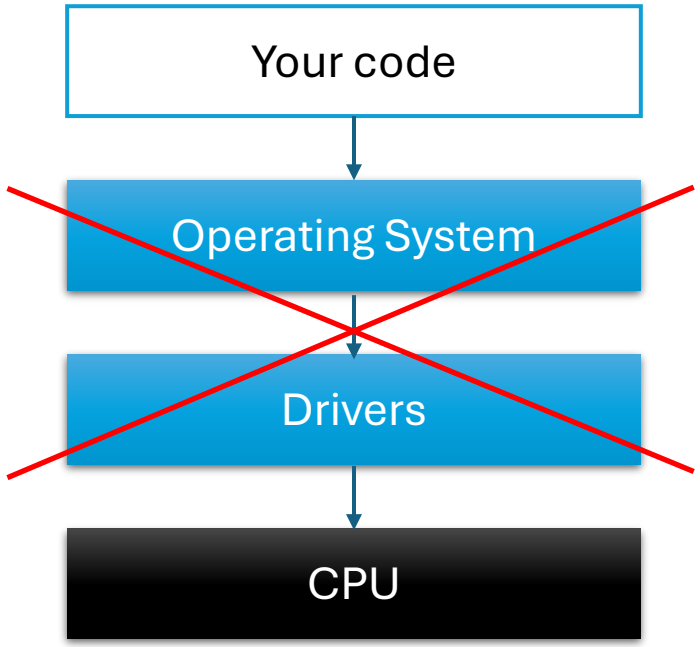
Introduction to Bare Metal Coding

with Olve Maudal

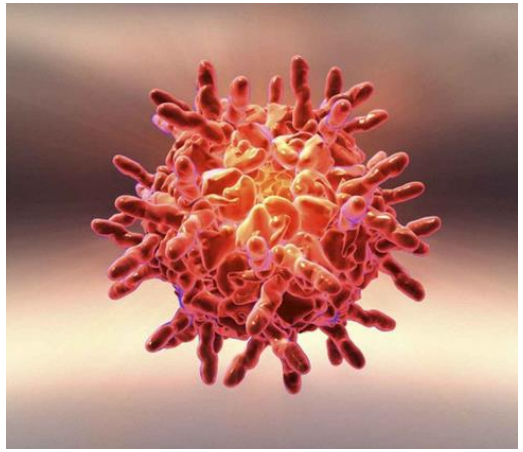


A 90 minute mini-course at GET Academy, May 2025

- Round of introduction (10 min)
- Reflection and discussion (10 min) - ultra-low-power "smart collar" for sheep
- Demo (15 min) - development board, bare metal coding
- Exercise (15 min) - "Hello, world!" on JSLinux
- Presentation (10 min) - Embedded, C, Assembler, C++, Examples, tiobe
- Demo (15 min) - microcontroller, blinky, bare metal C, pure SDK example, Arduino C, microPython
- Ad-hoc discussions / Summary / Wrap up (15 min)

Round of introduction

- Your name, and where are you right now?
- Favorite development environment, operating system and programming language?
- Any experience with C, assembler, microchips and bare metal programming?



Exercise: ultra-low-power "smart collar" for sheep

A fast-spreading virus threatens Australia’s sheep. If infected animals are identified within months, treatment works; if not, entire flocks could be lost.

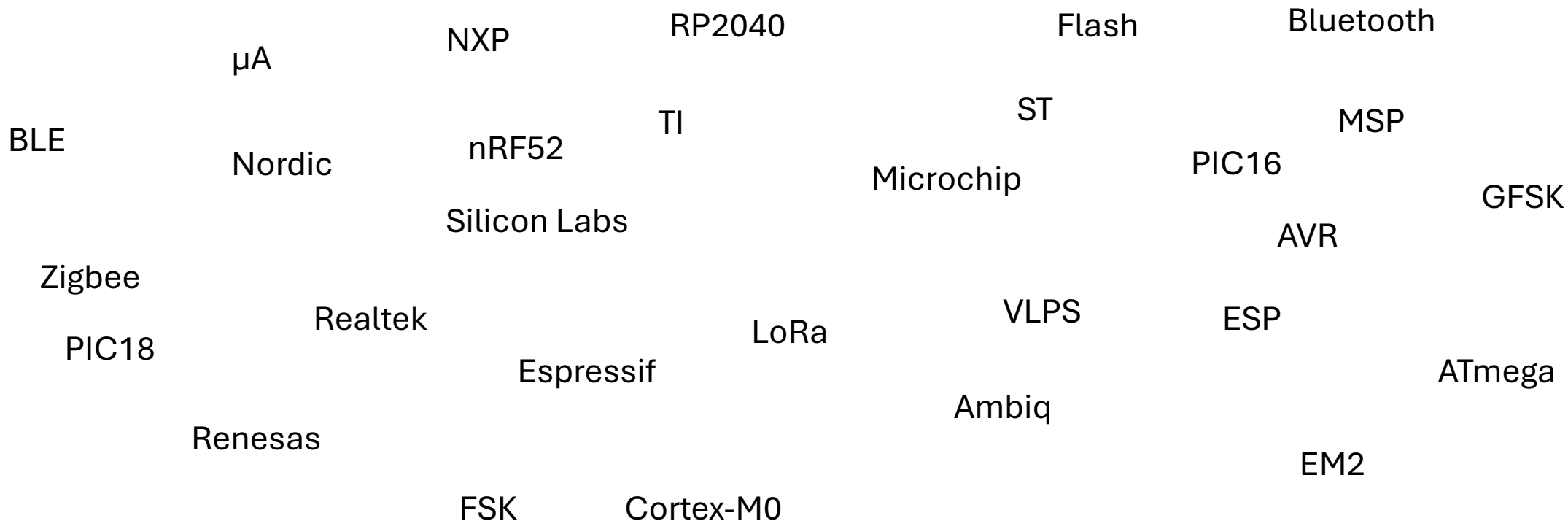
The federal government has asked your brilliant start-up company to deliver 100 working collars in 4 weeks and a concept plan within 7 days.

