

# ECE 421 Lecture 16

## Embedded Rust on Raspberry Pi Pico

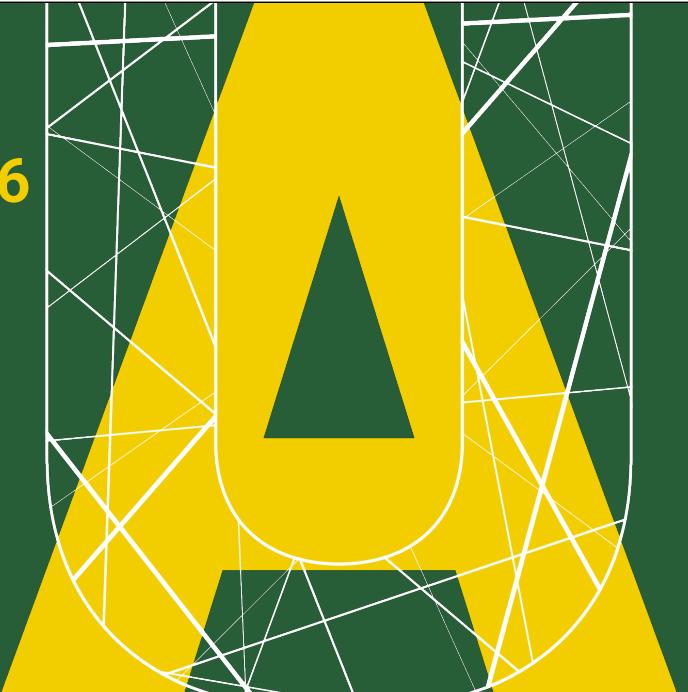
### Authors

Steven Knudsen, PhD, PEng

Ron Unrau, PhD



UNIVERSITY  
OF ALBERTA



# This Lecture – Embedded Rust on a Pico

- Set up for development and test on Pico
- Embedded Hello World - *blink*
- Debugging Rust



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

2

# Main References

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

<https://github.com/StevenKnudsen>



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

3

Design Patterns aka Gang of Four book is available for reading online at UofA Library.  
The others are not.

