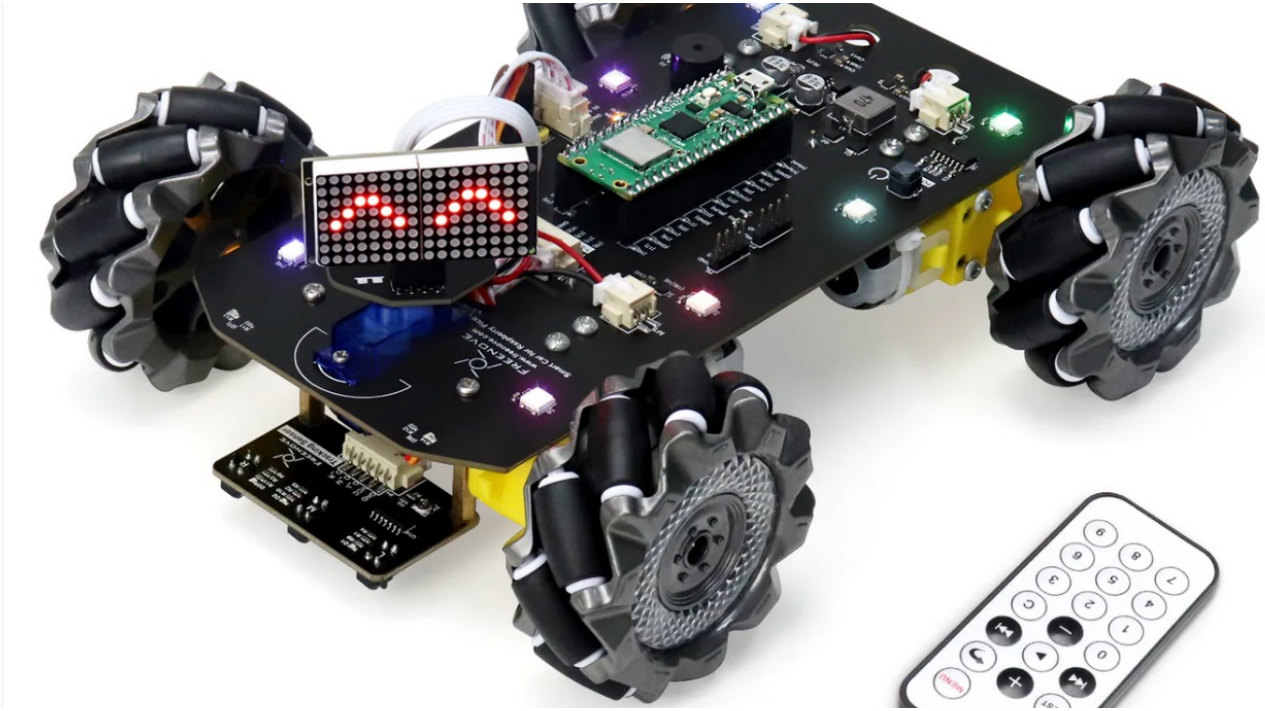


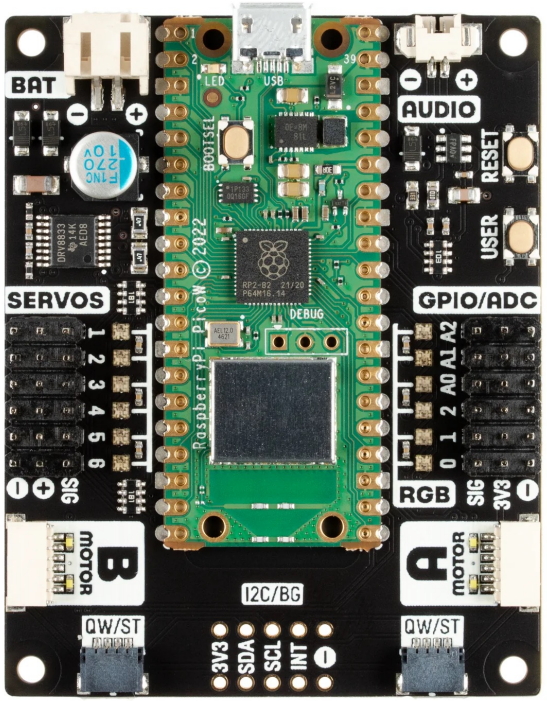


$CA7.63 & $CA10.68

at mouser.ca

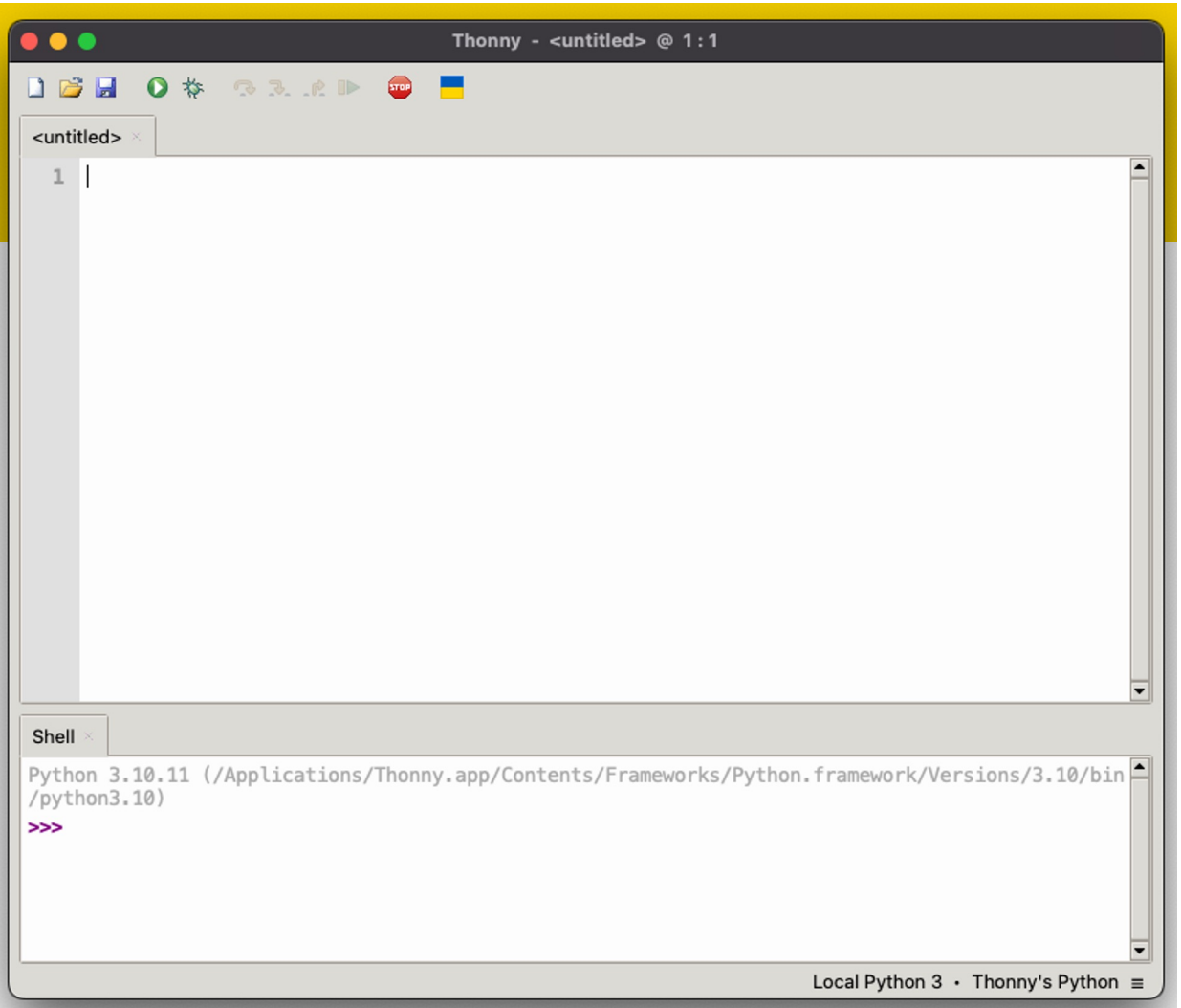
Pico will be core for Mechatronics





MicroPython

- Usual starting point is to use MicroPython
- Dead easy to use and program
- Not that interesting for us