Each collar must:

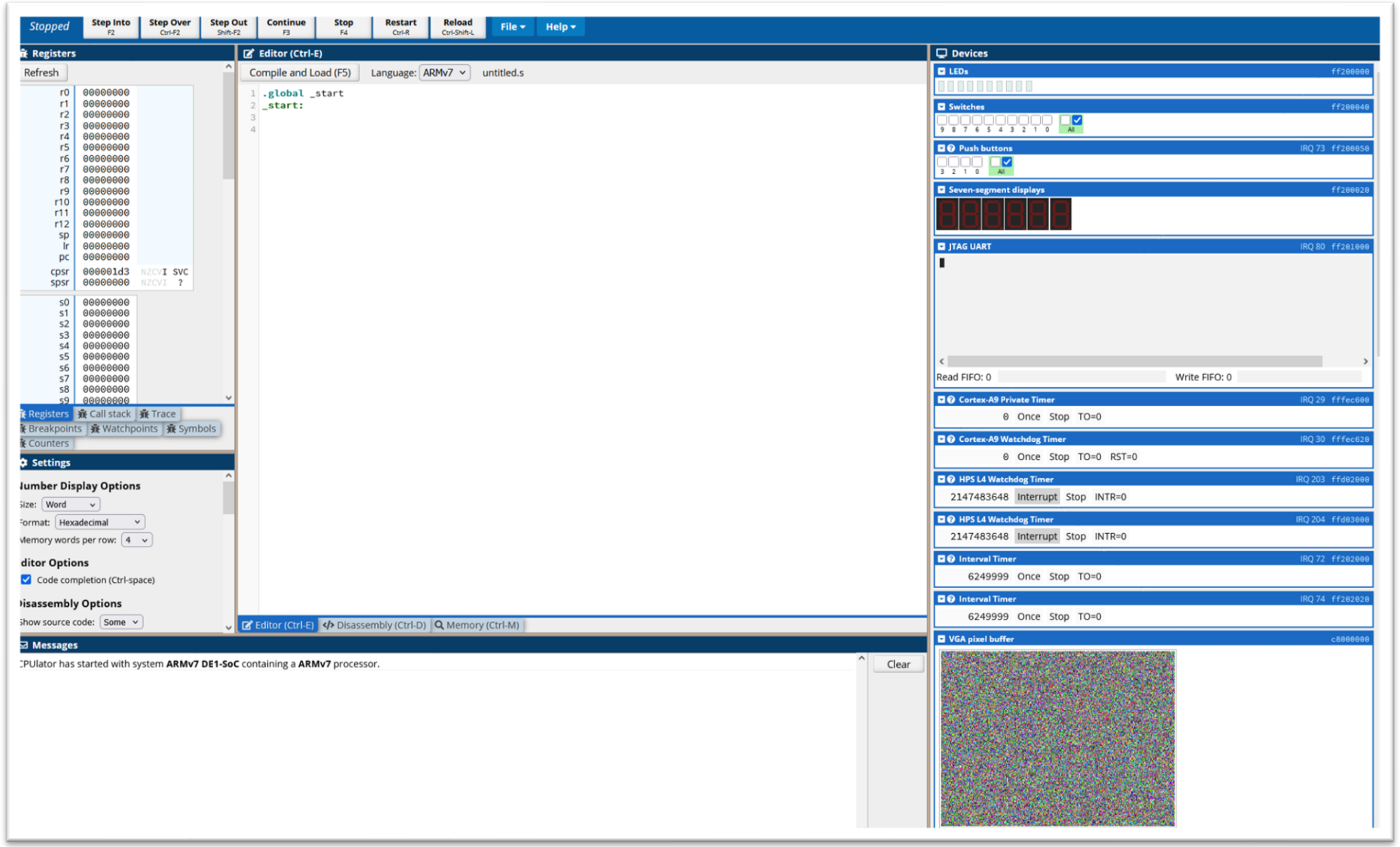
- Uniquely identify its wearer (128-bit UUID)
- Log the last 10 000 sheep encountered within approx 5 m proximity
- Upload the log automatically when a reading station is within 100 m
- Run for at least 12 months on a battery and survive rain, dust and head-butts

If trials succeed, production will scale to 1 million units per week at a bill-of-materials (BOM) under €5.