One can be found

# 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***



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

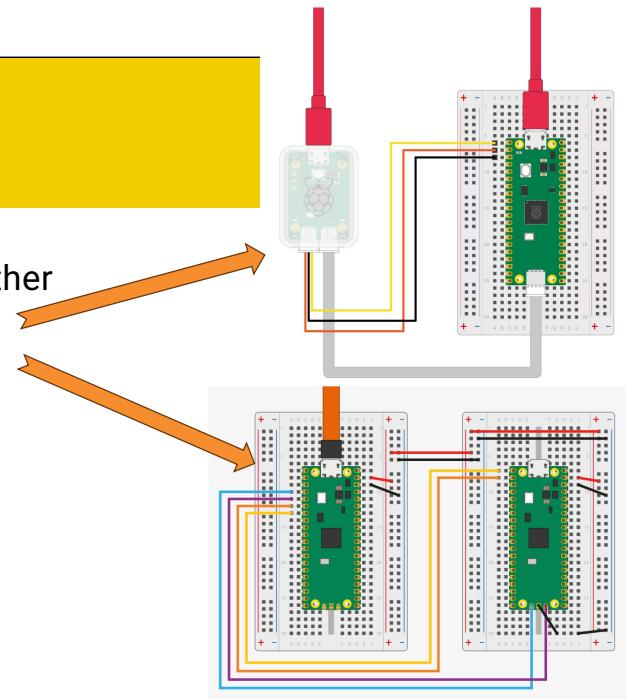
4

# 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

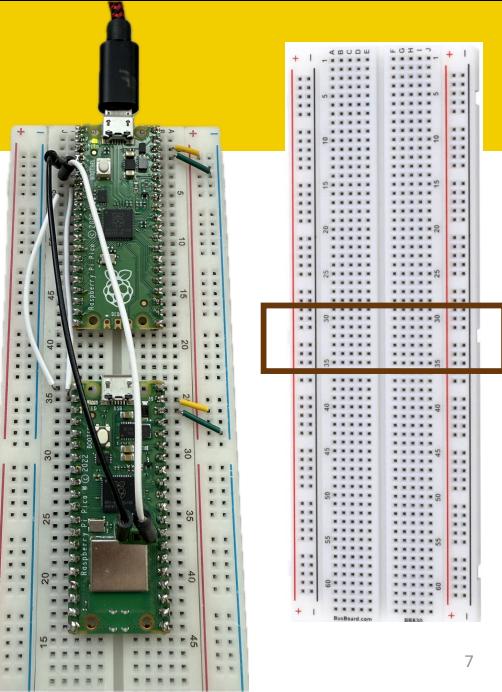
## How to debug cont'd

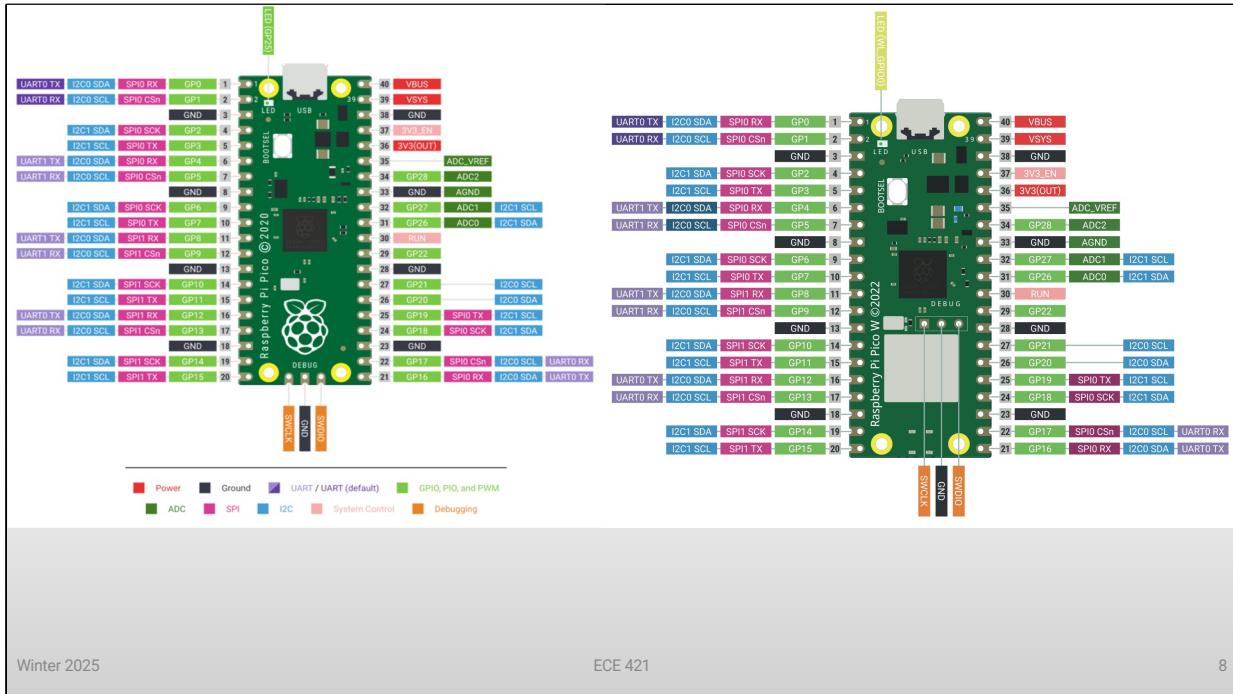
- Need a device to run the SWD. Either
  - (new) Raspberry Pi probe, or
  - Second Pico configured for SWD



