Hacking Embedded Rust

A Guide to Raspberry Pi Pico Development with Embassy

Kevin Thomas

June 7, 2025

Contents

Rust Embassy Pico Project

A simple embedded Rust project running on the Raspberry Pi Pico (RP2040), built with Embassy async framework and no_std runtime.

[1] https://github.com/mytechnotalent/Reverse-Engineering-Tutorial

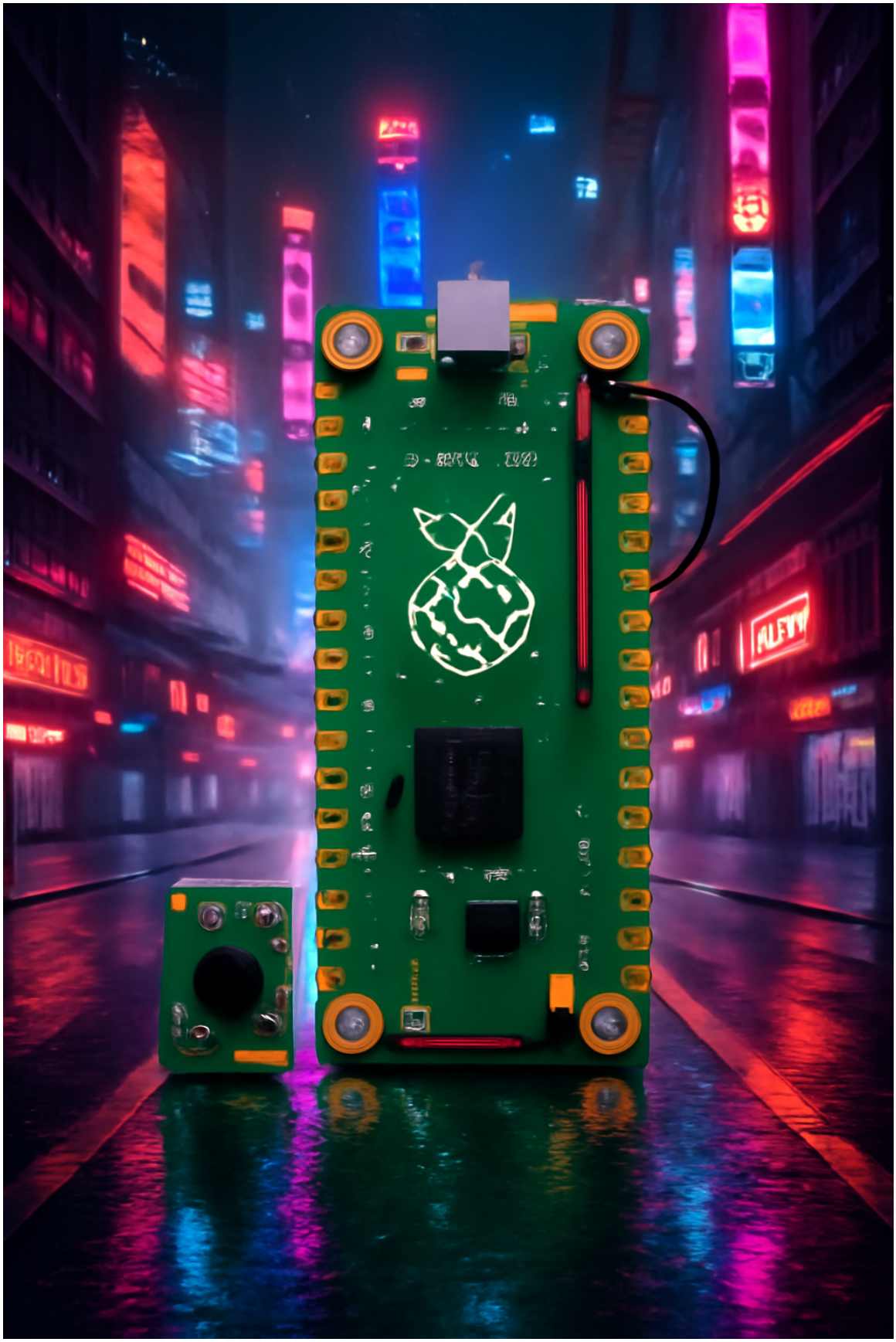
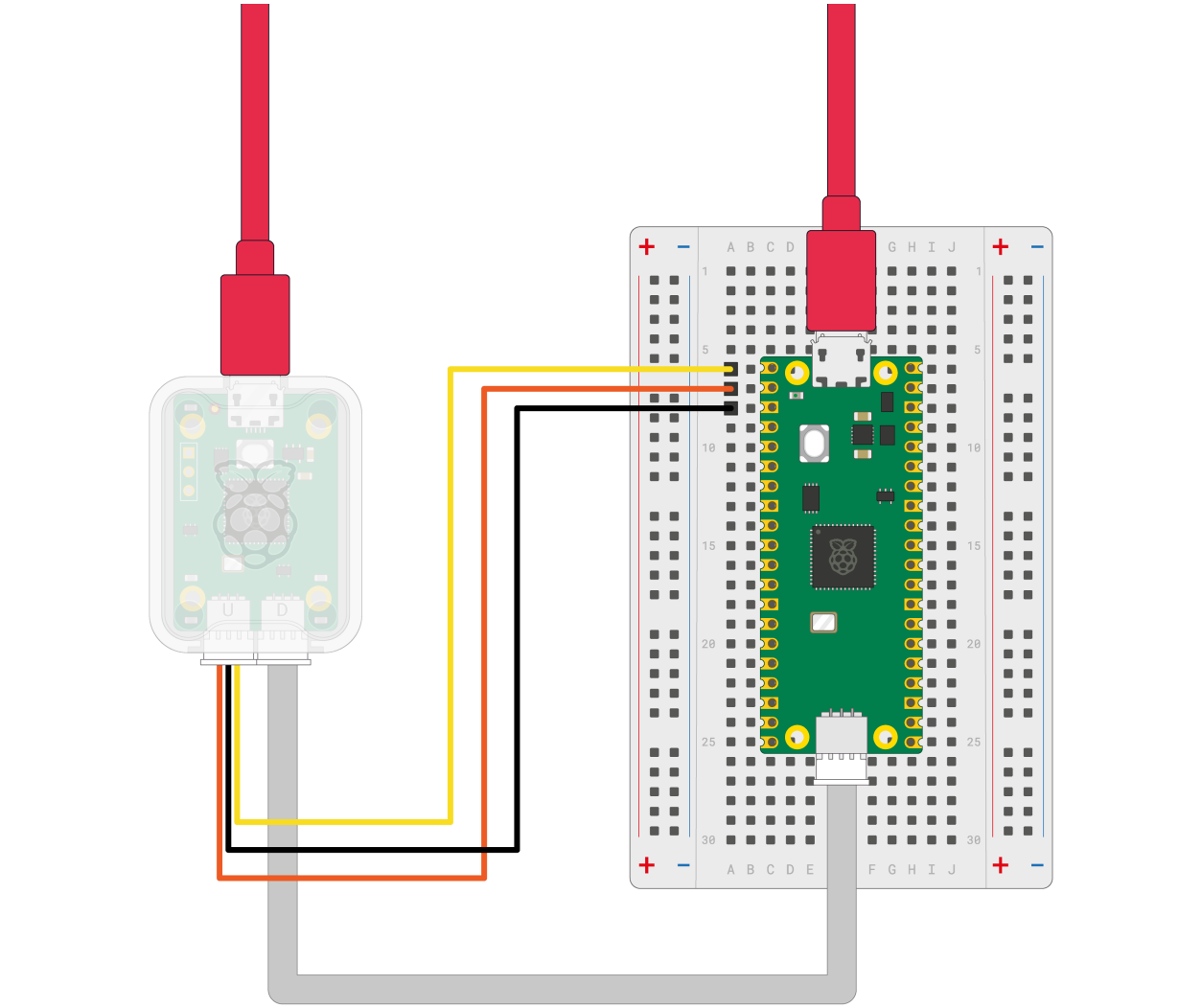