Focus on the smart collar - let's have a brainstorming session…


µA
NXP
RP2040
Flash
Bluetooth
BLE
Nordic
nRF52
TI
ST
MSP
PIC16
Microchip
GFSK
Silicon Labs
AVR
Zigbee
Realtek
VLPS
ESP
PIC18
LoRa
Espressif
ATmega
Ambiq
Renesas
EM2
FSK
Cortex-M0

CPUlator DE1SOC



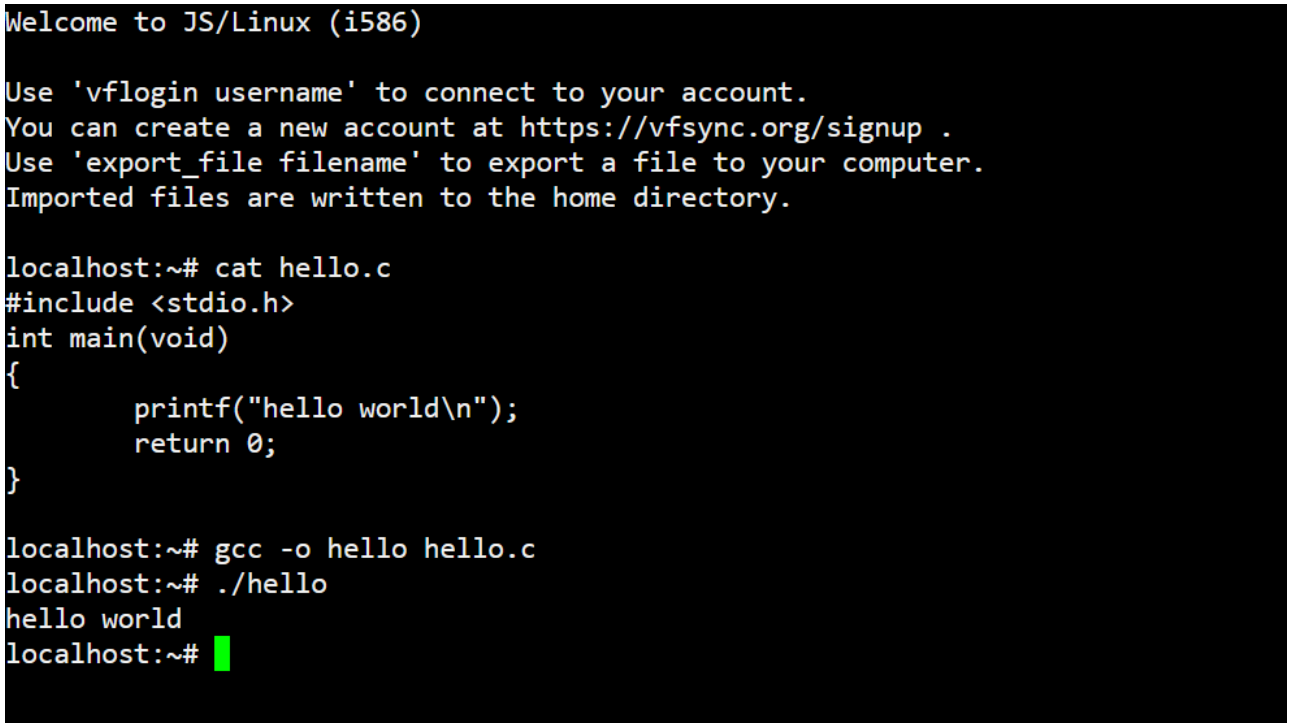
https://cpulator.01xz.net/?sys=arm-de1soc

```
.global _start

_start:
            ldr r0, =0xff200020
            ldr r1, =1
forever:
            str r1, [r0]
            add r1, #1
            bl delay
            b forever

delay:
            mov r2, #10000
waitloop:
            sub r2, #1
            cmp r2, #0
            bne waitloop
            bx lr
```

```
.global _start

.equ SWITCHES_BASE, 0xff200040
.equ DISPLAY_BASE, 0xff200020

_start:
            ldr r0, =SWITCHES_BASE
            ldr r1, =DISPLAY_BASE
forever:
            ldr r2, [r0]
            str r2, [r1]
            b forever
```

```
#include <stdint.h>
#include <stdio.h>

uint32_t * SWITCHES_BASE = (uint32_t*)0xff200040;
uint32_t * DISPLAY_BASE = (uint32_t*)0xff200020;

int main(void)
{
    uint32_t status = 0;
    uint32_t prev_status = 0;
    while (1) {
        status = *SWITCHES_BASE;
        *DISPLAY_BASE = status;
        if (status != prev_status) {
            printf("%08lx\n", status);
            prev_status = status;
        }
    }
}
```