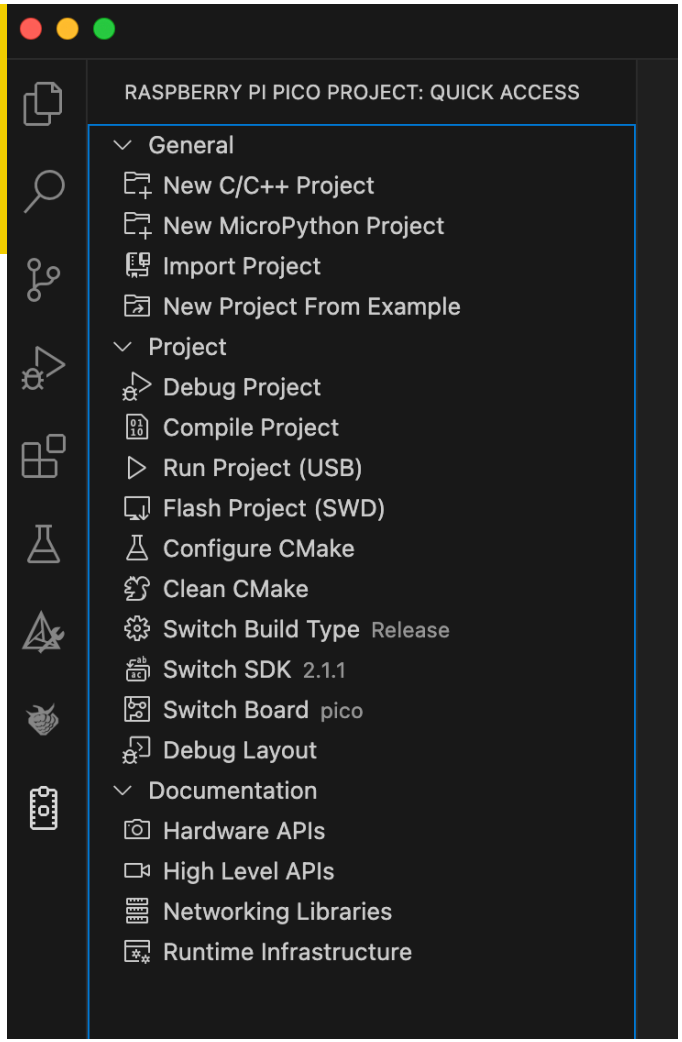


VSCode Setup

- Assume you know and use VSCode
- Install *Raspberry Pi Pico* extension
- Environment dependencies;
  - Windows — none
  - macOS — `xcode-select --install`
  - Linux — `sudo apt install python3 git tar build-essential`

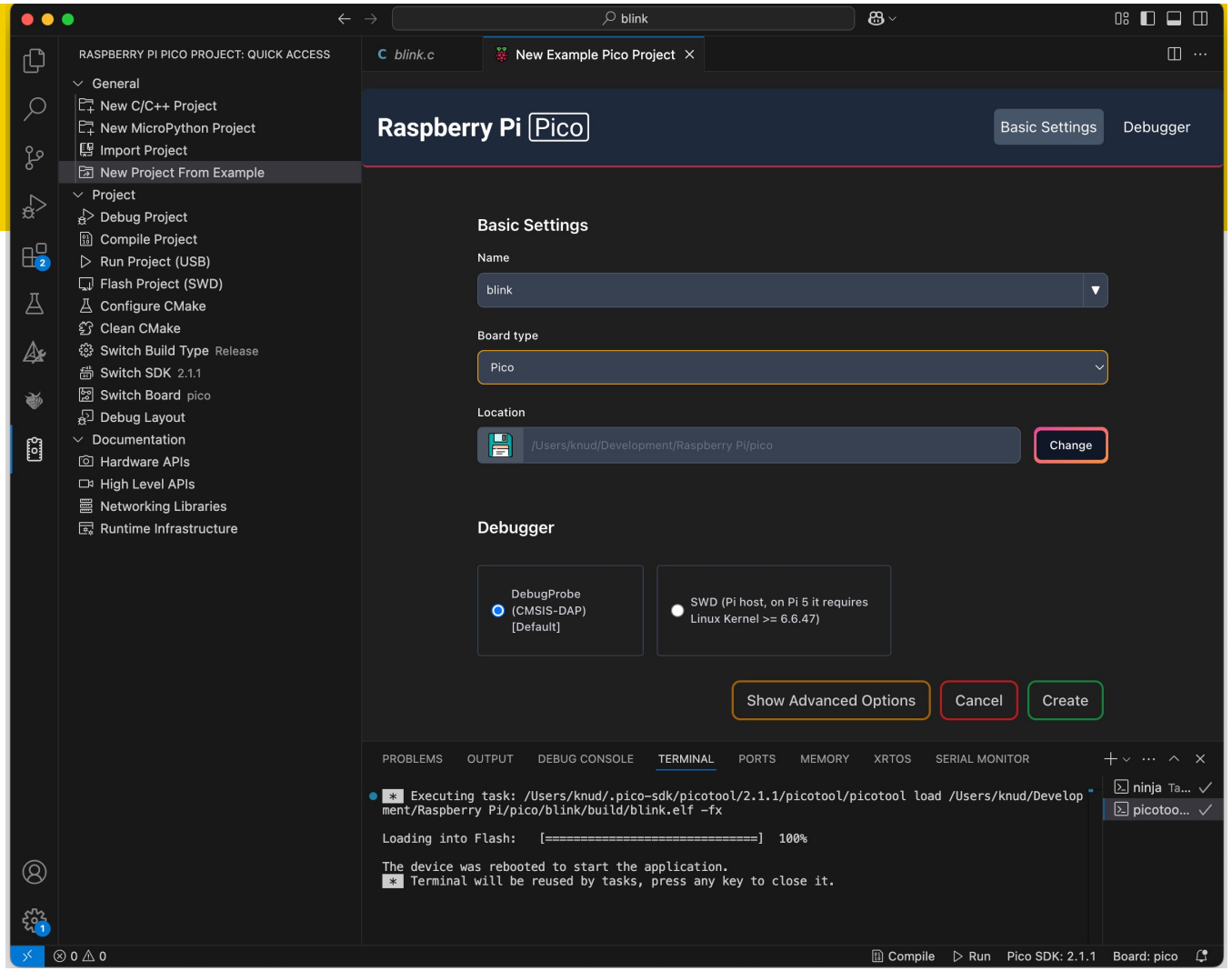
New Project from Example

- Click Pico icon on the left to get project quick access pane
- Click *New Project From Example*