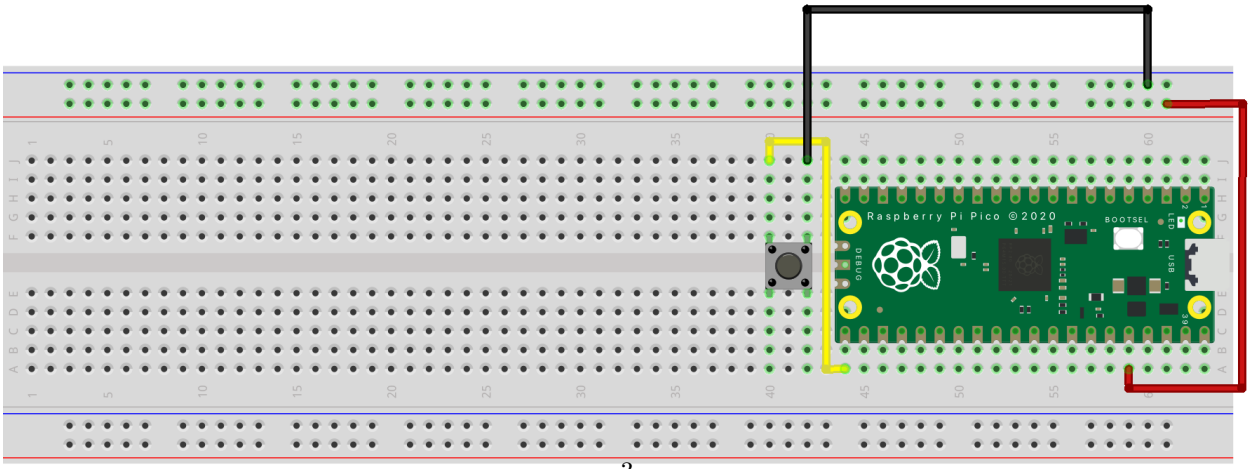


Figure 1: image

FREE Reverse Engineering Self-Study Course HERE[2]

Wiring





fritzing

[2] https://github.com/mytechnotalent/Reverse-Engineering-Tutorial

Features

- Configures an onboard LED (GPIO25).
- Configures a button (GPIO16) with internal pull-up.
- Turns LED on when button is pressed, off when released.
- Debounces the button with async timer.
- Runs under Embassy's async executor with no RTOS, no heap.