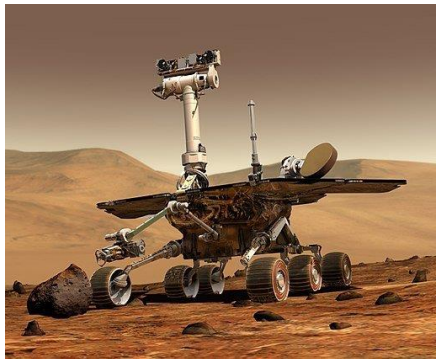
Hello, world!

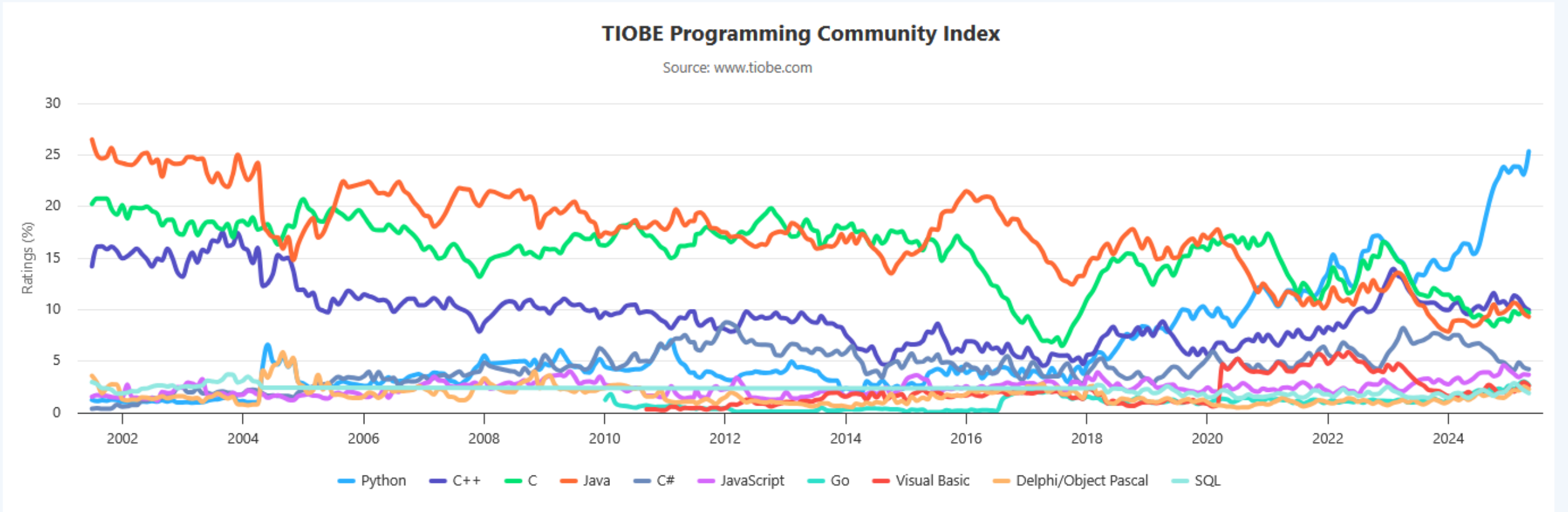



```c
#include <stdio.h>

int main(void)
{
    for (int n = 0; n < 20; n++) {
        int d = 2;
        while (d < n && n % d != 0)
            ++d;
        if (d == n)
            printf("%d\n", n);
    }
}
```

0) Visit https://bellard.org/jslinux/ and start x86 console emulator
1) Compile `hello.c` and run the program
2) Use the `nano` editor ("`nano hello.c`") to change the output string, recompile and run
3) Extra: type in the program on the right and try to figure out what it does