## How to debug reviewed

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





## Make sure Rust is up to date

- Visit [rust-lang.org/tools/install](https://rust-lang.org/tools/install) to install rustup
- Tool that installs and manages Rust
- Rust is developed on a 6-week sprint, good to update often

```
rustup self update  
rustup update stable
```



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

9

## rp-rs

- Github project that supports a Rust HAL (hardware abstraction layer) that enabled embedded programming in Rust on the Raspberry Pi Pico family

```
rustup target add thumbv6m-none-eabi  
cargo install flip-link
```

- May have dependency issues. Ubuntu 24.04 needed libudev-dev installed, for example



# VSCode

- Install rust-analyzer, Debugger for probe-rs
- Install crates – Keeps crates up to date
- Get `debugprobe_on_pico.uf2` from  
[github.com/raspberrypi/debugprobe/releases](https://github.com/raspberrypi/debugprobe/releases)
  - Drag and drop programming with pico in bootloader mode



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

11

# Probe-rs – effectively gdb

- Use probe-rs
- Check for pico probe and show output
- Could use cargo install --locked probe-rs-tools, but instead

```
$ curl --proto '=https' --tlsv1.2 -LsSf https://github.com/probe-rs/probe-rs/releases/latest/download/probe-rs-tools-installer.sh | sh
```

```
$ probe-rs list
```

The following debug probes were found:

```
[0]: Debugprobe on Pico (CMSIS-DAP) - 2e8a:000c:E6616407E33CA42F (CMSIS-DAP)
```

- Be sure to set up udev rules in Linux



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

12

# Blinky Program Example

- Borrow from Murray Todd on Medium

<https://murraytodd.medium.com/learning-rust-with-embedded-programming-on-rp2040-e784389d2d3d>

- Use hardware abstraction layer (HAL) from rp-rs project

<https://github.com/rp-rs>



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

13

# Blinky

- Go to rp2040-project-template, clone

```
git clone https://github.com/rp-rs/rp2040-project-template.git
```

- Might as well rename the project template folder to `blinky`



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

14

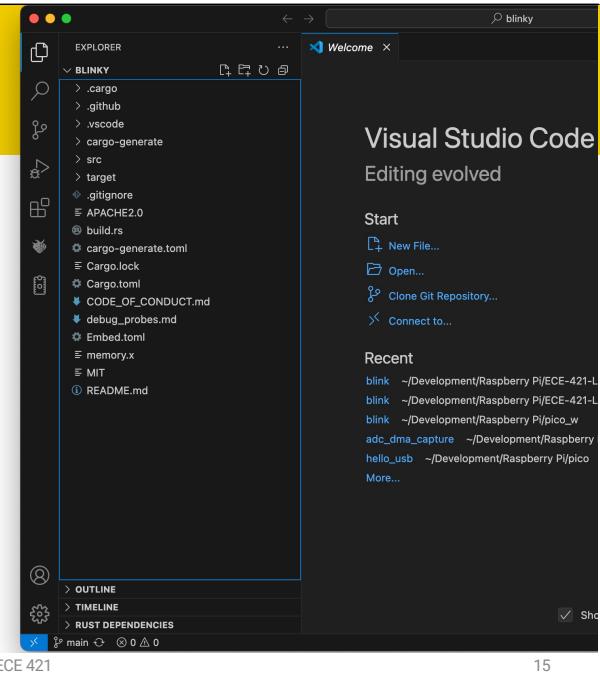
# VSCode work

- Open the `blinky` folder in VSCode
- We will be working in some of the folders to modify the template defaults



UNIVERSITY  
OF ALBERTA

Winter 2025



15

## VSCode work

- Check in .cargo/config.toml that

```
runner = "probe-rs run --chip RP2040 --protocol swd"
```