Project Structure

- `main.rs`: Initializes Embassy, spawns the main async task.
- `button.rs`: Provides button GPIO initialization.
- `led.rs`: Provides simple onboard LED control abstraction.
- `run_cycle.rs`: Defines a single button-press LED-control cycle.

How It Works (Step-by-Step)

1. **Startup**
   - The RP2040 boot ROM loads your program from flash.
   - The Cortex-M `cortex-m-rt` runtime (`#[no_main]`) skips the traditional `main()` and jumps into the reset vector.
   - The `__cortex_m_rt_main_trampoline` is called automatically at startup.
   - `__cortex_m_rt_main_trampoline` calls `__cortex_m_rt_main`, which manually initializes and starts the async executor.
2. **Executor Initialization**
   - A static instance of `Executor` is created by `transmute`-ing a stack object to `'static` lifetime.
   - `executor.run()` is called, entering Embassy's async runtime loop.
   - A `Spawner` is provided by the executor, allowing you to spawn tasks.
3. **Task Spawning (Enqueue Operation)**
   - `Spawner::must_spawn(__embassy_main(spawner))` is called.
   - The `__embassy_main_task` future is created and wrapped into a `Task`.
   - **Enqueue**:
     - The Task is added to the **Task Queue**, a statically allocated double-ended queue (deque) implemented internally by the Executor.
     - This is a lock-free queue; in single-core systems like RP2040, no locks are needed.
     - Enqueue happens at the **tail** (back) of the queue — FIFO behavior is preserved.
4. **Executor Main Loop (Deque + Polling)**
   - The Executor enters its main loop:
     - **Dequeue**:
       - A Task is popped from the **head** (front) of the queue.
     - The Executor **polls** the Task's future by calling its `poll()` method.
       - If the Future returns `Poll::Pending`, it means it cannot complete immediately:
         - The Task registers a `Waker` — when an awaited event (e.g., timer, GPIO interrupt) completes, the Waker re-schedules the task.
         - The task is then re-**enqueued** at the tail for future polling.
       - If the Future returns `Poll::Ready`, the Task has completed:
         - The Task is **dropped** and removed permanently from the system.
     - If there are no tasks left in the queue, the Executor executes a `WFI` (Wait-For-Interrupt) instruction, entering low-power sleep until an interrupt occurs.
5. **Peripheral Setup**
   - In the spawned `____embassy_main_task`:
     - `embassy_rp::init()` is called to set up clocks, watchdog, and peripherals.
     - `PIN_16` is configured as an input GPIO with an internal pull-up resistor (for the button).
     - `PIN_25` is configured as an output GPIO (for the onboard LED).
6. **Task Execution**

- Inside the `loop {}`:
  - `run_cycle` is called and awaited:
    - Waits for a button press (falling edge detected on the input pin).
    - Turns the LED on by driving `PIN_25` high.
    - Waits for a button release (rising edge).
    - Turns the LED off by driving `PIN_25` low.
    - Waits 10ms for debounce using `Timer::after_millis`.
  - `run_cycle` returns `Poll::Ready`, but since it's inside an infinite loop, a new future is immediately constructed for the next cycle.
- As `run_cycle` awaits on GPIO events and timers, the task yields control back to the Executor, causing the task to be **re-enqueued** and other pending tasks (if any) to be polled.

7. **Event Handling and Re-Scheduling**
   - When the awaited GPIO or timer event completes:
     - The Task's registered `Waker` is triggered.
     - The Task is **re-enqueued** into the Task Queue tail.
     - On the next executor loop iteration, it will be **dequeued** and `poll()` will resume where it left off in its await.
8. **Continuous Loop**
   - The `____embassy_main_task` is never terminated due to its infinite loop.
   - This cycle continues indefinitely, reacting to button presses/releases, toggling the LED accordingly.