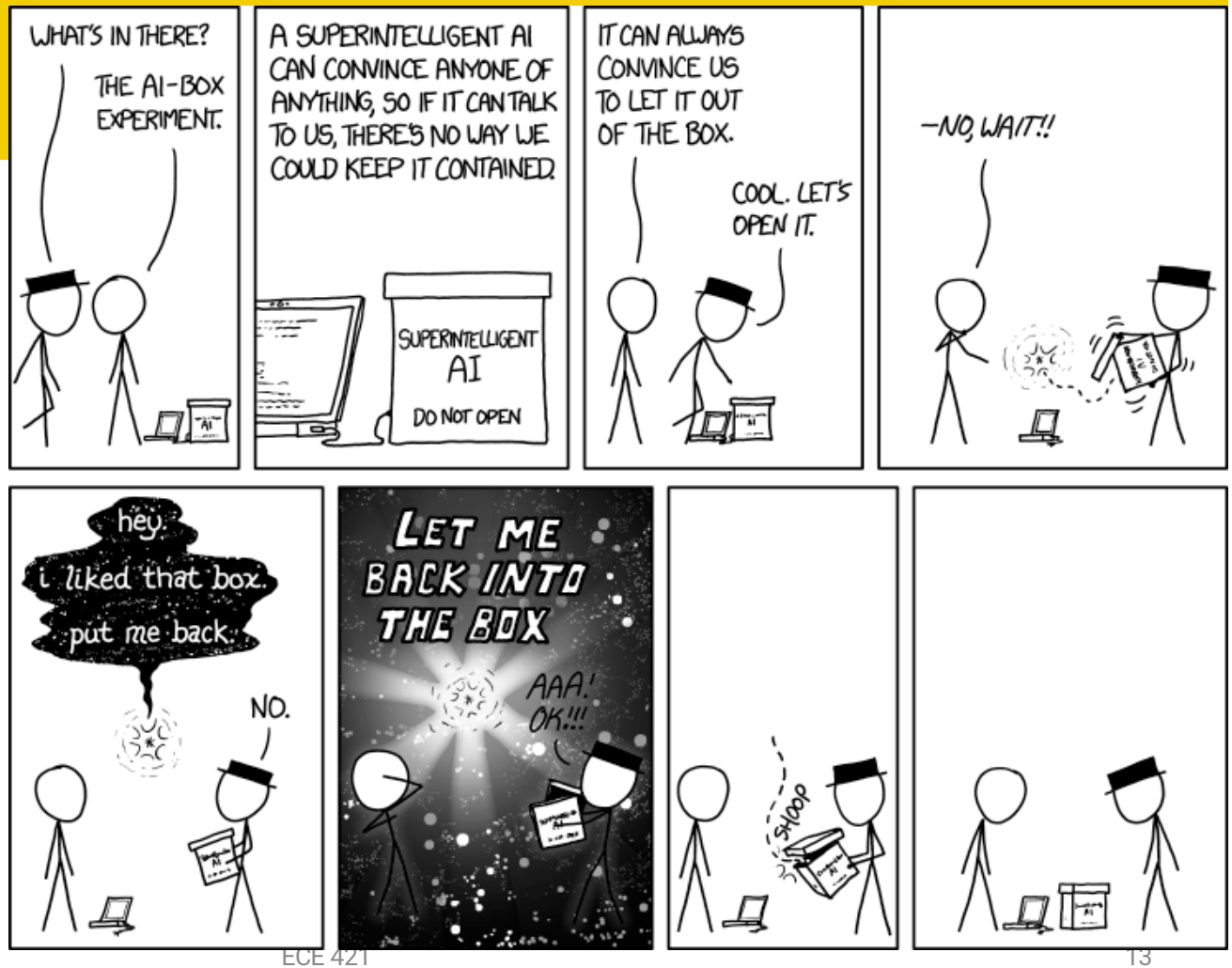


blink

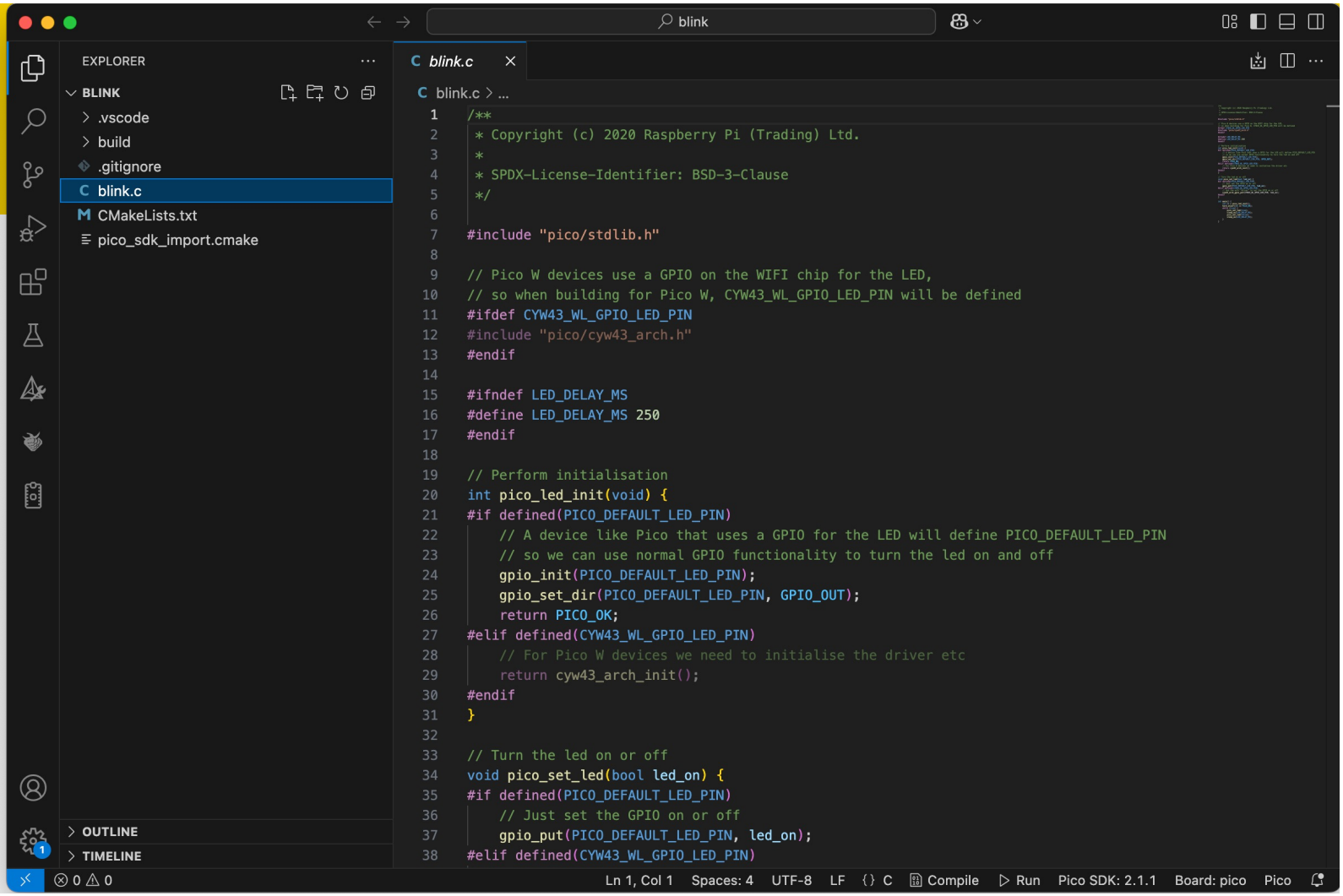
- Choose the example
- Set board type
  - Pico
- Set location
- Leave Debugger default
- Click Create



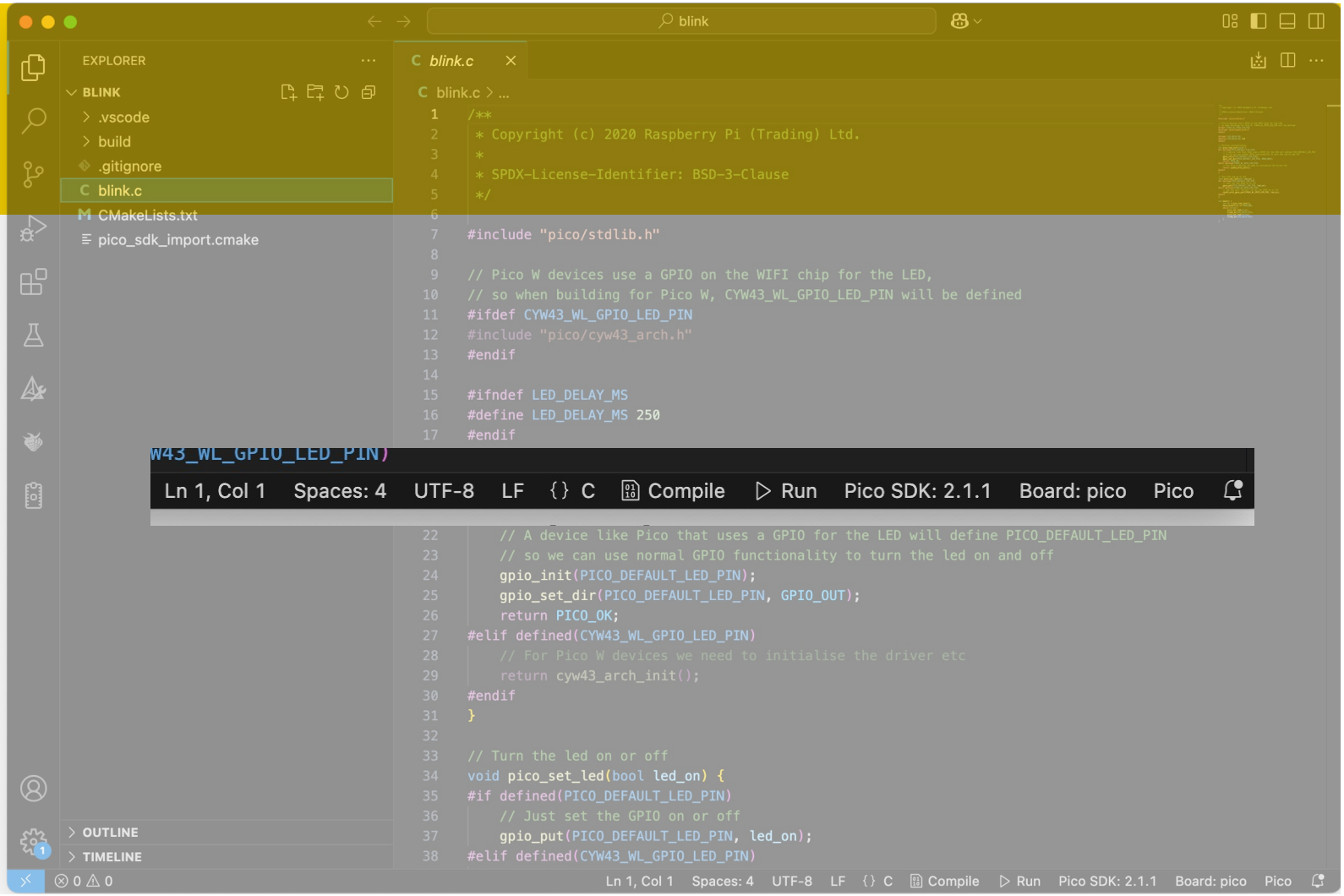
While you wait...