Why C





https://www.tiobe.com/tiobe-index/

```c
#include <stdio.h>
#include <assert.h>
#include <stdbool.h>

static bool is_divisible(int numerator, int denominator) {
    return numerator % denominator == 0;
}

static bool isprime(int number)
{
    if (number < 2)
        return false;
    int denominator = 2;
    while (denominator < number) {
        if (is_divisible(number, denominator))
            return false;
        denominator += 1;
    }
    return true;
}

static int count_primes_in_range(int begin, int end)
{
    int count = 0;
    for (int n = begin; n < end; n++)
        if (isprime(n))
            count++;
    return count;
}

int main(void)
{
    assert(1 + 2 == 3);
    assert(isprime(3) == true);
    assert(isprime(4) == false);
    assert(isprime(5) == true);
    assert(isprime(2) == true);
    assert(isprime(1) == false);
    assert(count_primes_in_range(0,10) == 4);
    assert(count_primes_in_range(0,20) == 8);
    fprintf(stderr, "All tests passed\n");
    return 0;
}
```

```
$ cc -std=c17 -Wall -Wextra -pedantic count_primes.c && ./a.out
All tests passed
$
```

primelib.h

```
#ifndef PRIMELIB_H_INCLUDED
#define PRIMELIB_H_INCLUDED

#include <stdbool.h>

bool isprime(int number);
int count_primes(int begin, int end);

#endif
```

primelib.c

```
#include "primelib.h"

static bool superslow_isprime(int number)
{
    if (number < 2)
        return false;
    int denominator = 2;
    while (denominator < number) {
        if (number % denominator == 0)
            return false;
        ++denominator;
    }
    return true;
}

bool isprime(int number)
{
    return superslow_isprime(number);
}

int count_primes(int begin, int end)
{
    int count = 0;
    for (int n = begin; n < end; ++n)
        if (isprime(n))
            ++count;
    return count;
}
```

primelib_tests.c

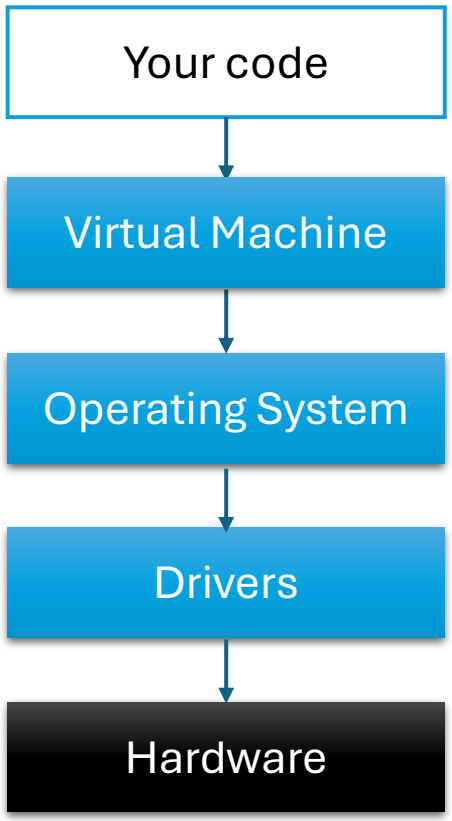
```
#include "primelib.h"

#include <assert.h>
#include <stdio.h>

int main(void)
{
    assert(isprime(0) == false);
    assert(isprime(1) == false);
    assert(isprime(2) == true);
    assert(isprime(3) == true);
    assert(isprime(4) == false);
    assert(isprime(5) == true);
    assert(count_primes(0,10) == 4);
    assert(count_primes(2,10) == 4);
    assert(count_primes(10,20) == 4);
    assert(count_primes(11,19) == 3);
    assert(count_primes(1,20) == 8);
    assert(count_primes(1000,2000) == 135);

    fprintf(stderr, "All tests passed\n");
    return 0;
}
```

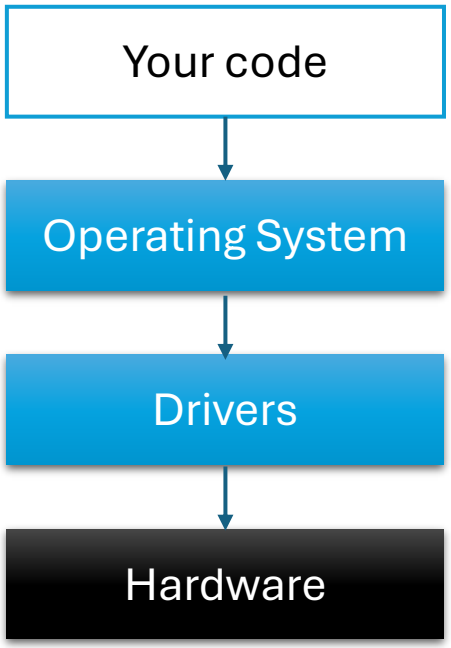
```
$ cc -c primelib.c
$ cc -o primelib_tests primelib_tests.c primelib.o
$ ./primelib_tests
All tests passed
$ cc -o primedemo primedemo.c primelib.o
$ ./primedemo
Number of primes in range [4000, 5000) is 119
$ echo $?
0
$
```