Embassy Executor Enqueue/Dequeue In Detail

- **Enqueue (Push-Back)**:
  - When a `Future` yields `Poll::Pending`, the Task's Waker will call `spawn()`.
  - Internally, this pushes the Task to the **back** of the Task Queue.
  - The Task Queue is lock-free, array-backed with bounded capacity.
  - Waker ensures the task is only enqueued if it was not already enqueued (no duplication).
- **Dequeue (Pop-Front)**:
  - Executor pops a Task from the **front** of the queue (FIFO order).
  - Calls `poll()` on the Task.
  - If `Poll::Pending`, the Task will re-enqueue after its awaited event is ready.
  - If `Poll::Ready`, the Task is cleaned up and removed.
- **Task Scheduling**:
  - Tasks are cooperatively scheduled.
  - No preemption — a task must yield (await) to allow others to run.
  - If all tasks are `Pending`, Executor enters WFI (low-power wait).
- **Wakers**:
  - Embassy provides a lightweight Waker implementation tied to the Task.
  - When a peripheral (e.g., Timer or GPIO interrupt) completes, the Waker triggers the task re-enqueue.
- **No Dynamic Memory**:
  - All Tasks are statically allocated.
  - The queue and task structures are baked into flash/ram at compile time.
  - Ensures no heap fragmentation and determinism — critical for embedded systems.

Deep Dive: From Reset to `run_cycle` Poll

Below is **every** assembler instruction captured, with mangled demangled names and addresses.

1. Reset Vector (`Reset` @ 0x100001c0)

```
0x100001c0 <+0>:   bl      0x10007a94 <__pre_init>
0x100001c4 <+4>:   ldr     r0, [pc, #32]   @ (0x100001e8)
0x100001c6 <+6>:   ldr     r1, [pc, #36]   @ (0x100001ec)
0x100001c8 <+8>:   movs    r2, #0
0x100001ca <+10>:  cmp     r1, r0
0x100001cc <+12>:  beq.n   0x100001d2 <Reset+18>
0x100001ce <+14>:  stmia   r0!, {r2}
0x100001d0 <+16>:  b.n     0x100001ca <Reset+10>
0x100001d2 <+18>:  ldr     r0, [pc, #28]   @ (0x100001f0)
0x100001d4 <+20>:  ldr     r1, [pc, #28]   @ (0x100001f4)
0x100001d6 <+22>:  ldr     r2, [pc, #32]   @ (0x100001f8)
0x100001d8 <+24>:  cmp     r1, r0
0x100001da <+26>:  beq.n   0x100001e2 <Reset+34>
0x100001dc <+28>:  ldmia   r2!, {r3}
0x100001de <+30>:  stmia   r0!, {r3}
0x100001e0 <+32>:  b.n     0x100001d8 <Reset+24>
0x100001e2 <+34>:  bl      0x1000066c <main>
0x100001e6 <+38>:  udf     #0
```

2. `main` Trampoline (@ 0x1000066c)

```
0x1000066c <+0>:   push    {r7, lr}
0x1000066e <+2>:   add     r7, sp, #0
0x10000670 <+4>:   bl      0x10000674 <_ZN25rust_embassy_pico_project18__cortex_m_rt_main17h…>
```

3. `__cortex_m_rt_main` (@ 0x10000674)

```
0x10000674 <+0>:   push    {r7, lr}
0x10000676 <+2>:   add     r7, sp, #0
0x10000678 <+4>:   sub     sp, #16
0x1000067a <+6>:   bl      0x10009a20 <_ZN16embassy_executor4arch6thread8Executor3new17h…>
0x1000067e <+10>:  str     r0, [sp, #4]
0x10000680 <+12>:  str     r1, [sp, #8]
0x10000682 <+14>:  add     r0, sp, #4
0x10000684 <+16>:  bl      0x10000e24 <_ZN25rust_embassy_pico_project18__cortex_m_rt_main13__make_stati
0x10000688 <+20>:  str     r0, [sp, #12]
0x1000068a <+22>:  bl      0x1000163c <_ZN16embassy_executor4arch6thread8Executor3run17h…>
```

4. `Executor::new` → `raw::Executor::new` → `SyncExecutor::new`

- **Executor::new** (`arch::cortex_m.rs:77`): calls `raw::Executor::new(THREAD_PENDER)`
- **raw::Executor::new** (mod.rs:495): creates `SyncExecutor::new`
- **SyncExecutor::new** (mod.rs:383): `run_queue: RunQueue::new()`
- **RunQueue::new** (run_queue_critical_section.rs:35): `head: Mutex::new(Cell::new(None))`, etc.

5. `Executor::run` (@ 0x1000163c)

```
0x1000163c <+0>:   push    {r7, lr}
0x1000163e <+2>:   add     r7, sp, #0
0x10001640 <+4>:   sub     sp, #16
0x10001642 <+6>:   str     r0, [sp, #4]
```

```
0x10001644 <+8>:   str     r0, [sp, #8]
0x10001646 <+10>: bl      0x10009dba <raw::Executor::spawner>
0x1000164a <+14>: bl      0x10000e30 <__cortex_m_rt_main::{closure}>
0x10001650 <+20>: ldr     r0, [sp, #4]
0x10001652 <+22>: bl      0x10009daa <raw::Executor::poll>
0x10001656 <+26>: wfe
0x10001658 <+28>: b.n     0x10001650
```