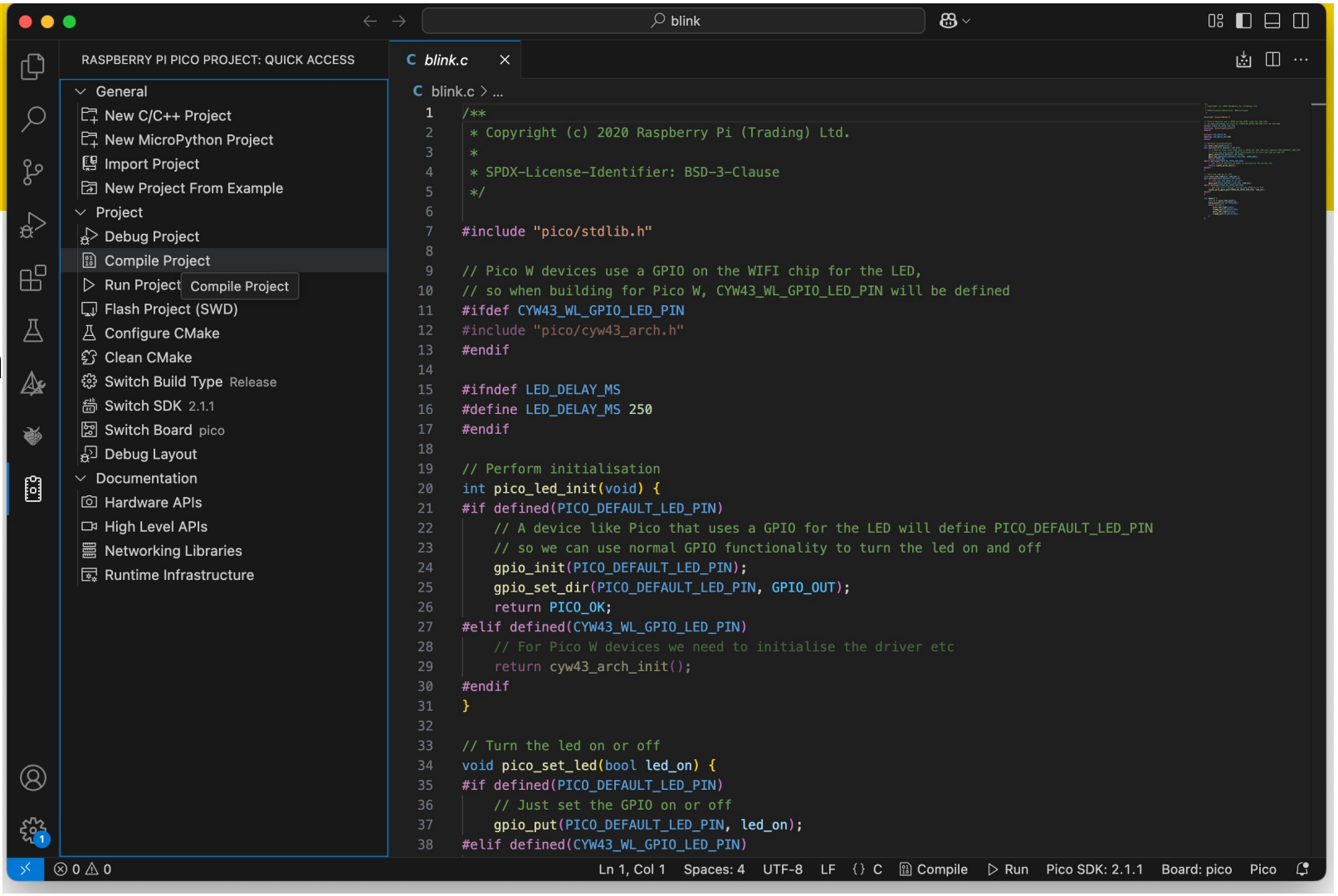
- Notice the convenience items at the bottom



- Notice the convenience items at the bottom



- Click the Pico icon on the left
- Compile Project



Pico in boot select (bootsel) mode

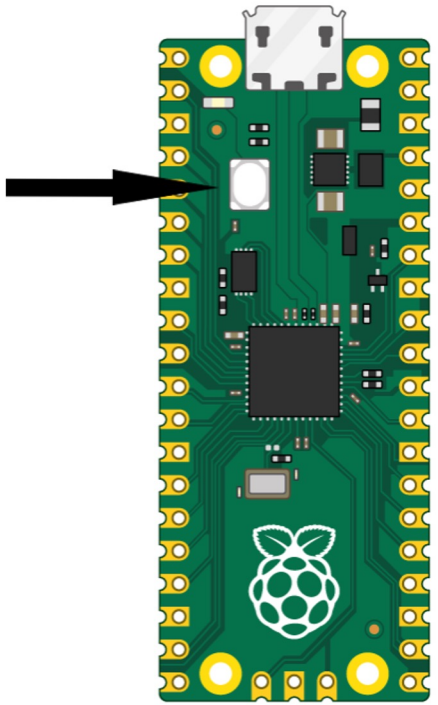
Unplug Pico from USB

Press and hold `bootsel` button

Plug Pico into USB and wait a few seconds

Release `bootsel` button

Pico appears as storage device

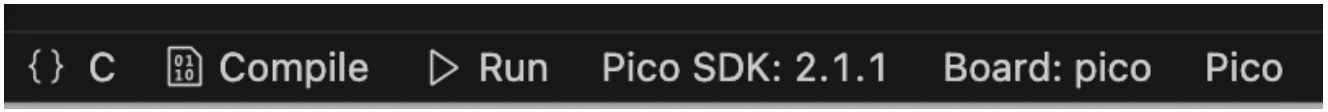


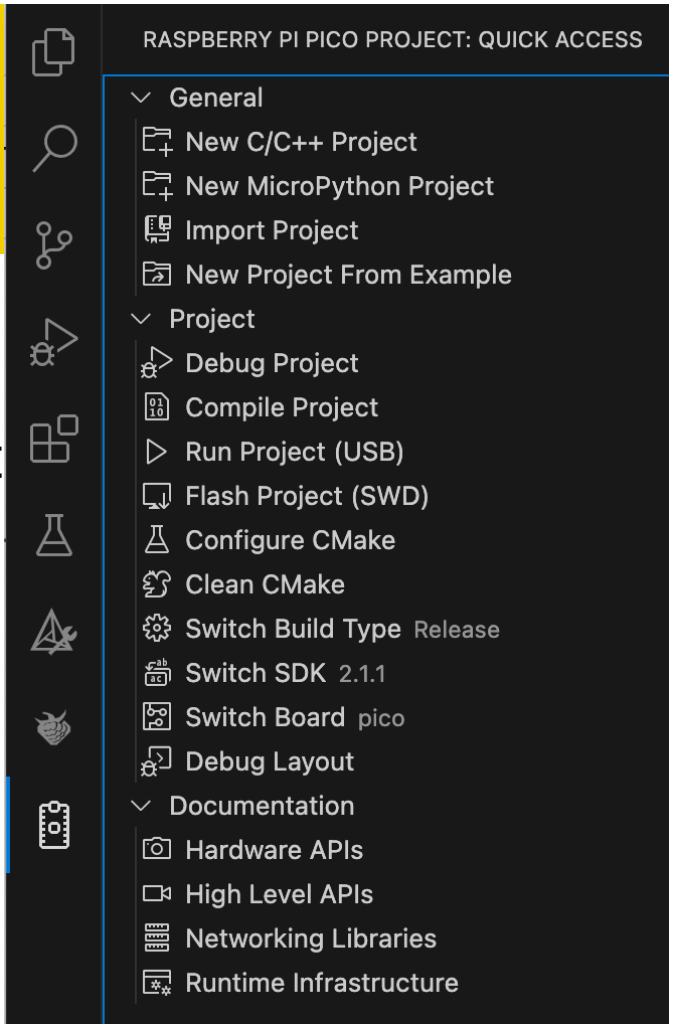
https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/3

Run it!