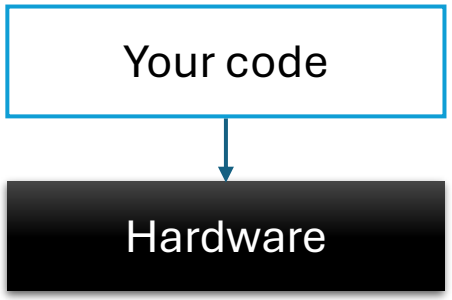
primedemo.c

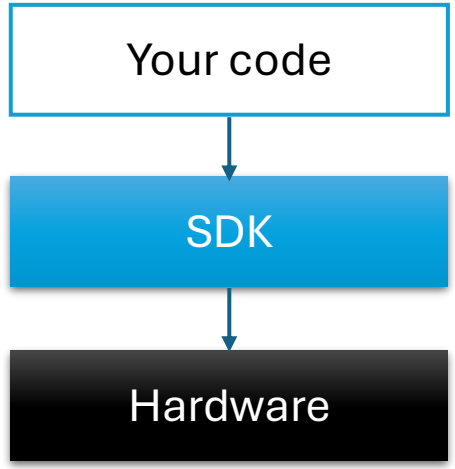
```
#include "primelib.h"

#include <stdio.h>

int main(void)
{
    int lower_bound = 4000;
    int upper_bound = 5000;

    printf("Number of primes in range [%d, %d) is ", lower_bound, upper_bound);
    fflush(stdout);

    int number_of_primes = count_primes_in_range(lower_bound, upper_bound);

    printf("%d\n", number_of_primes);

    return 0;
}
```

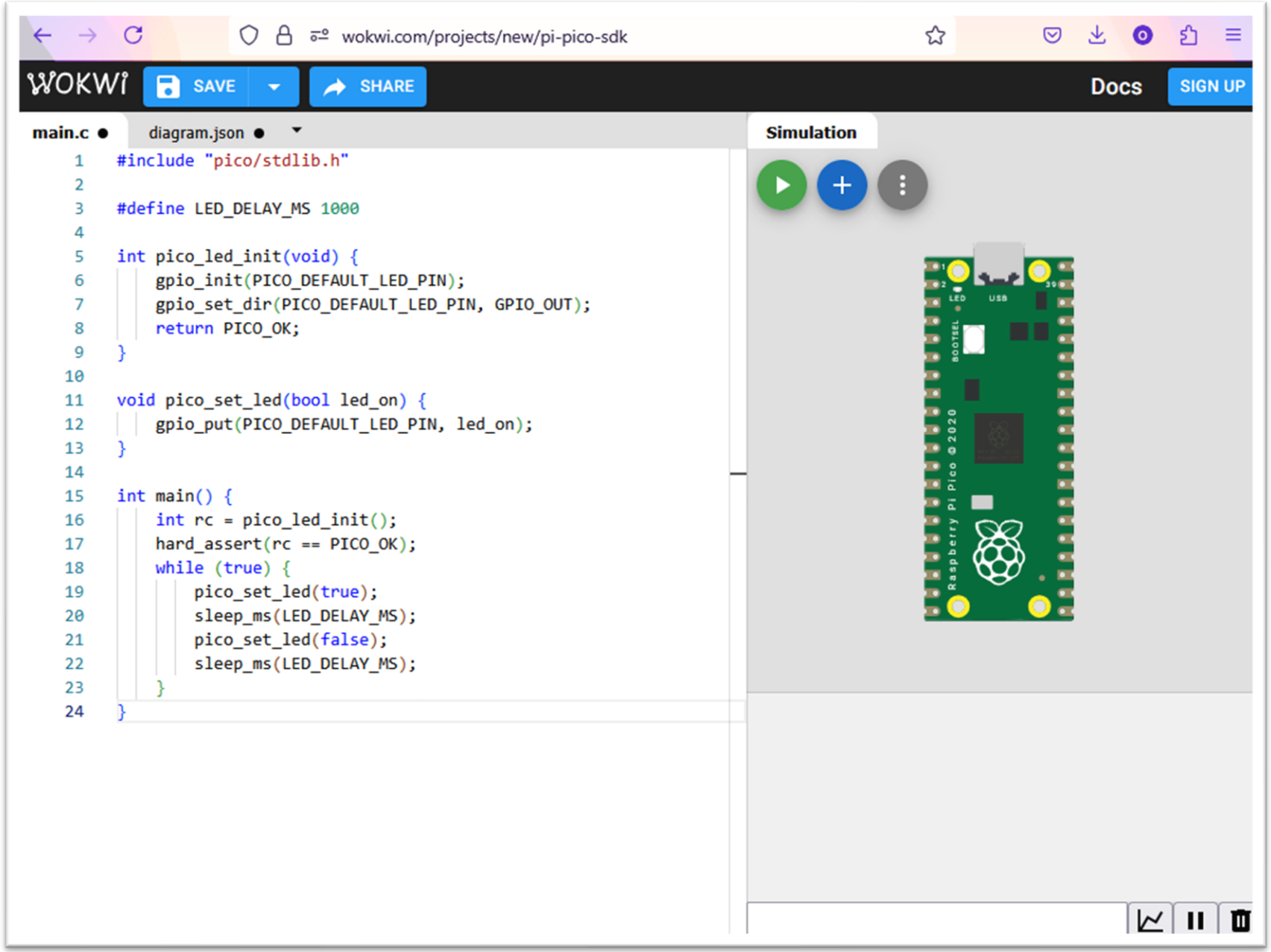

Raspberry Pi Pico
© 2020
BOOTSEL
LED
USB
DEBUG


Your code
Virtual Machine
Operating System
Drivers
Hardware


Your code
Operating System
Drivers
Hardware


Your code
Hardware

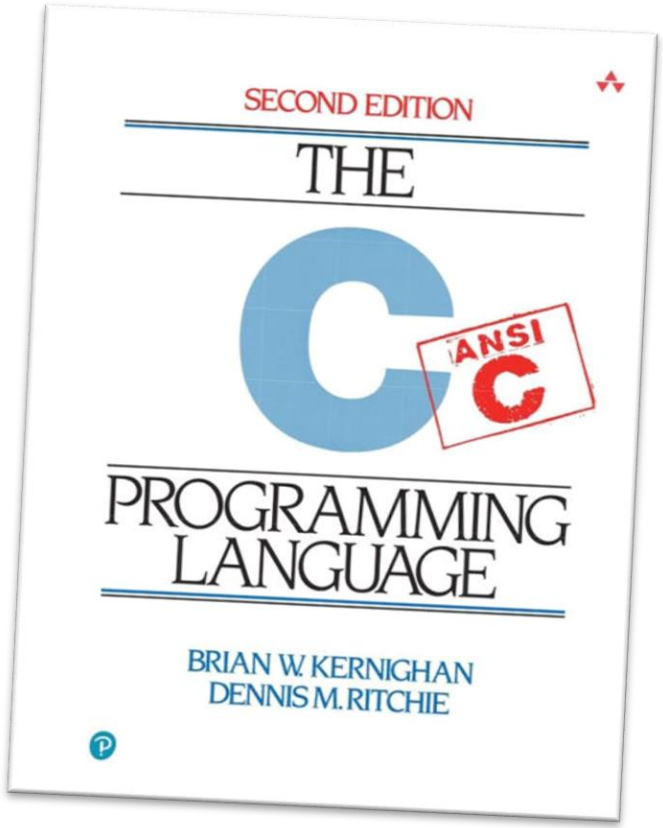

Your code
SDK
Hardware

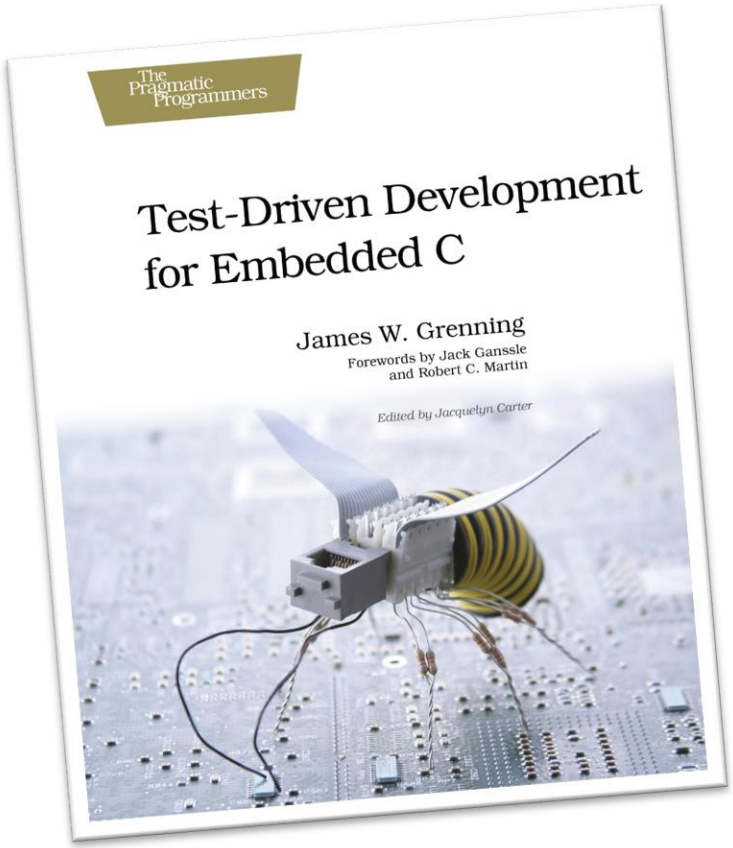

wokwi.com/projects/new/pi-pico-sdk
WOKWI
SAVE
SHARE
Docs
SIGN UP
main.c
diagram.json
Simulation
#include "pico/stdlib.h"
#define LED_DELAY_MS 1000
int pico_led_init(void) {
gpio_init(PICO_DEFAULT_LED_PIN);
gpio_set_dir(PICO_DEFAULT_LED_PIN, GPIO_OUT);
return PICO_OK;
}
void pico_set_led(bool led_on) {
gpio_put(PICO_DEFAULT_LED_PIN, led_on);
}
int main() {
int rc = pico_led_init();
hard_assert(rc == PICO_OK);
while (true) {
pico_set_led(true);
sleep_ms(LED_DELAY_MS);
pico_set_led(false);
sleep_ms(LED_DELAY_MS);
}
}
LED
USB
BOOTSEL
Raspberry Pi Pico © 2020

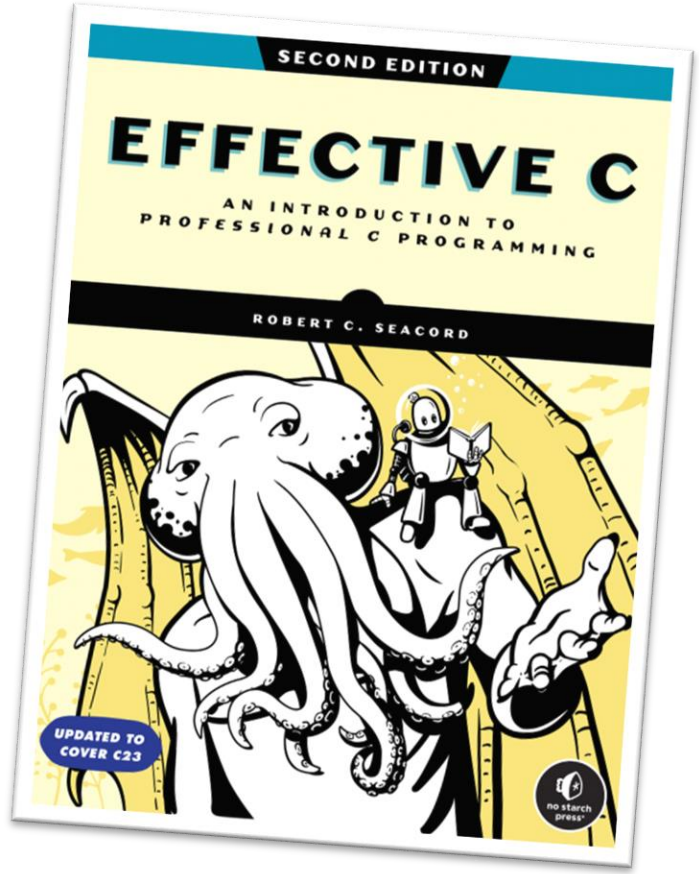
```c
#include "pico/stdlib.h"

#define LED_DELAY_MS 1000

int pico_led_init(void) {
    gpio_init(PICO_DEFAULT_LED_PIN);
    gpio_set_dir(PICO_DEFAULT_LED_PIN, GPIO_OUT);
    return PICO_OK;
}

void pico_set_led(bool led_on) {
    gpio_put(PICO_DEFAULT_LED_PIN, led_on);
}

int main() {
    int rc = pico_led_init();
    hard_assert(rc == PICO_OK);
    while (true) {
        pico_set_led(true);