6. `run_cycle` Future Constructor (@ 0x10001904)

```
0x10001904 <+0>:   push    {r7, lr}
0x10001906 <+2>:   add     r7, sp, #0
0x10001908 <+4>:   sub     sp, #12
0x1000190a <+6>:   str     r1, [sp, #0]     # button ptr
0x1000190c <+8>:   mov     r1, r0
0x1000190e <+10>:  ldr     r0, [sp, #0]     # button
0x10001910 <+12>:  str     r0, [sp, #4]     # store in struct
0x10001912 <+14>:  str     r2, [sp, #8]     # store led ptr
0x10001914 <+16>:  str     r0, [r1, #0]
0x10001916 <+18>:  str     r2, [r1, #4]
0x10001918 <+20>:  movs    r0, #0           # initial state=0
0x1000191a <+22>:  strb    r0, [r1, #16]    # .state=0
0x1000191c <+24>:  add     sp, #12
0x1000191e <+26>:  pop     {r7, pc}
```

7. `run_cycle` Poll State Machine (@ 0x100006a0)

```
0x100006a0 <+0>:     push    {r4, r6, r7, lr}
0x100006a2 <+2>:     add     r7, sp, #8
0x100006a4 <+4>:     sub     sp, #192        @ 0xc0
0x100006a6 <+6>:     str     r1, [sp, #20]
0x100006a8 <+8>:     str     r0, [sp, #28]
0x100006aa <+10>:    str     r1, [sp, #136]  @ 0x88
0x100006ac <+12>:    ldr     r0, [sp, #28]
0x100006ae <+14>:    ldrb    r0, [r0, #16]
0x100006b0 <+16>:    str     r0, [sp, #24]
0x100006b2 <+18>:    ldr     r0, [sp, #24]
0x100006b4 <+20>:    lsls    r1, r0, #2
0x100006b6 <+22>:    add     r0, pc, #4      @ (adr r0, 0x100006bc <_ZN25rust_embassy_pico_project9run_
0x100006b8 <+24>:    ldr     r0, [r0, r1]
0x100006ba <+26>:    mov     pc, r0
0x100006bc <+28>:    lsls    r3, r3, #27
0x100006be <+30>:    asrs    r0, r0, #32
0x100006c0 <+32>:    lsls    r1, r7, #27
0x100006c2 <+34>:    asrs    r0, r0, #32
0x100006c4 <+36>:    lsls    r1, r3, #27
0x100006c6 <+38>:    asrs    r0, r0, #32
0x100006c8 <+40>:    lsls    r1, r0, #28
0x100006ca <+42>:    asrs    r0, r0, #32
0x100006cc <+44>:    lsls    r7, r0, #28
0x100006ce <+46>:    asrs    r0, r0, #32
0x100006d0 <+48>:    lsls    r5, r1, #28
0x100006d2 <+50>:    asrs    r0, r0, #32
```

```
0x100006d4 <+52>:    lsls    r3, r2, #28
0x100006d6 <+54>:    asrs    r0, r0, #32
0x100006d8 <+56>:    udf     #254    @ 0xfe
0x100006da <+58>:    ldr     r0, [sp, #20]
0x100006dc <+60>:    str     r0, [sp, #132]  @ 0x84
0x100006de <+62>:    ldr     r1, [sp, #28]
0x100006e0 <+64>:    ldr     r0, [r1, #0]
0x100006e2 <+66>:    str     r0, [r1, #8]
0x100006e4 <+68>:    ldr     r1, [sp, #28]
0x100006e6 <+70>:    ldr     r0, [r1, #4]
0x100006e8 <+72>:    str     r0, [r1, #12]
0x100006ea <+74>:    ldr     r0, [sp, #28]
0x100006ec <+76>:    ldr     r0, [r0, #12]
0x100006ee <+78>:    bl      0x100005d6 <_ZN10embassy_rp4gpio5Input6is_low17h279e6840f083881cE>
0x100006f2 <+82>:    cmp     r0, #0
0x100006f4 <+84>:    bne.n   0x10000740 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b$
0x100006f6 <+86>:    b.n     0x10000718 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b$
0x100006f8 <+88>:    movs    r0, #0
0x100006fa <+90>:    cmp     r0, #0
0x100006fc <+92>:    bne.n   0x100006f8 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b$
0x100006fe <+94>:    b.n     0x1000077a <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b$
0x10000700 <+96>:    ldr     r0, [sp, #20]
0x10000702 <+98>:    str     r0, [sp, #132]  @ 0x84
0x10000704 <+100>:   b.n     0x10000768 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b$
0x10000706 <+102>:   ldr     r0, [sp, #20]
0x10000708 <+104>:   str     r0, [sp, #132]  @ 0x84
0x1000070a <+106>:   b.n     0x1000079c <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b$
0x1000070c <+108>:   ldr     r0, [sp, #20]
0x1000070e <+110>:   str     r0, [sp, #132]  @ 0x84
0x10000710 <+112>:   b.n     0x100007f8 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b$