- In Cargo.toml rename the project to `blinky`
- In .vscode/launch.json change “name” to `blinky_pwm` (line 9) and on (line 26)

```
"programBinary": "target/thumbv6m-none-eabi/debug/rp2040-project-template"
```

to

```
"programBinary": "target/thumbv6m-none-eabi/debug/blinky"
```



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

16

## VSCode work cont'd

- In .vscode/launch.json, uncomment the line

```
"svdFile": "./.vscode/rp2040.svd"
```

- Manually download rp2040.svd from

```
curl -O https://raw.githubusercontent.com/raspberrypi/pico-sdk/1.3.1/src/rp2040/hardware_regs/rp2040.svd
```

and place it in the .vscode directory so that the debugger can find it



## VSCode work cont'd

- In VSCode, install the Debugger for probe-*rs* extension
- Time to run the debugger!



UNIVERSITY  
OF ALBERTA

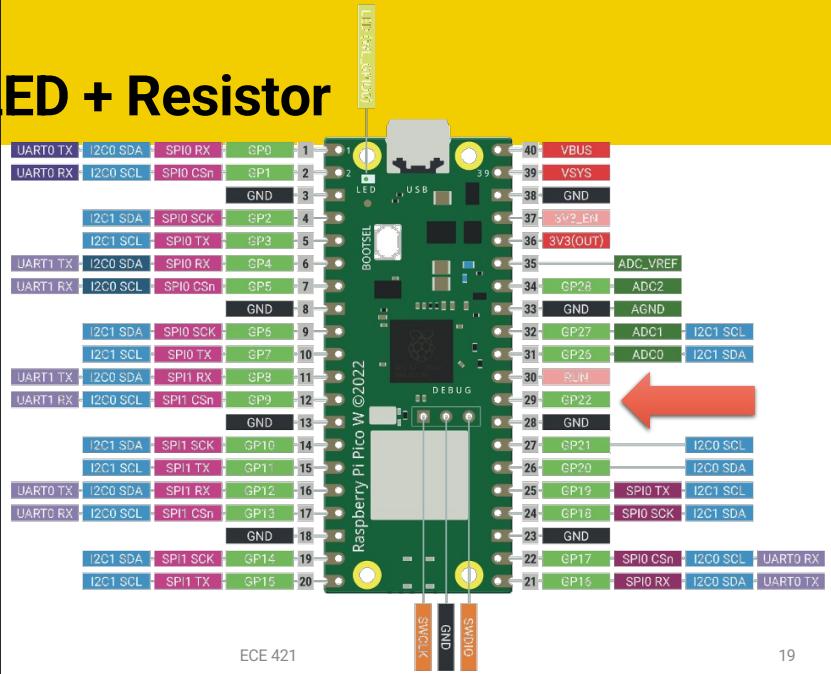
Winter 2025

ECE 421

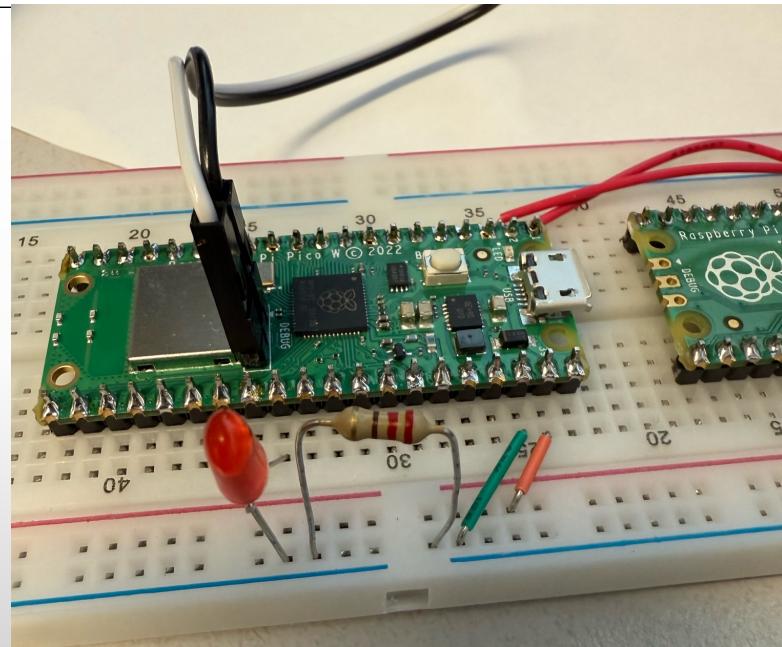
18

## Need to add LED + Resistor

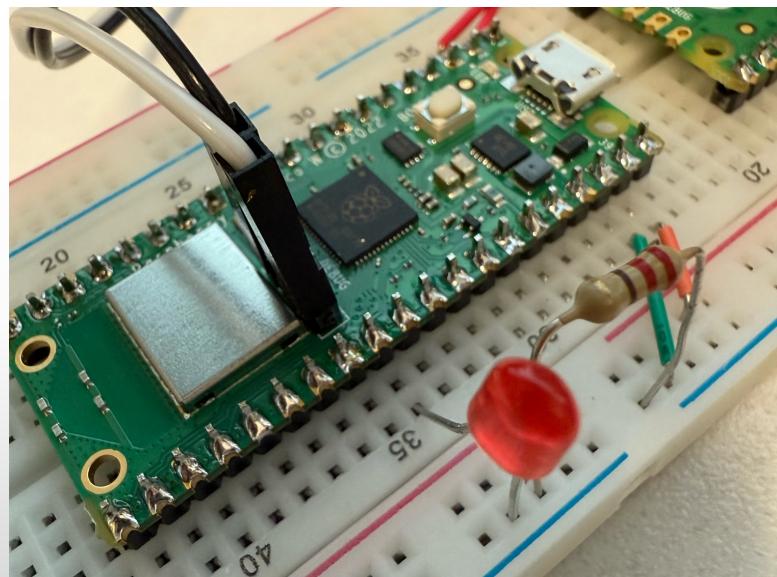
- We need to add an LED and resistor
- The red-blue rails of the breadboard are not connected in the middle.
- Connect the blue rails using the resistor



- The red–blue rails of the breadboard are not connected in the middle.
- Connect the blue rails using the resistor



- Put the long lead of the LED in the breadboard at GPIO22 pin's location
- Put the short lead in the blue rail



# Set breakpoints

- Set breakpoints on the `info!()` statements
- Select Run/Start Debugging

```
src > main.rs > main
26 fn main() -> ! {
57     // on-board LED, it might need to be config
58     //
59     // Notably, on the Pico W, the LED is
60     // One way to do that is by using [emb
61     //
62     // If you have a Pico W and want to connect
63     // LED to one of the GPIO pins, and re
64     // in series with the LED.
65     let mut led_pin: Pin<Gpio22, FunctionsS
66
67     loop {
68         info!("on!");
69         led_pin.set_high().unwrap();
70         delay.delay_ms(500);
71         info!("off!");
72         led_pin.set_low().unwrap();
73         delay.delay_ms(500);
74     }
75 } fn main
76
77 // End of file
```



# Stepping

- Ta Da
- Can't set breakpoints on some statements, which is probably a compilation setting I am missing...

```
src > main.rs > main
26 fn main() -> ! {
27
28     // This is the correct pin on the Rasp
29     // on-board LED, it might need to be c
30     //
31     // Notably, on the Pico W, the LED is
32     // One way to do that is by using [emb
33     //
34     // If you have a Pico W and want to to
35     // LED to one of the GPIO pins, and re
36     // in series with the LED.
37     let mut led_pin: Pin<Gpio22, FunctionS
38
39     loop {
40         info!("on!");
41         led_pin.set_high().unwrap();
42         delay.delay_ms(500);
43         info!("off!");
44         led_pin.set_low().unwrap();
45         delay.delay_ms(500);
46     }
47 }
48 } fn main
49
50 // End of file
```