- With Pico in `BOOTSEL` mode,
- Click Run at bottom or Run Project (USB) on left

- Yay, a blinking LED



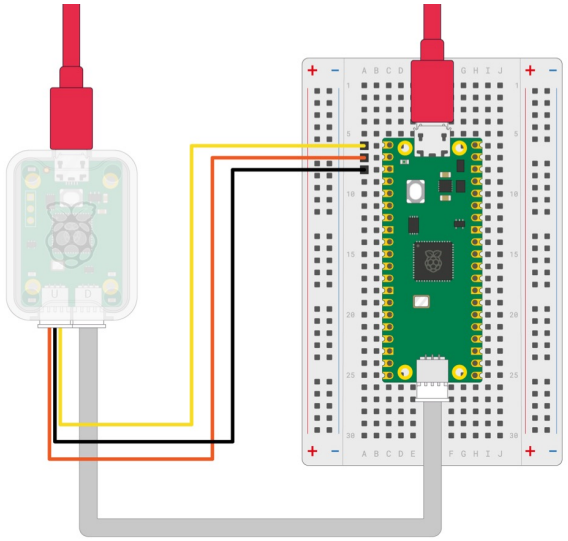


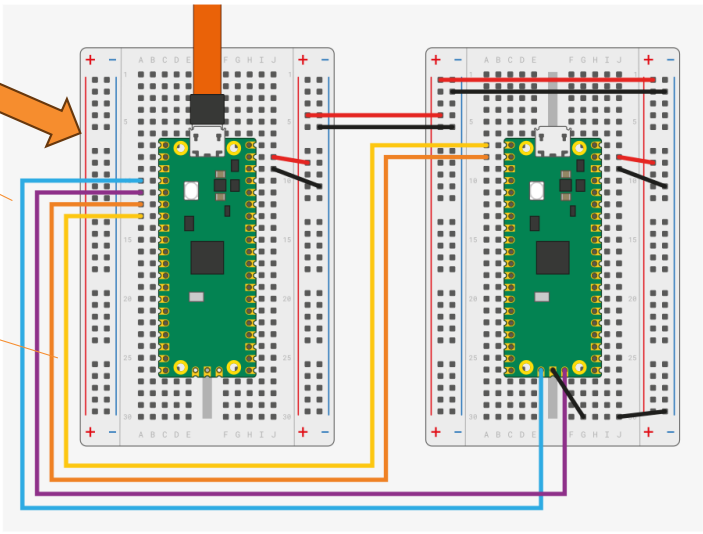
How to debug

- Use the Serial Wire Debug (SWD)
- Serial Wire Debug (SWD) is a 2-pin (SWDIO/SWCLK) electrical alternative Joint Test Action Group (JTAG) interface
- JTAG protocol used on top of the physical interface.
- SWD uses an ARM CPU standard bi-directional wire protocol
- Enables the debugger to become another Advanced Microcontroller Bus Architecture (AMBA) bus master for access to system memory and peripheral or debug registers

How to debug cont'd

- Need a device to run the SWD. Either
  - (new) Raspberry Pi probe, or
  - Second Pico configured for SWD



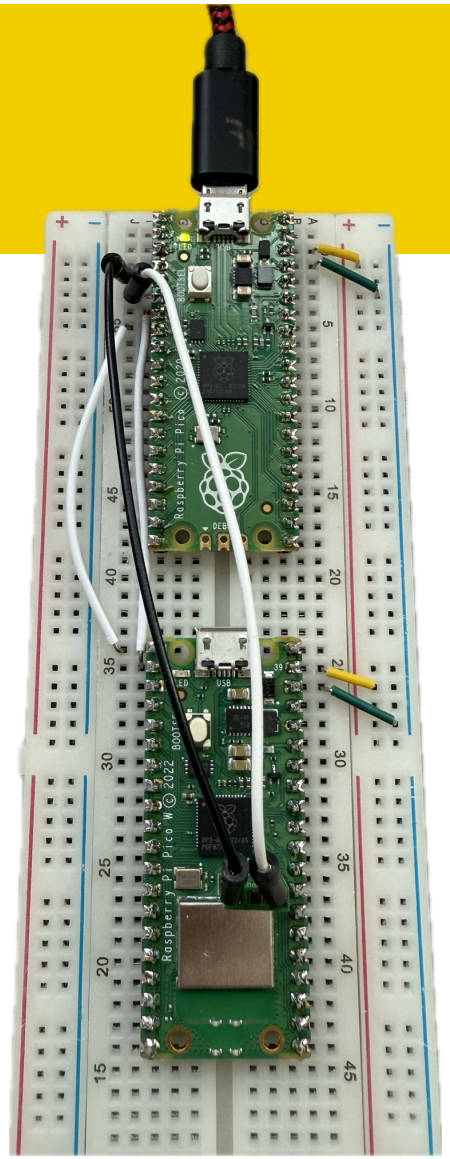


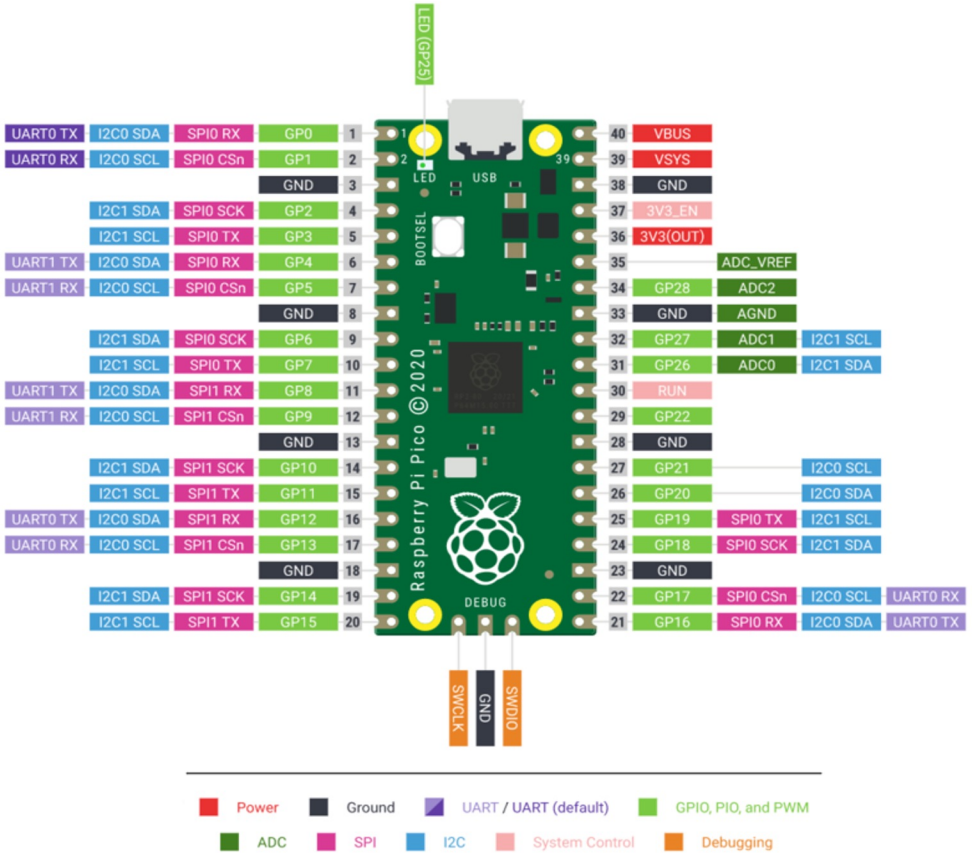
Blue and Purple are SWD lines

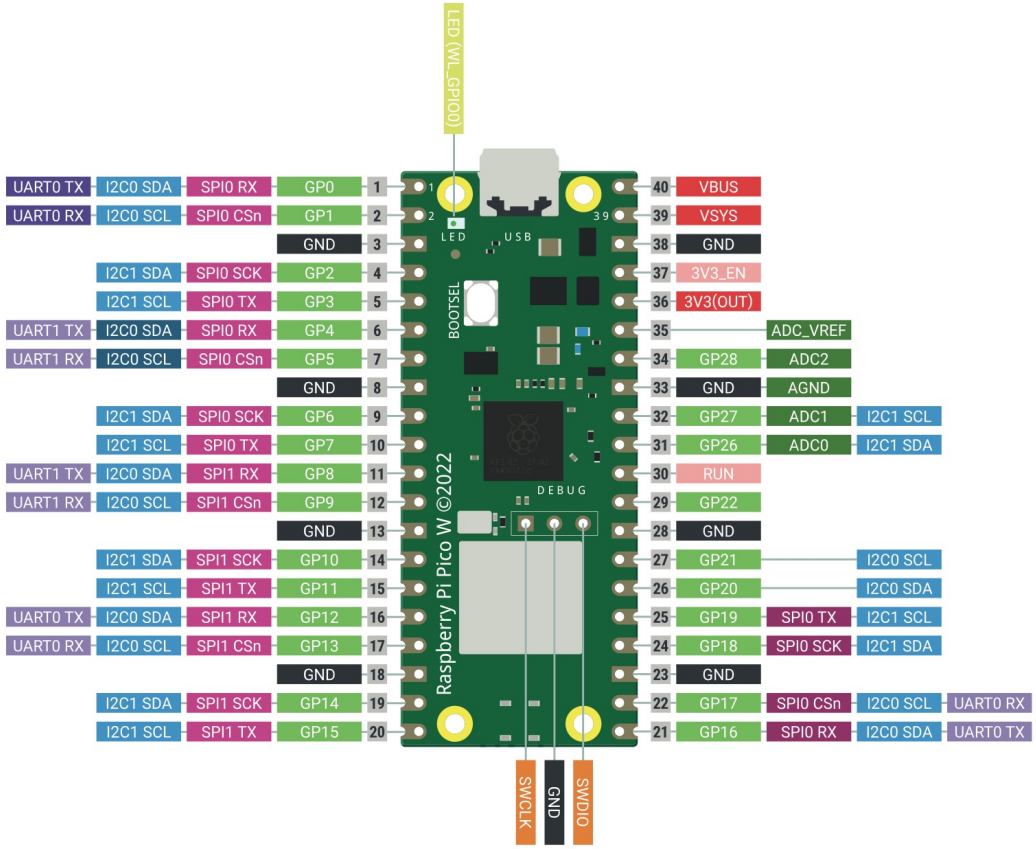
Yellow and Orange are UART/serial lines

How to debug cont'd

- Need a device to run the SWD. Either
  - (new) Raspberry Pi probe, or
  - Second Pico configured for SWD
- Here, target is Pico W
- B&W wires are SWD



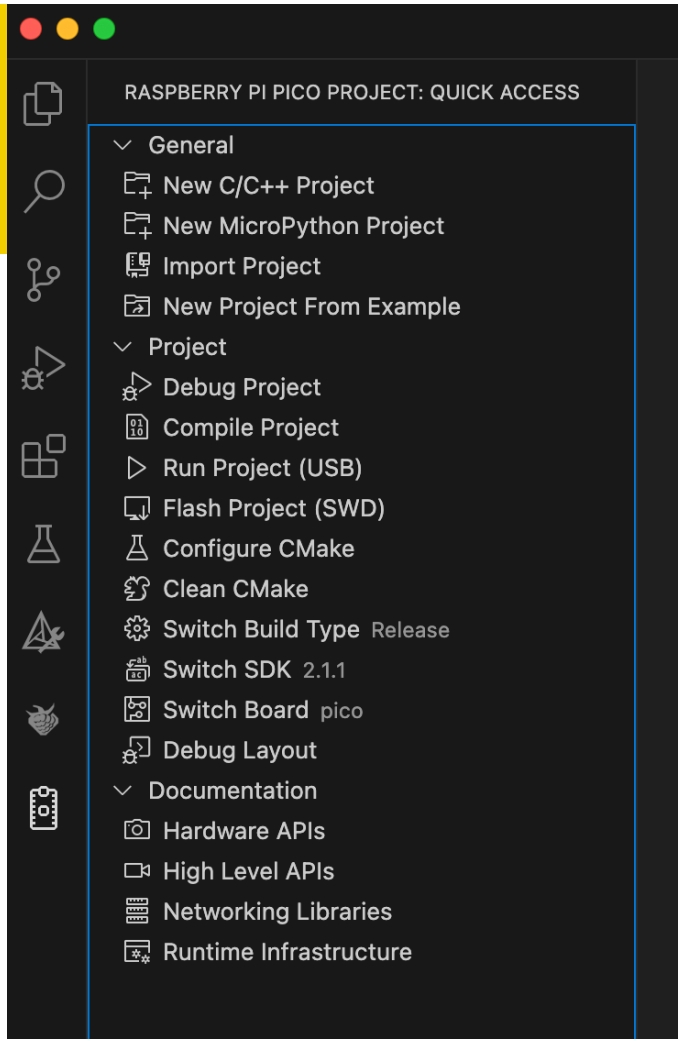

LED (GP25)
Raspberry Pi Pico ©2020
BOOTSEL
USB
LED
DEBUG
SWCLK
GND
SWDIO
Power
Ground
UART / UART (default)
GPIO, PIO, and PWM
ADC
SPI
I2C
System Control
Debugging


LED (WL_GPIO0)
Raspberry Pi Pico W ©2022
BOOTSEL
USB
LED
DEBUG
SWCLK
GND
SWDIO

New Project from Example

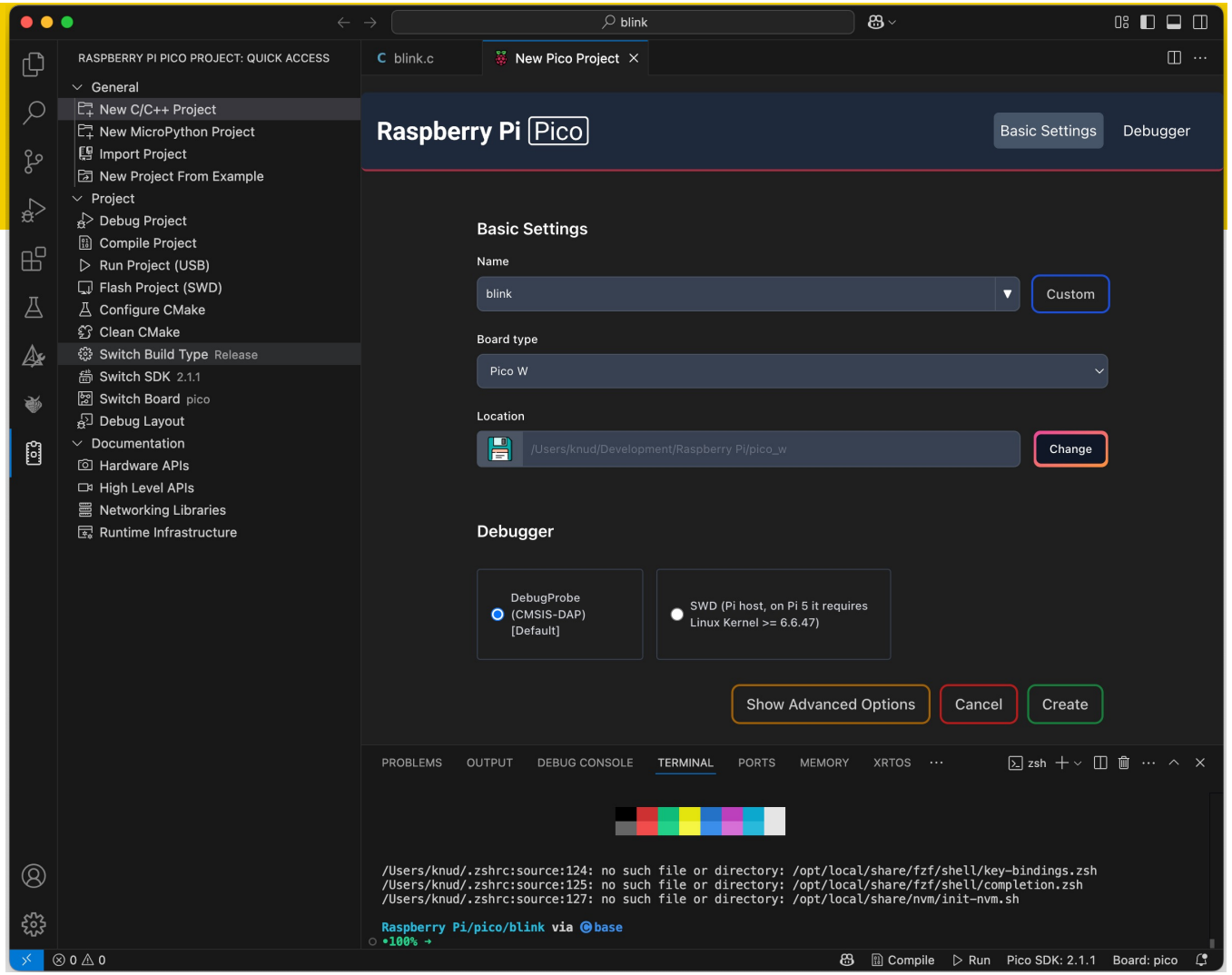
- Need to select the Pico W as target
- So need new example

- Click Pico icon on the left to get project quick access pane
- Click *New Project From Example*

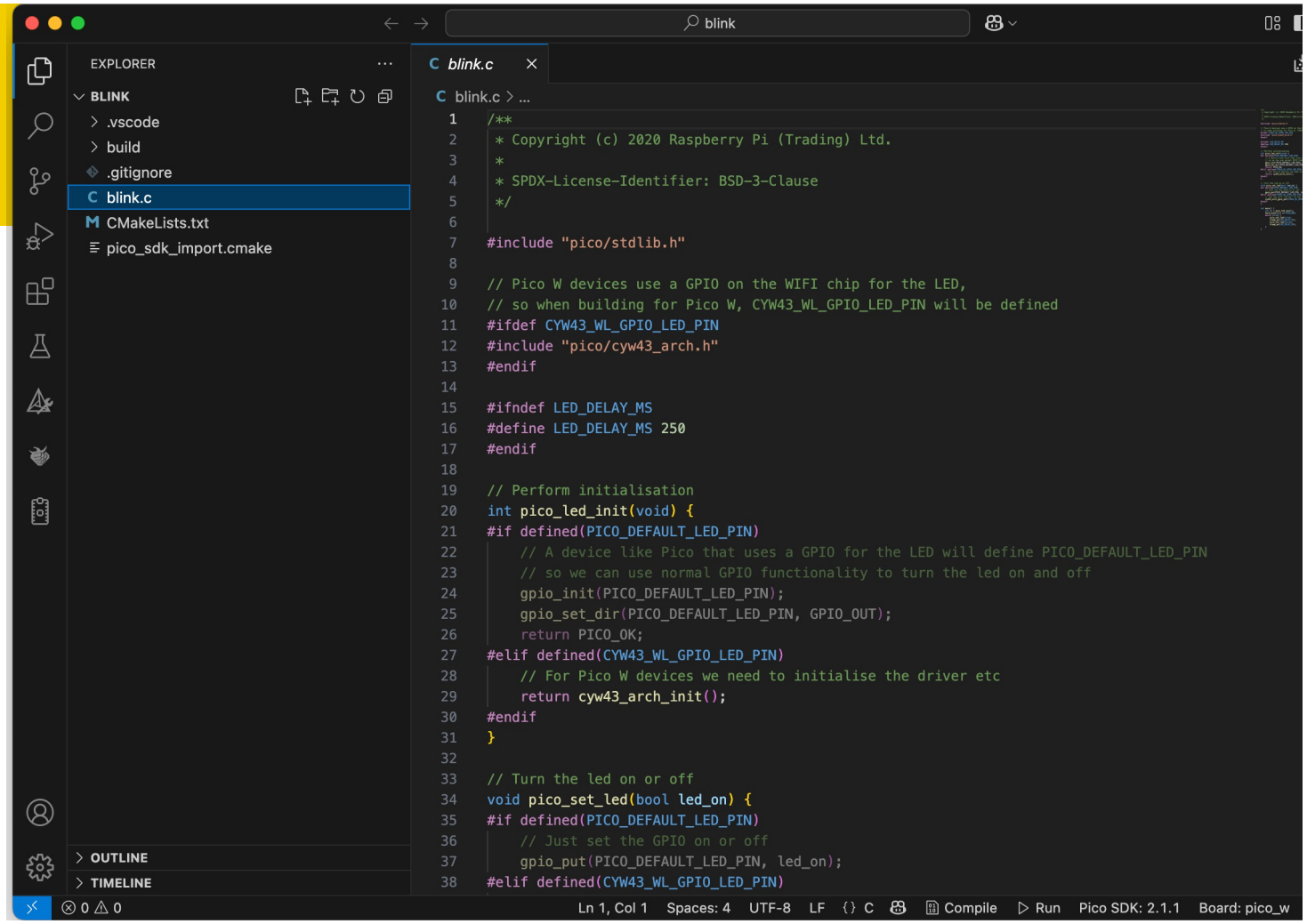


blink

- Choose the example
- Set board type
  - Pico W
- Set location
- Leave Debugger default
- Click Create



- Notice the `pico_led_init` is different
- CYW43 selected

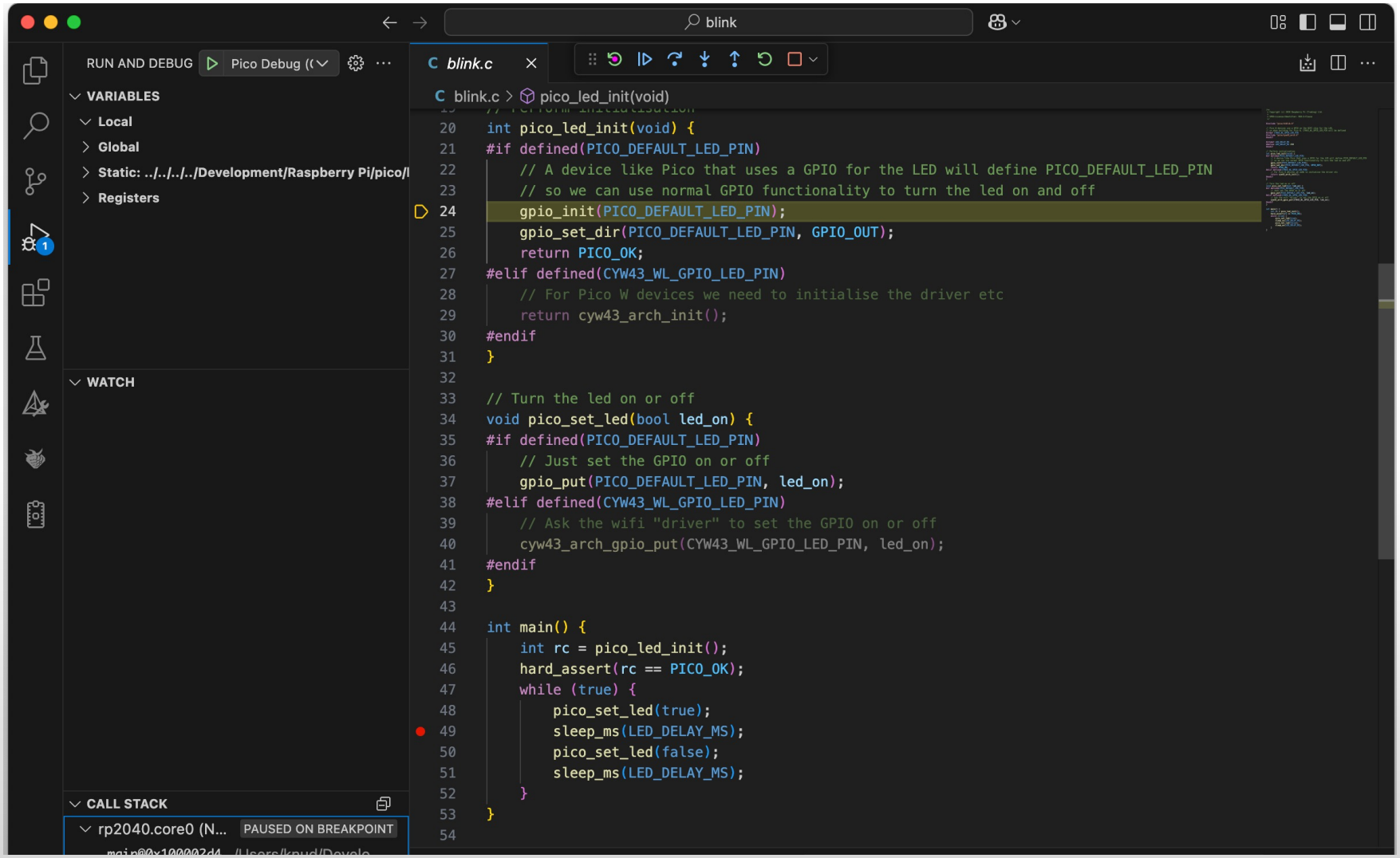


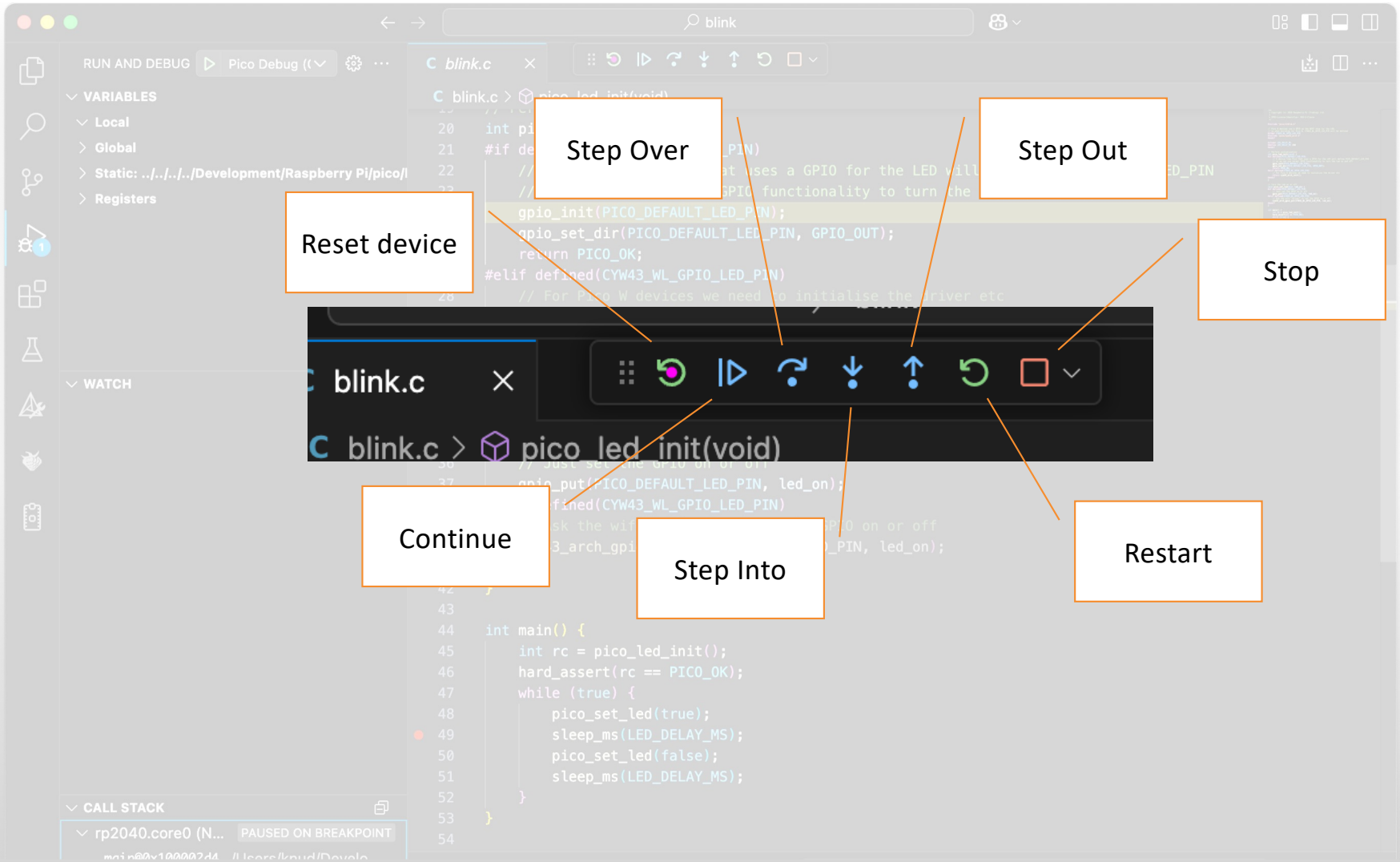
Configure Pico debugprobe

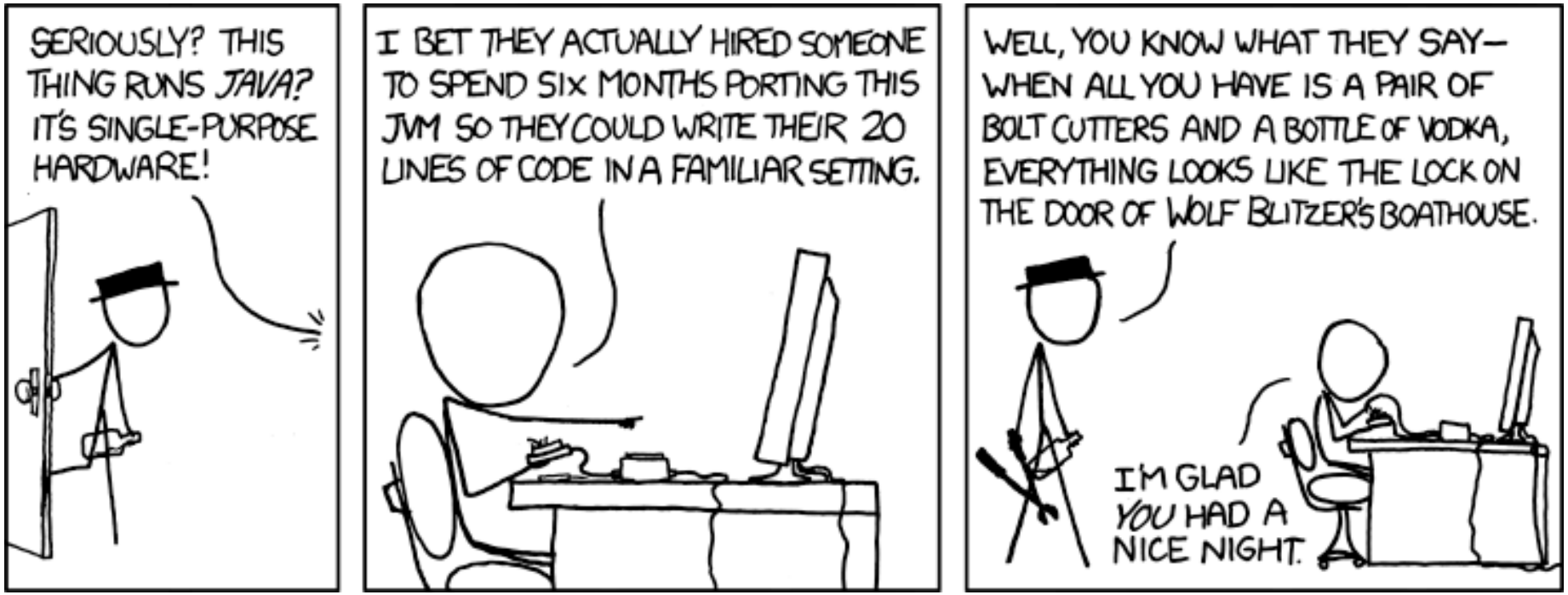
- https://github.com/raspberrypi/debugprobe/releases/tag/debugprobe-v2.2.1
- debugprobe_on_pico.uf2
- Put Pico into storage mode (hold BOOTSEL while applying power)
- Drag and drop the `uf2` file to program it
- For good measure, power it off/on

Time to debug…

- Select Debug Project or press F5.
- If prompted to select a debugger, choose `Pico Debug (Cortex-Debug)`
- The program will start, connected to the debugger, and stop


blink
RUN AND DEBUG
Pico Debug ((
VARIABLES
Local
Global
Static: ../../../../Development/Raspberry Pi/pico/l
Registers
WATCH
CALL STACK
rp2040.core0 (N...
PAUSED ON BREAKPOINT
blink.c
blink.c > pico_led_init(void)
20 int pico_led_init(void) {
21 #if defined(PICO_DEFAULT_LED_PIN)
22 // A device like Pico that uses a GPIO for the LED will define PICO_DEFAULT_LED_PIN
23 // so we can use normal GPIO functionality to turn the led on and off
24 gpio_init(PICO_DEFAULT_LED_PIN);
25 gpio_set_dir(PICO_DEFAULT_LED_PIN, GPIO_OUT);
26 return PICO_OK;
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)
36 // Just set the GPIO on or off
37 gpio_put(PICO_DEFAULT_LED_PIN, led_on);
38 #elif defined(CYW43_WL_GPIO_LED_PIN)
39 // Ask the wifi "driver" to set the GPIO on or off
40 cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, led_on);
41 #endif
42 }
43
44 int main() {
45 int rc = pico_led_init();
46 hard_assert(rc == PICO_OK);
47 while (true) {
48 pico_set_led(true);
49 sleep_ms(LED_DELAY_MS);
50 pico_set_led(false);
51 sleep_ms(LED_DELAY_MS);
52 }
53 }
54


Step Over
Step Out
Reset device
Stop
blink.c
blink.c > pico_led_init(void)
Continue
Step Into
Restart



https://xkcd.com/801/

Questions

https://www.monkeyuser.com/2018/focus/?ref=comic