0x10000712 <+114>:   ldr     r0, [sp, #20]
0x10000714 <+116>:   str     r0, [sp, #132]  @ 0x84
0x10000716 <+118>:   b.n     0x1000085c <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b$
0x10000718 <+120>:   ldr     r0, [sp, #28]
0x1000071a <+122>:   ldr     r1, [r0, #12]
0x1000071c <+124>:   add     r0, sp, #84     @ 0x54
0x1000071e <+126>:   str     r0, [sp, #16]
0x10000720 <+128>:   bl      0x100005a6 <_ZN10embassy_rp4gpio5Input12wait_for_low17h2cb7b01158f39cecE>
0x10000724 <+132>:   ldr     r1, [sp, #16]
0x10000726 <+134>:   add     r0, sp, #68     @ 0x44
0x10000728 <+136>:   bl      0x10001378 <_ZN59_$LT$F$u20$as$u20$core..future..into_future..IntoFuture$G
0x1000072c <+140>:   ldr     r1, [sp, #28]
0x1000072e <+142>:   ldr     r0, [sp, #80]   @ 0x50
0x10000730 <+144>:   str     r0, [r1, #32]
0x10000732 <+146>:   ldr     r0, [sp, #76]   @ 0x4c
0x10000734 <+148>:   str     r0, [r1, #28]
0x10000736 <+150>:   ldr     r0, [sp, #72]   @ 0x48
0x10000738 <+152>:   str     r0, [r1, #24]
0x1000073a <+154>:   ldr     r0, [sp, #68]   @ 0x44
0x1000073c <+156>:   str     r0, [r1, #20]
0x1000073e <+158>:   b.n     0x1000079c <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b$
0x10000740 <+160>:   ldr     r0, [sp, #28]
0x10000742 <+162>:   ldr     r1, [r0, #12]
```

```
0x10000744 <+164>:   add     r0, sp, #52     @ 0x34
0x10000746 <+166>:   str     r0, [sp, #12]
0x10000748 <+168>:   bl      0x100005be <_ZN10embassy_rp4gpio5Input13wait_for_high17h9118f9d341070139E>
0x1000074c <+172>:   ldr     r1, [sp, #12]
0x1000074e <+174>:   add     r0, sp, #36     @ 0x24
0x10000750 <+176>:   bl      0x100013d6 <_ZN59_$LT$F$u20$as$u20$core..future..into_future..IntoFuture$G
0x10000754 <+180>:   ldr     r1, [sp, #28]
0x10000756 <+182>:   ldr     r0, [sp, #48]   @ 0x30
0x10000758 <+184>:   str     r0, [r1, #32]
0x1000075a <+186>:   ldr     r0, [sp, #44]   @ 0x2c
0x1000075c <+188>:   str     r0, [r1, #28]
0x1000075e <+190>:   ldr     r0, [sp, #40]   @ 0x28
0x10000760 <+192>:   str     r0, [r1, #24]
0x10000762 <+194>:   ldr     r0, [sp, #36]   @ 0x24
0x10000764 <+196>:   str     r0, [r1, #20]
0x10000766 <+198>:   b.n     0x10000768 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x10000768 <+200>:   ldr     r0, [sp, #28]
0x1000076a <+202>:   adds    r0, #20
0x1000076c <+204>:   str     r0, [sp, #188]  @ 0xbc
0x1000076e <+206>:   ldr     r1, [sp, #132]  @ 0x84
0x10000770 <+208>:   bl      0x10001274 <_ZN10embassy_rp4gpio5Input13wait_for_high28_$u7b$$u7b$closure$u
0x10000774 <+212>:   cmp     r0, #0
0x10000776 <+214>:   bne.n   0x10000780 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x10000778 <+216>:   b.n     0x10000792 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x1000077a <+218>:   ldr     r0, [pc, #280]  @ (0x10000894 <_ZN25rust_embassy_pico_project9run_cycle9ru
0x1000077c <+220>:   bl      0x1000b38c <_ZN4core9panicking11panic_const28panic_const_async_fn_resumed1
0x10000780 <+224>:   add     r0, sp, #32
0x10000782 <+226>:   movs    r1, #1
0x10000784 <+228>:   strb    r1, [r0, #0]
0x10000786 <+230>:   ldr     r2, [sp, #28]
0x10000788 <+232>:   movs    r1, #3
0x1000078a <+234>:   strb    r1, [r2, #16]
0x1000078c <+236>:   ldrb    r0, [r0, #0]
0x1000078e <+238>:   add     sp, #192        @ 0xc0
0x10000790 <+240>:   pop     {r4, r6, r7, pc}
0x10000792 <+242>:   ldr     r0, [sp, #28]
0x10000794 <+244>:   adds    r0, #20
0x10000796 <+246>:   bl      0x10001ae8 <_ZN4core3ptr88drop_in_place$LT$embassy_rp..gpio..Input..wait_f
0x1000079a <+250>:   b.n     0x10000718 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x1000079c <+252>:   ldr     r0, [sp, #28]