        sleep_ms(LED_DELAY_MS);
        pico_set_led(false);
        sleep_ms(LED_DELAY_MS);
    }
}
```

- HackadayU: Raspberry Pi Pico and RP2040 (YouTube)
- RP2040 - Introduction to Assembly Language (YouTube)
- https://github.com/raspberrypi/pico-bootrom-rp2040
- https://github.com/raspberrypi/pico-sdk
- https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf
- https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf
- https://cpulator.01xz.net/?sys=arm-de1soc
- https://fpgacademy.org/Downloads/DE1-SoC_Computer_ARM.pdf
- https://developer.arm.com/documentation/dui0662/latest/
- Assembly Language Programming with ARM https://www.youtube.com/watch?v=gfmRrPjnEw4
- https://godbolt.org/
- Reintroduction to C https://www.youtube.com/watch?v=Kq7WIexp6JU

?







https://olvemaudal.com/talks/

!

About bootstraps and booting a computer

The phrase “pull yourself up by your bootstraps” originated shortly before the turn of the 20th century. It’s attributed to a late-1800s physics schoolbook that contained the example question “Why can not a man lift himself by pulling up on his bootstraps?”

…

This idiom is also the source of “booting” a computer, as well the mathematical Bootstrap Method.

https://uselessetymology.com/2019/11/07/the-origins-of-the-phrase-pull-yourself-up-by-your-bootstraps/