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

23

# Information

- In debug mode we have access to the Peripherals via the Hardware Abstraction Layer (HAL)
- All kinds of interesting statics
- Scroll down the variables to the pins and pop it open



UNIVERSITY  
OF ALBERTA

Winter 2025

The screenshot shows a debugger interface with the following details:

- VARIABLES** pane:
  - Selected item: `> Peripherals`
  - Other items: `> Static`, `> Registers`, `> Variables`
  - Variables listed:
    - `> pac = Peripherals @ <unknown value>`
    - `> core = Peripherals @ <unknown value>`
    - `> watchdog = Watchdog @ 0x2003FB70`
    - `> sio = Sio @ <unknown value>`
    - `external_xtal_freq_hz = Unimplemented:...`
    - `> clocks = ClocksManager @ <unknown value>`
    - `> delay = Delay @ 0x2003FB74`
- WATCH** pane: A single red dot indicates a breakpoint.
- CALL STACK** pane:
  - Halted on breakpoint (Unknown) @0x...
  - Items: `__cortex_m_rt_main`, `main.rs 68:9`, `<unknown function @ 0x1000020e> Un...`

ECE

24

## Variables - pins

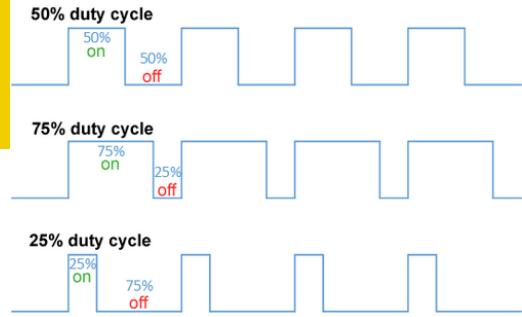
- All kinds of information is exposed via the HAL

The screenshot shows a debugger interface with the following details:

- RUN AND DEBUG** button and dropdown set to "rp2040-proje".
- VARIABLES** pane:
  - Variables** section:
    - delay** = Delay @ 0x2003FB74  
frequency = 12500000
    - pins** = Pins @ <unknown value>
      - > **gpio0** = Pin<rp2040\_hal::gpio::pin::ba...
      - > **gpio1** = Pin<rp2040\_hal::gpio::pin::ba...
      - > **gpio2** = Pin<rp2040\_hal::gpio::pin... [01]
      - > **gpio3** = Pin<rp2040\_hal::gpio::pin::bank0::Gpio...
      - > **gpio4** = rp2040\_hal::gpio::pull::PullDown>
      - > **gpio5** = Pin<rp2040\_hal::gpio::pin... [00]
      - > **gpio6** = Pin<rp2040\_hal::gpio::pin::ba...
      - > **gpio7** = Pin<rp2040\_hal::gpio::pin::ba...
  - WATCH** section (empty).
- SIDE TOOLBAR**: Includes icons for file, search, and other debugger functions.

# Blinky with PWM

- We can use Pulse Width Modulation to change the intensity of the LED



- Go to rp2040-project-template, clone

```
git clone https://github.com/rp-rs/rp2040-project-template.git
```

- Rename the project template folder to `blinky_pwm`

Wikipedia contributors. "Pulse-width modulation." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 24 Mar. 2025. Web. 28 Mar. 2025.



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

26

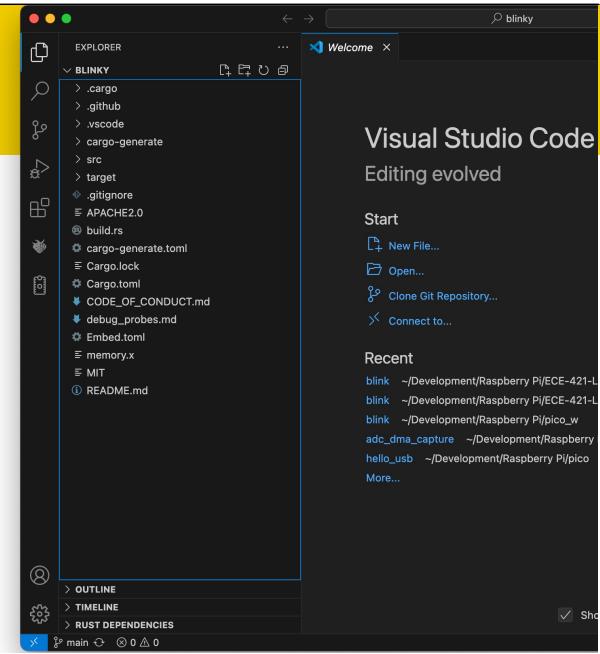
# VSCode work

- Open the `blinky` folder in VSCode
- We will be working in some of the folders to modify the template defaults



UNIVERSITY  
OF ALBERTA

Winter 2025



27

## VSCode work

- Check in .cargo/config.toml that

```
runner = "probe-rs run --chip RP2040 --protocol swd"
```

- In Cargo.toml rename the project to `blinky_pwm`
- In .vscode/launch.json change “name” to `blinky_pwm` (line 9) and on (line 26)

```
"programBinary": "target/thumbv6m-none-eabi/debug/rp2040-project-template"
```

to

```
"programBinary": "target/thumbv6m-none-eabi/debug/blinky_pwm"
```



UNIVERSITY  
OF ALBERTA

Winter 2025

ECE 421

28

## VSCode work cont'd

- In .vscode/launch.json, uncomment the line

```
"svdFile": "./.vscode/rp2040.svd"
```

- Manually download rp2040.svd from

```
curl -O https://raw.githubusercontent.com/raspberrypi/pico-sdk/1.3.1/src/rp2040/hardware_regs/rp2040.svd
```

and place it in the .vscode directory so that the debugger can find it



## VSCode work cont'd

- In VSCode, install the Debugger for probe-rs extension (should be done already)
- Make sure we have panic-halt : cargo add panic-halt
- Run main.rs with downloaded main.rs
- Time to run the program!
  - Won't use the debugger, so just select Run/Run Without Debugging



## Questions

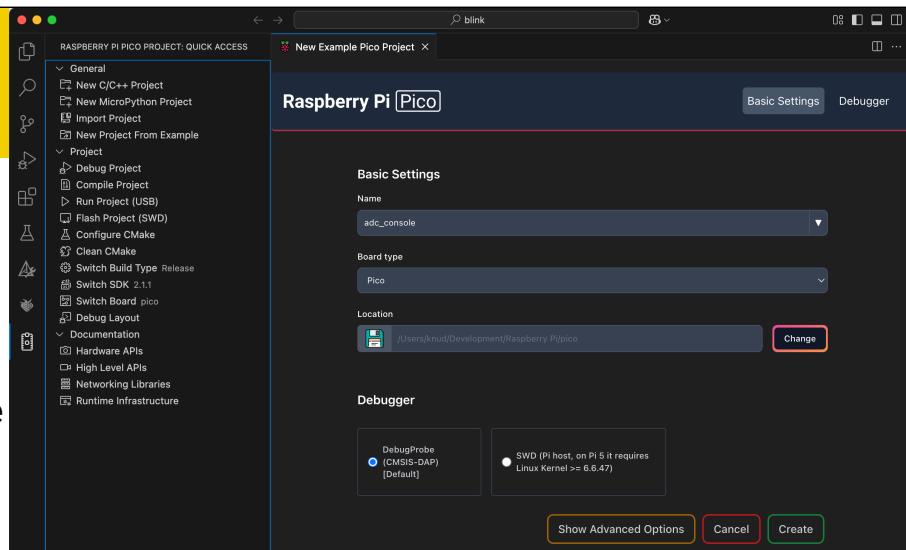
FOCUS



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

# ADC

- Let's try another example.
- Analogue to Digital converter
- Note, back to the Pico, not Pico W

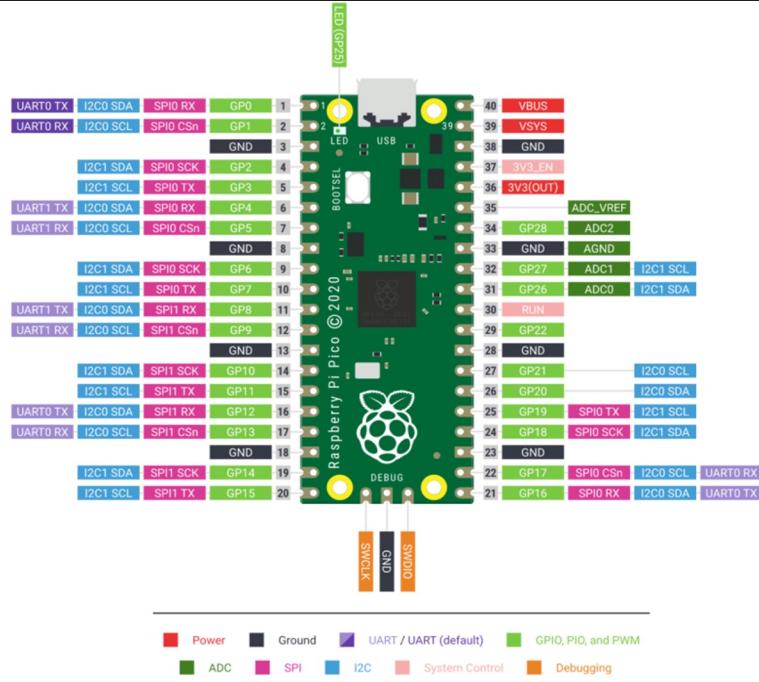


```
1 #include <stdio.h>
2 #include "pico/stlplib.h"
3 #include "hardware/gpio.h"
4 #include "hardware/adc.h"
5
6 #define N_SAMPLES 1000
7 uint16_t sample_buf[N_SAMPLES];
8
9 void printhelp() {
10     puts("\nCommands:");
11     puts("c<n>: Select ADC channel n");
12     puts("s<t>: Sample once");
13     puts("S<t>: Sample many");
14     puts("w<t>: Wiggle pins");
15 }
16
17 void __not_in_flash_func(adc_capture)(uint16_t *buf, size_t count) {
18     adc_fifo_setup(true, false, 0, false, false);
19     adc_run(true);
20     for (size_t i = 0; i < count; i = i + 1)
21         buf[i] = adc_fifo_get_blocking();
22     adc_run(false);
23     adc_fifo_drain();
24 }
25
26 int main(void) {
27     stdio_init_all();
28     adc_init();
29     adc_set_temp_sensor_enabled(true);
30
31     // Set all pins to input (as far as SIO is concerned)
32     gpio_set_dir_all_bits(0);
33     for (int i = 2; i < 30; ++i) {
34         gpio_set_function(i, GPIO_FUNC_SIO);
35         if (i >= 26) {
36             gpio_disable_pulls(i);
37             gpio_set_input_enabled(i, false);
38         }
39     }
40 }
```

Ln 24, Col 2 Spaces: 4 UTF-8 LF {} C ⚙️ Compile ▶ Run Pico SDK: 2.1.1 Board: pico\_w Pico

# Can't debug

- The program relies on terminal I/O
  - Asks for the ADC pin to use
- Where are ADC channels?
- Pin 31, 32, 34
- Pin 33 is AGND



## VSCode work cont'd

- In Cargo.toml, uncomment the two lines with rp2040\*

```
# If you're not going to use a Board Support Package you'll need these:  
rp2040-hal = { version="0.10", features=["rt", "critical-section-impl"] }  
rp2040-boot2 = "0.3"
```

so that the HAL is included

- In VSCode, install the Debugger for probe-rs extension
- Time to run the debugger!



Winter 2025

ECE 421

35