0x1000079e <+254>:   adds    r0, #20
0x100007a0 <+256>:   str     r0, [sp, #176]  @ 0xb0
0x100007a2 <+258>:   ldr     r1, [sp, #132]  @ 0x84
0x100007a4 <+260>:   bl      0x100011d4 <_ZN10embassy_rp4gpio5Input12wait_for_low28_$u7b$$u7b$closure$u
0x100007a8 <+264>:   cmp     r0, #0
0x100007aa <+266>:   beq.n   0x100007c0 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x100007ac <+268>:   b.n     0x100007ae <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x100007ae <+270>:   add     r0, sp, #32
0x100007b0 <+272>:   movs    r1, #1
0x100007b2 <+274>:   strb    r1, [r0, #0]
0x100007b4 <+276>:   ldr     r2, [sp, #28]
0x100007b6 <+278>:   movs    r1, #4
0x100007b8 <+280>:   strb    r1, [r2, #16]
```

```
0x100007ba <+282>:   ldrb    r0, [r0, #0]
0x100007bc <+284>:   add     sp, #192        @ 0xc0
0x100007be <+286>:   pop     {r4, r6, r7, pc}
0x100007c0 <+288>:   ldr     r0, [sp, #28]
0x100007c2 <+290>:   adds    r0, #20
0x100007c4 <+292>:   bl      0x10001ab8 <_ZN4core3ptr87drop_in_place$LT$embassy_rp..gpio..Input..wait_f
0x100007c8 <+296>:   ldr     r0, [sp, #28]
0x100007ca <+298>:   ldr     r0, [r0, #8]
0x100007cc <+300>:   bl      0x1000020a <_ZN25rust_embassy_pico_project3led3Led2on17h75f7deaba411d497E>
0x100007d0 <+304>:   ldr     r0, [sp, #28]
0x100007d2 <+306>:   ldr     r1, [r0, #12]
0x100007d4 <+308>:   add     r0, sp, #116    @ 0x74
0x100007d6 <+310>:   str     r0, [sp, #8]
0x100007d8 <+312>:   bl      0x100005be <_ZN10embassy_rp4gpio5Input13wait_for_high17h9118f9d341070139E>
0x100007dc <+316>:   ldr     r1, [sp, #8]
0x100007de <+318>:   add     r0, sp, #100    @ 0x64
0x100007e0 <+320>:   bl      0x100013d6 <_ZN59_$LT$F$u20$as$u20$core..future..into_future..IntoFuture$G
0x100007e4 <+324>:   ldr     r1, [sp, #28]
0x100007e6 <+326>:   ldr     r0, [sp, #112]  @ 0x70
0x100007e8 <+328>:   str     r0, [r1, #32]
0x100007ea <+330>:   ldr     r0, [sp, #108]  @ 0x6c
0x100007ec <+332>:   str     r0, [r1, #28]
0x100007ee <+334>:   ldr     r0, [sp, #104]  @ 0x68
0x100007f0 <+336>:   str     r0, [r1, #24]
0x100007f2 <+338>:   ldr     r0, [sp, #100]  @ 0x64
0x100007f4 <+340>:   str     r0, [r1, #20]
0x100007f6 <+342>:   b.n     0x100007f8 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x100007f8 <+344>:   ldr     r0, [sp, #28]
0x100007fa <+346>:   adds    r0, #20
0x100007fc <+348>:   str     r0, [sp, #184]  @ 0xb8
0x100007fe <+350>:   ldr     r1, [sp, #132]  @ 0x84
0x10000800 <+352>:   bl      0x10001274 <_ZN10embassy_rp4gpio5Input13wait_for_high28_$u7b$$u7b$closure$u
0x10000804 <+356>:   cmp     r0, #0
0x10000806 <+358>:   beq.n   0x1000081c <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x10000808 <+360>:   b.n     0x1000080a <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x1000080a <+362>:   add     r0, sp, #32
0x1000080c <+364>:   movs    r1, #1
0x1000080e <+366>:   strb    r1, [r0, #0]
0x10000810 <+368>:   ldr     r2, [sp, #28]
0x10000812 <+370>:   movs    r1, #5
0x10000814 <+372>:   strb    r1, [r2, #16]
0x10000816 <+374>:   ldrb    r0, [r0, #0]
0x10000818 <+376>:   add     sp, #192        @ 0xc0
0x1000081a <+378>:   pop     {r4, r6, r7, pc}
0x1000081c <+380>:   ldr     r0, [sp, #28]
0x1000081e <+382>:   adds    r0, #20
0x10000820 <+384>:   bl      0x10001ae8 <_ZN4core3ptr88drop_in_place$LT$embassy_rp..gpio..Input..wait_f
0x10000824 <+388>:   ldr     r0, [sp, #28]
0x10000826 <+390>:   ldr     r0, [r0, #8]
0x10000828 <+392>:   bl      0x1000021a <_ZN25rust_embassy_pico_project3led3Led3off17h3284da408f5fdb1cE
0x1000082c <+396>:   add     r0, sp, #144    @ 0x90
0x1000082e <+398>:   str     r0, [sp, #4]
0x10000830 <+400>:   movs    r2, #10
```

```
0x10000832 <+402>:   movs    r3, #0
0x10000834 <+404>:   bl      0x100018de <_ZN12embassy_time5timer5Timer12after_millis17h2c899eceea774000
0x10000838 <+408>:   ldr     r0, [sp, #4]
0x1000083a <+410>:   ldr     r3, [sp, #148]  @ 0x94
0x1000083c <+412>:   ldr     r2, [sp, #144]  @ 0x90
0x1000083e <+414>:   ldrb    r0, [r0, #8]
0x10000840 <+416>:   str     r0, [sp, #0]
0x10000842 <+418>:   add     r0, sp, #160    @ 0xa0
0x10000844 <+420>:   bl      0x10000266 <_ZN59_$LT$F$u20$as$u20$core..future..into_future..IntoFuture$G
0x10000848 <+424>:   ldr     r2, [sp, #160]  @ 0xa0
0x1000084a <+426>:   ldr     r4, [sp, #164]  @ 0xa4
0x1000084c <+428>:   ldr     r0, [sp, #168]  @ 0xa8
0x1000084e <+430>:   ldr     r3, [sp, #28]
0x10000850 <+432>:   mov     r1, r3
0x10000852 <+434>:   adds    r1, #24
0x10000854 <+436>:   str     r4, [r3, #28]
0x10000856 <+438>:   str     r2, [r3, #24]
0x10000858 <+440>:   strb    r0, [r1, #8]
0x1000085a <+442>:   b.n     0x1000085c <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x1000085c <+444>:   ldr     r0, [sp, #28]
0x1000085e <+446>:   adds    r0, #24
0x10000860 <+448>:   str     r0, [sp, #180]  @ 0xb4
0x10000862 <+450>:   ldr     r1, [sp, #132]  @ 0x84
0x10000864 <+452>:   bl      0x10008ec0 <_ZN75_$LT$embassy_time..timer..Timer$u20$as$u20$core..future..
0x10000868 <+456>:   cmp     r0, #0
0x1000086a <+458>:   beq.n   0x10000880 <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x1000086c <+460>:   b.n     0x1000086e <_ZN25rust_embassy_pico_project9run_cycle9run_cycle28_$u7b$$u7b
0x1000086e <+462>:   add     r0, sp, #32
0x10000870 <+464>:   movs    r1, #1
0x10000872 <+466>:   strb    r1, [r0, #0]
0x10000874 <+468>:   ldr     r2, [sp, #28]
0x10000876 <+470>:   movs    r1, #6
0x10000878 <+472>:   strb    r1, [r2, #16]
0x1000087a <+474>:   ldrb    r0, [r0, #0]
0x1000087c <+476>:   add     sp, #192        @ 0xc0
0x1000087e <+478>:   pop     {r4, r6, r7, pc}
0x10000880 <+480>:   add     r0, sp, #32
0x10000882 <+482>:   movs    r1, #0
0x10000884 <+484>:   strb    r1, [r0, #0]
0x10000886 <+486>:   ldr     r2, [sp, #28]
0x10000888 <+488>:   movs    r1, #1
0x1000088a <+490>:   strb    r1, [r2, #16]
0x1000088c <+492>:   ldrb    r0, [r0, #0]
0x1000088e <+494>:   add     sp, #192        @ 0xc0
0x10000890 <+496>:   pop     {r4, r6, r7, pc}
0x10000892 <+498>:   nop                     @ (mov r8, r8)
0x10000894 <+500>:   stmia   r0!, {r4}
0x10000896 <+502>:   asrs    r0, r0, #32
```

Each `bne` or `b.n` jump corresponds to one of the `await` suspension points:

- `state = 0` → evaluate `button.is_low()` and possibly await `wait_for_high()`
- `state = 1` → resumed after `wait_for_high().await`
- `state = 2` → await `wait_for_low().await`

- `state = 3` → execute `led.on()`
- `state = 4` → await `wait_for_high().await`
- `state = 5` → execute `led.off()`
- `state = 6` → await `Timer::after_millis().await`
- `state = DONE` → return `Poll::Ready` and exit

By placing a breakpoint at the closure entry (`0x100006a0`), you dive directly into your button-press/LED logic, skipping the trivial constructor at `0x10001904`.

Building and Flashing

Make sure you have: - Rust toolchain with thumbv6m-none-eabi target installed. - Probe-rs or OpenOCD for flashing.

Build:

```
cargo build
```

Flash:

```
cargo flash
```

Requirements

- Rust nightly (for async embedded features).
- Embassy (embassy-rp crate) for async HAL support.
- A Raspberry Pi Pico board.

License

Apache-2.0 License

References

- Embassy: Embedded async executor — https://embassy.dev/
- Raspberry Pi Pico Datasheet — https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf