Rusting up your GreatFET

richö butts
dominic stupid

Who are these jerks

- dominic stupid
- "Extraordinary"
- Senior Computer Jerk
- Great Scott Gadgets

- Ubertooth stuff
- Second best hair in this talk

- richö butts
- slightly less "Extraordinary"
- Senior Computer Jerk
- Stripe

- The umlaut is a historical artifact
- Got up a bit late to write this slide

Who are these jerks

Who are these jerks

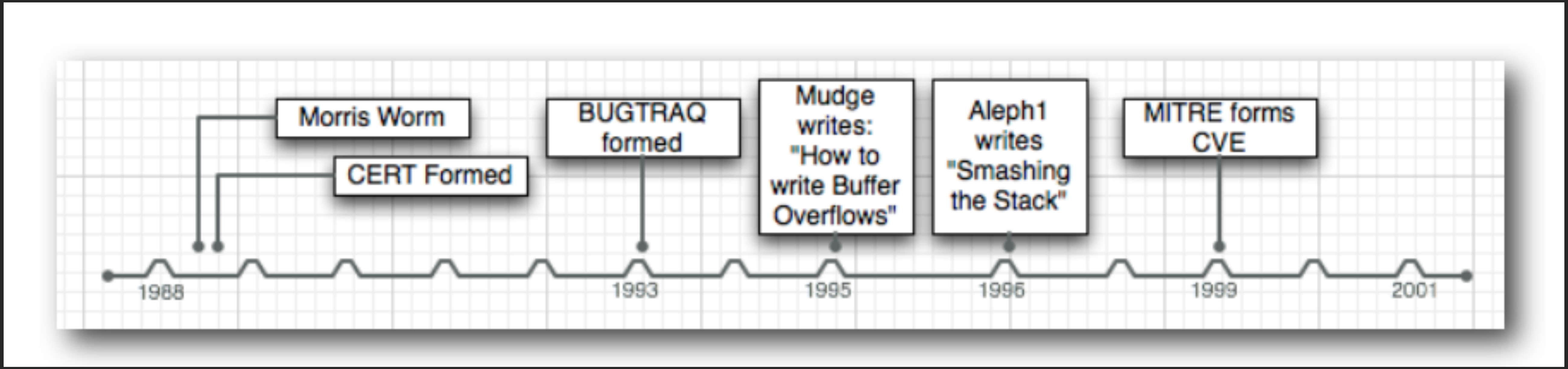
Who are these jerks

Why do you care

Presumably you're in this talk

- Embedded stuff sucks
    - Lol how do I pointers
    - Lol how do I buffers
    - Updates are hard
    - Operability
    - Tooling support
    - Compile times

Why do you care


Morris Worm
CERT Formed
BUGTRAQ formed
Mudge writes: "How to write Buffer Overflows"
Aleph1 writes "Smashing the Stack"
MITRE forms CVE
1988
1993
1995
1996
1999
2001

- Credit: Haroon Meer

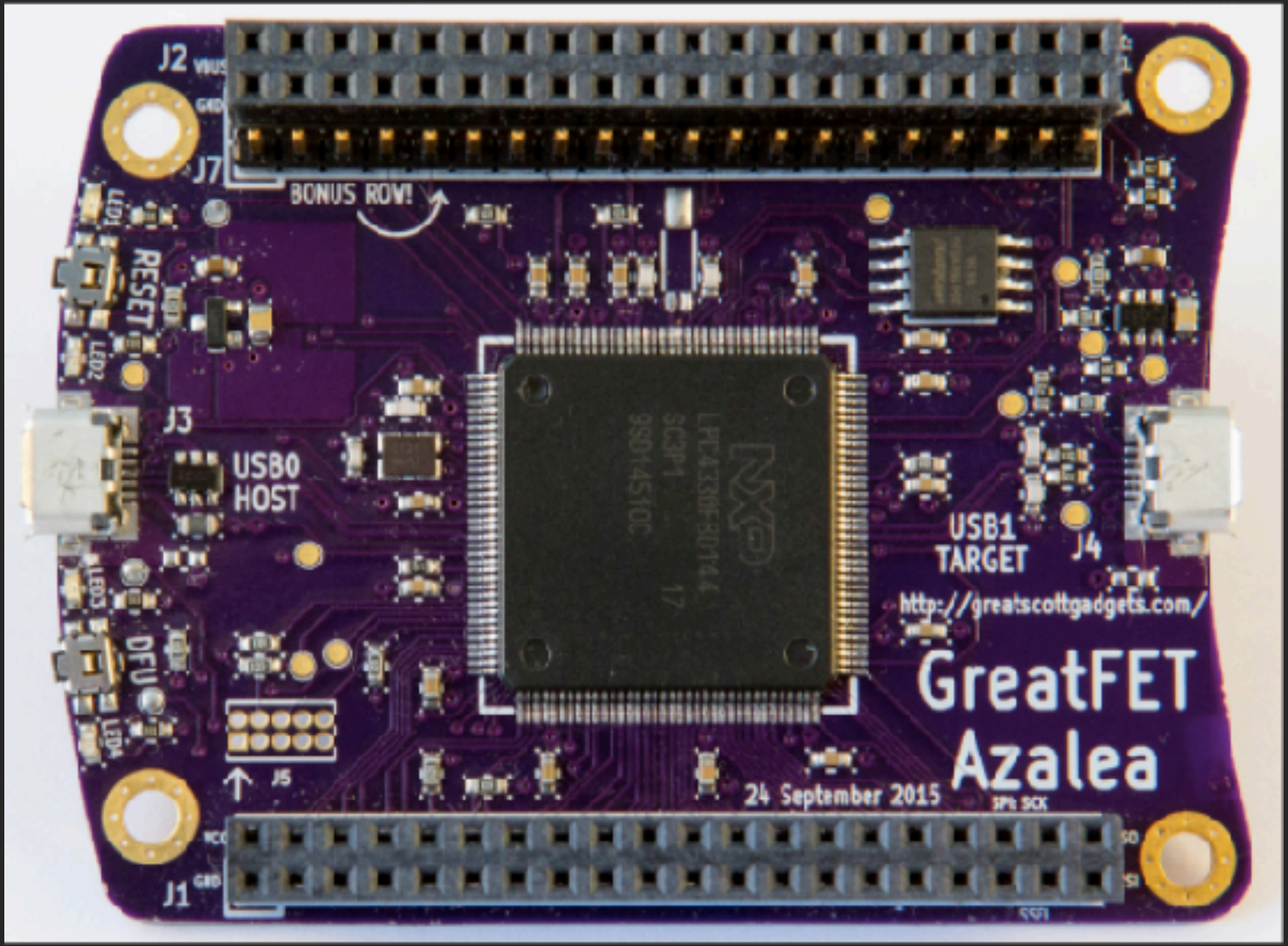
Disclaimer

We swear we sort of know what we're doing

- richö is not a very hardware person
- dominic sort of knows how to program computers

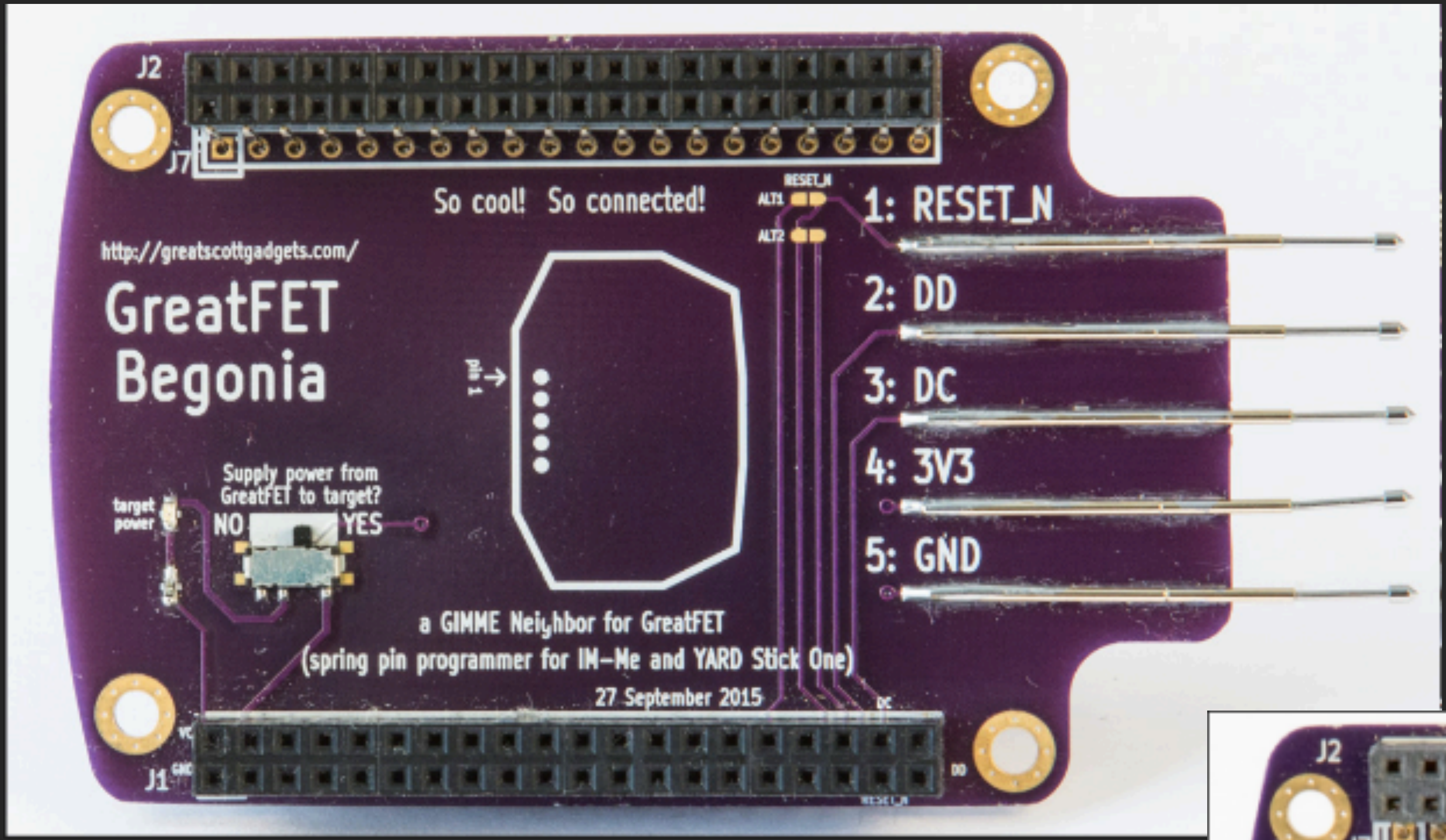
GreatFET

- Hardware hacking platform
- LPC4330 breakout board
- Firmware based on HackRF
- SPI, JTAG, UART, ADC, DAC, GPIO, USB x2
- SGPIO, DMA, Logic Analyser

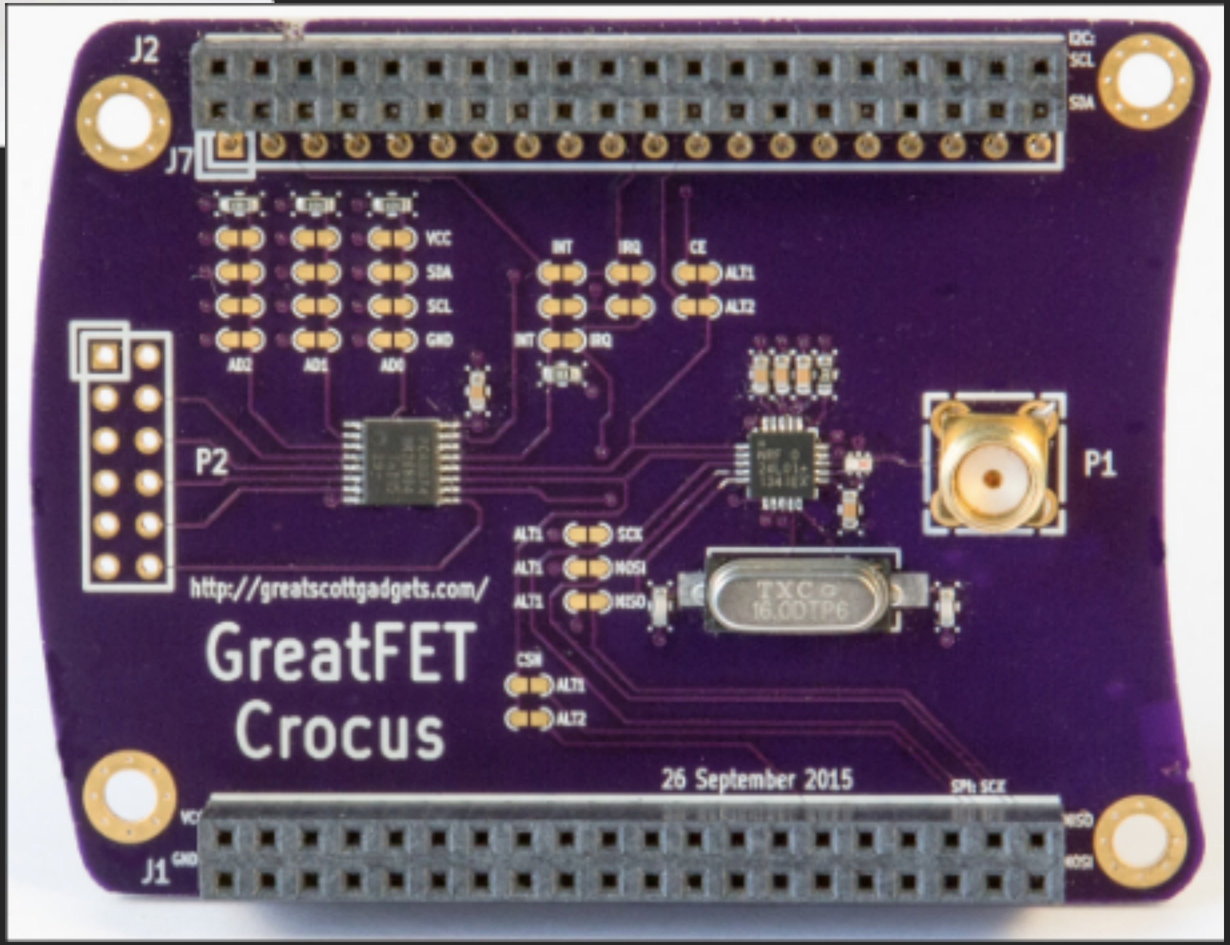


GreatFET

Neighborly af



J2
J7
J1
So cool! So connected!
http://greatscottgadgets.com/
GreatFET
Begonia
RESET_N
ALT1
ALT2
1: RESET_N
2: DD
3: DC
4: 3V3
5: GND
pin 1
Supply power from GreatFET to target?
target power
NO
YES
a GIMME Neighbor for GreatFET
(spring pin programmer for IM-Me and YARD Stick One)
27 September 2015



why not ______?

- Micropython:
  - Concurrency issues
  - Code size
  - Still have to write a lot of C
  - Overheads
  - Debugging hassles
- Incremental C
  - shares many pain points of C
  - Template hell
- µrubby

Rust

Mozilla research project, out of control

- Memory safe
- Static lifetimes
- Coherent package management
- C interoperability
- Big boy generics
- Powerful macro system

- Prevents non-exploitable bugs too!

Rust

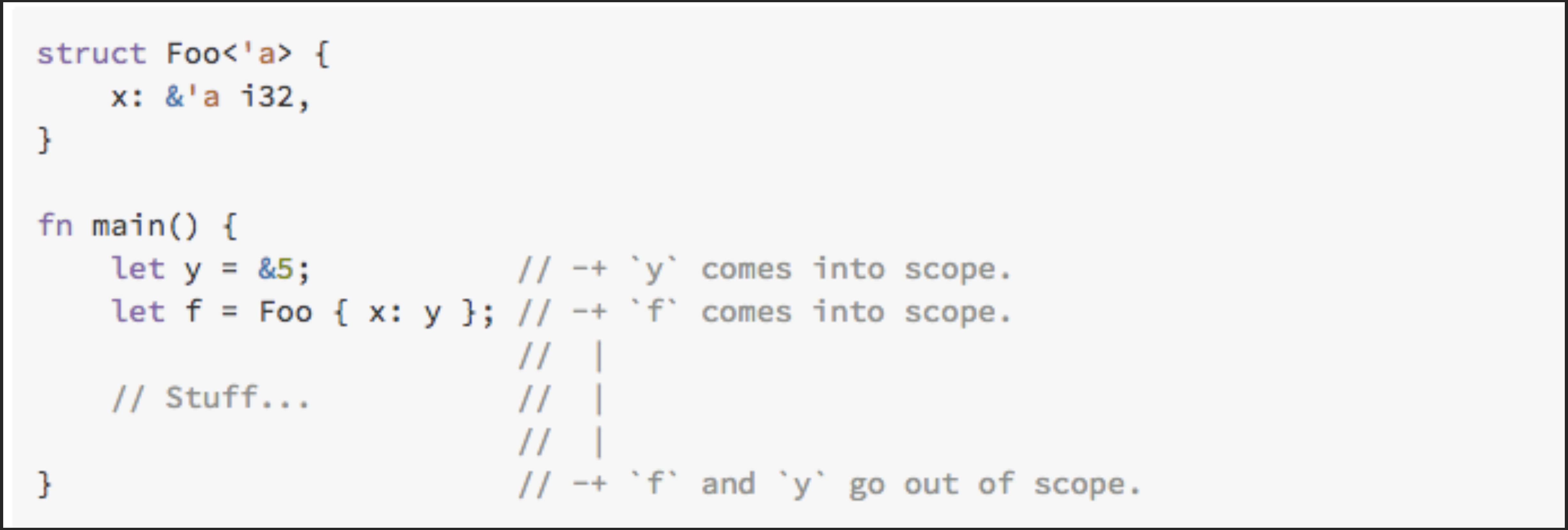
Mozilla research project, out of control

- ✨`lifetimes`✨

Rust

Mozilla research project, out of control

- ✨lifetimes✨

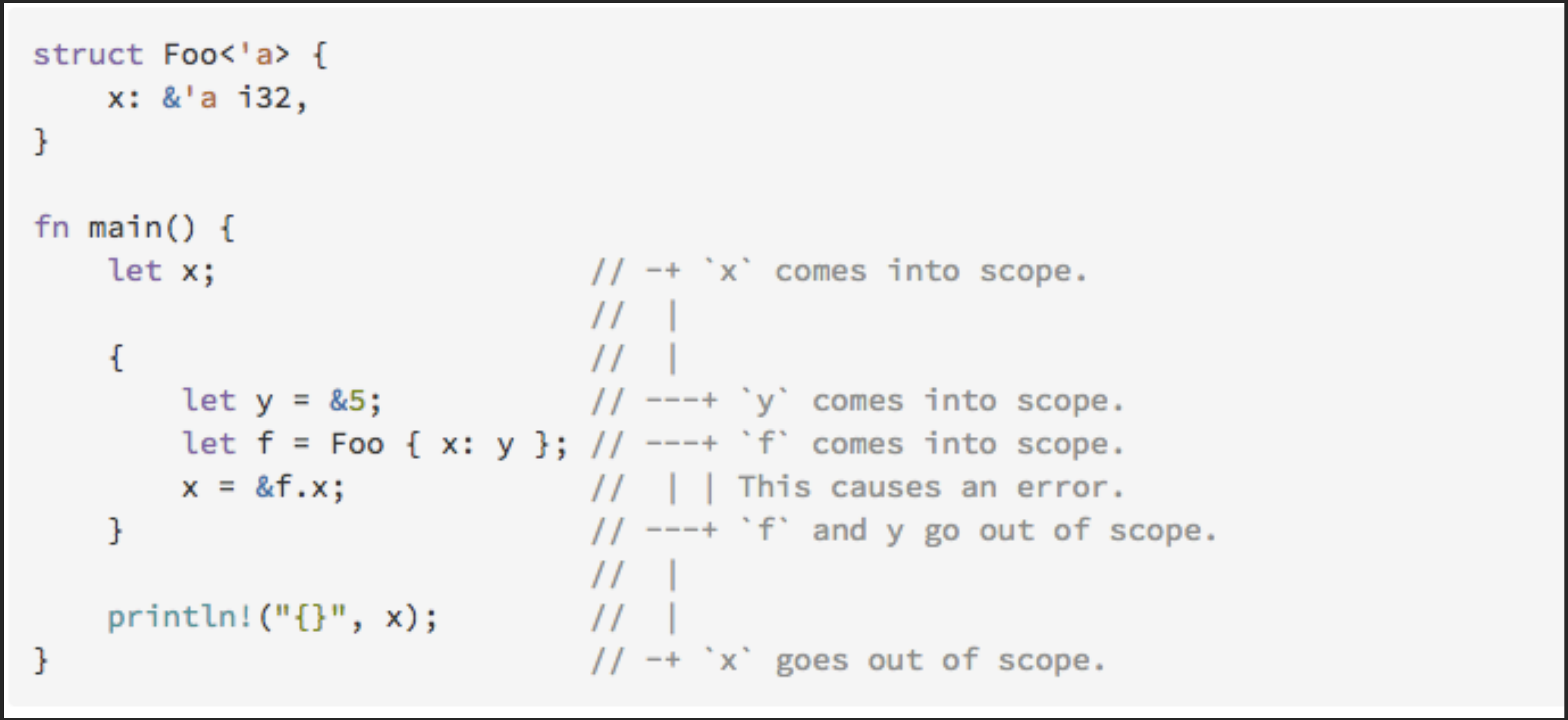


```
struct Foo<'a> {
    x: &'a i32,
}

fn main() {
    let y = &5;           // -+ `y` comes into scope.
    let f = Foo { x: y }; // -+ `f` comes into scope.
                          //  |
    // Stuff...           //  |
                          //  |
}                         // -+ `f` and `y` go out of scope.
```

Rust

Mozilla research project, out of control

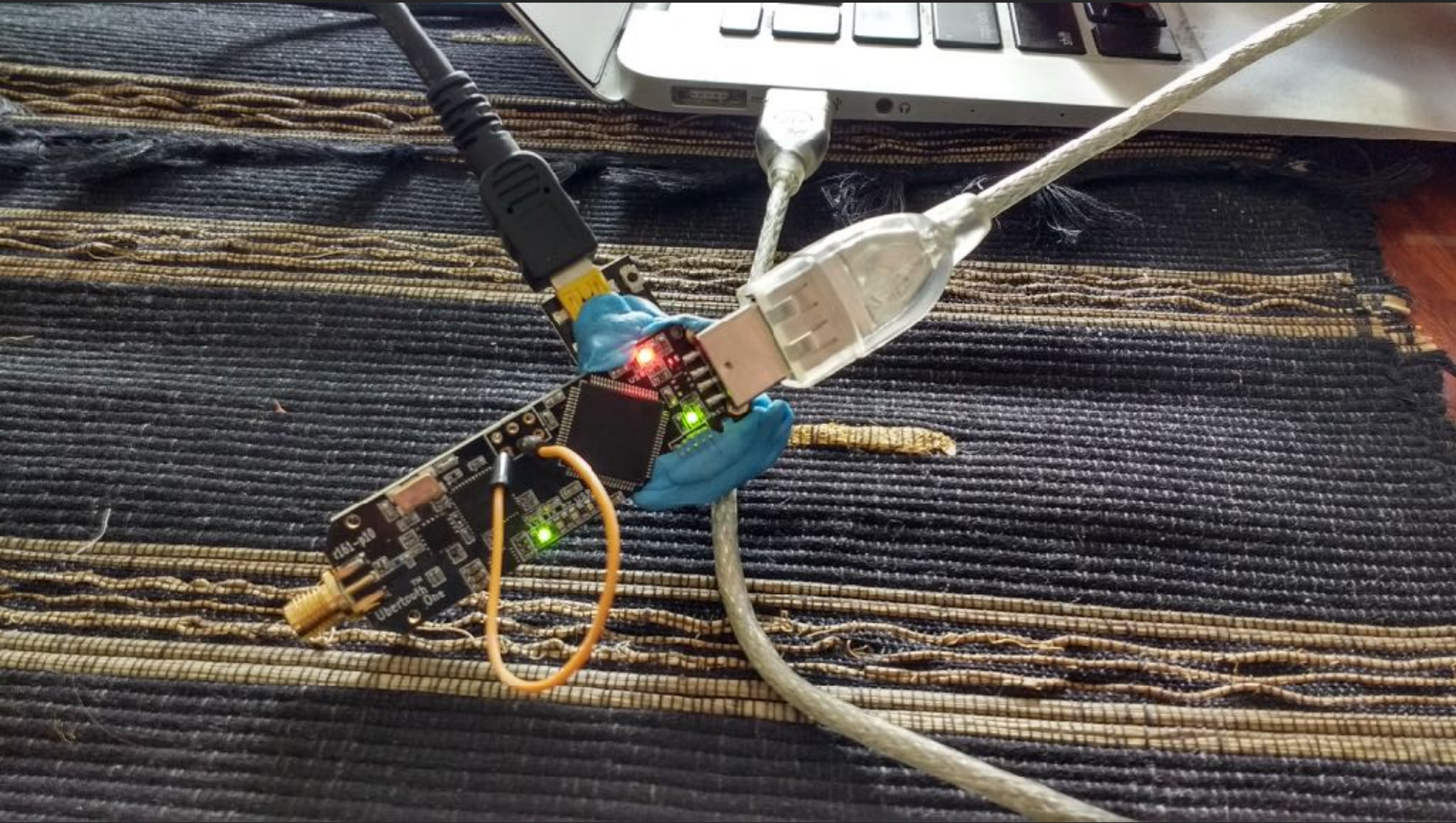
- ✨lifetimes✨



```rust
struct Foo<'a> {
    x: &'a i32,
}

fn main() {
    let x;                    // -+ `x` comes into scope.
                              //  |
    {                         //  |
        let y = &5;           // ---+ `y` comes into scope.
        let f = Foo { x: y }; // ---+ `f` comes into scope.
        x = &f.x;             //  | | This causes an error.
    }                         // ---+ `f` and y go out of scope.
                              //  |
    println!("{}", x);        //  |
}                             // -+ `x` goes out of scope.
```

Last time richo did hardware

his ubertooth still has blutack on it



Making it go
haha! it's a golang joke

- Two main goals:
  - Be able to write a pure rust firmware for GreatFET
  - Embed rust code into an existing firmware codebase

Prior art

jerks who beat us to the punch

- zinc
    - hardware abstraction layer for embedded platforms
- tock
    - experimental RTOS
- http://www.acrawford.com/2017/03/09/rust-on-the-cortex-m3.html
    - bare metal rust on cortex m3

zero to hero

- Pick a project that seems plausible
- Randomly twiddle bits in linker scripts until you're satisfied with the results

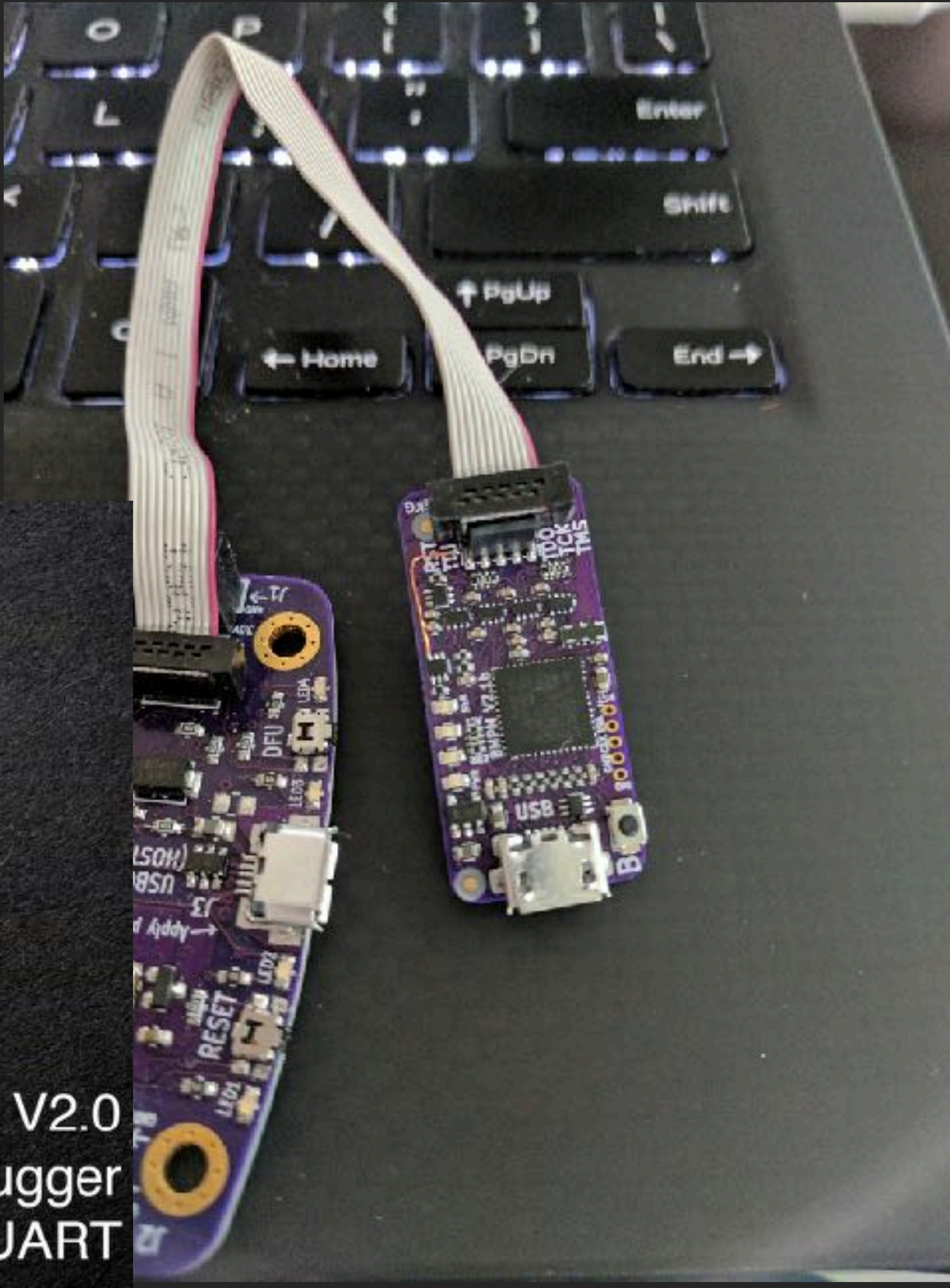
- ?????
- Speak at TROOPERS!

Get you a greatfet

protip: Forget shit you need, find brian

Look into your GreatFET

- Black Magic Probe
- Natively talks gdb
- Exactly zero openocd is the right amount



Black Magic Probe V2.0
Open Source JTAG & SWD GNU Debugger and Programmer with built in GDB server & UART

Goal 1
Blink some LEDs

- Configure GPIO (poke memory)
- Configure the pin (poke memory)
- lpc4330 has 8 gpio ports, each with 32 pins
    - greatfet package has 144 pins
    - not all can be used for GPIO
        - Selfishly, it needs power and stuff
- Set Direction (poke memory)
- Write data to pin (poke memory)

Goal 0

Execute code on a greatfet

- Futz around with the existing build pipeline for GreatFET to translate an elf object into something that can be written to flash

- ... or!
- Use black magic probe + gdb's support for writing an elf into memory

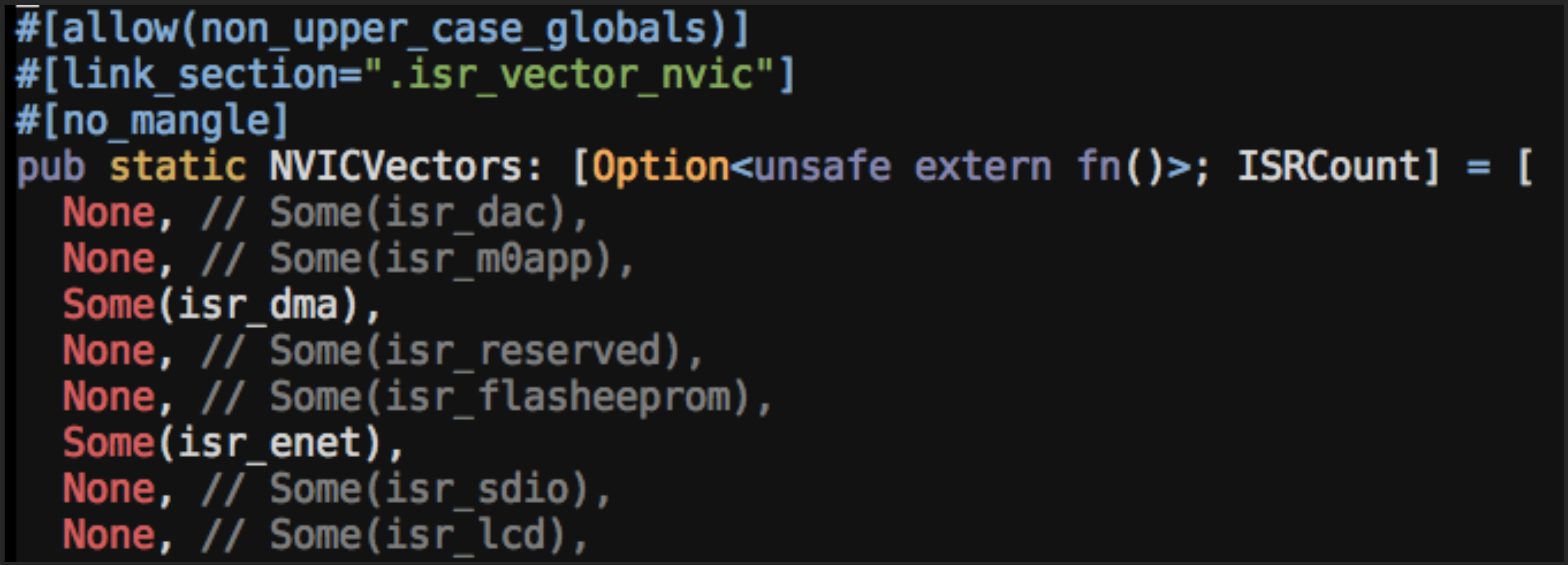
Goal 0.5

Execute code on a greatfet

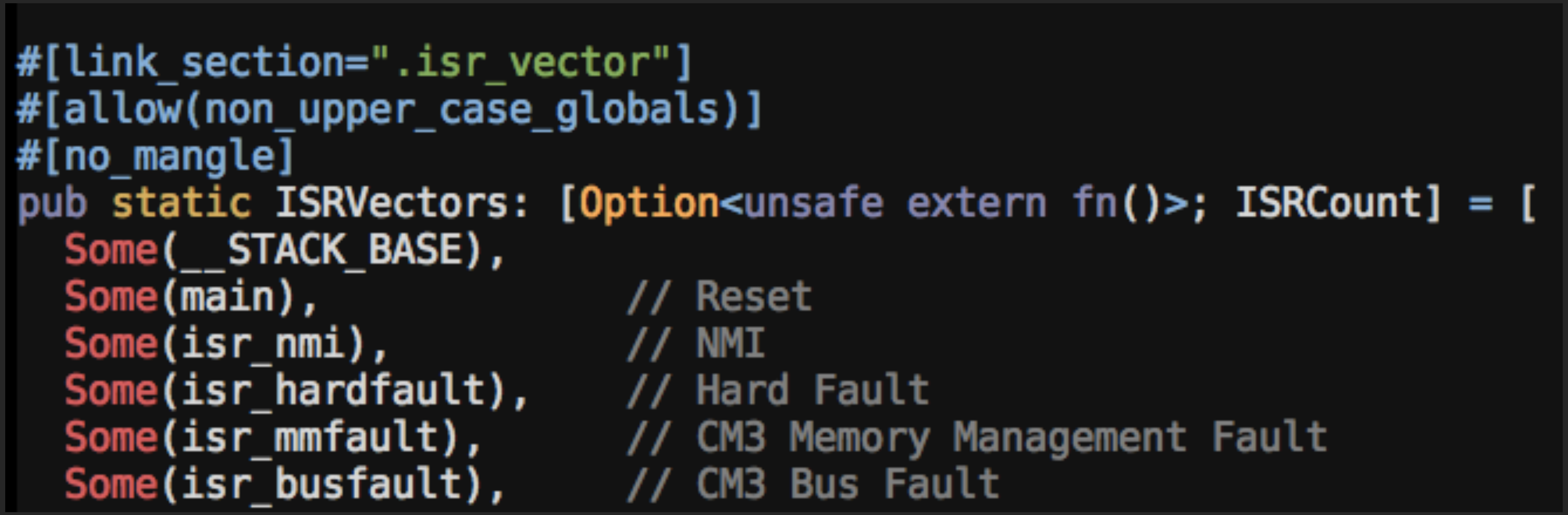
- On a "normal computer" having a stack, heap, executable mapped into memory, etc is free
- On embedded, you need to setup your own stack, install interrupt handlers, etc before you get too carried away
- `zinc::hal::mem_init::init_stack();`
- `zinc::hal::mem_init::init_data();`

Goal 0.7

This metaphor has gotten away from me a little



```
#[allow(non_upper_case_globals)]
#[link_section=".isr_vector_nvic"]
#[no_mangle]
pub static NVICVectors: [Option<unsafe extern fn()>; ISRCount] = [
  None, // Some(isr_dac),
  None, // Some(isr_m0app),
  Some(isr_dma),
  None, // Some(isr_reserved),
  None, // Some(isr_flasheeprom),
  Some(isr_enet),
  None, // Some(isr_sdio),
  None, // Some(isr_lcd),
```

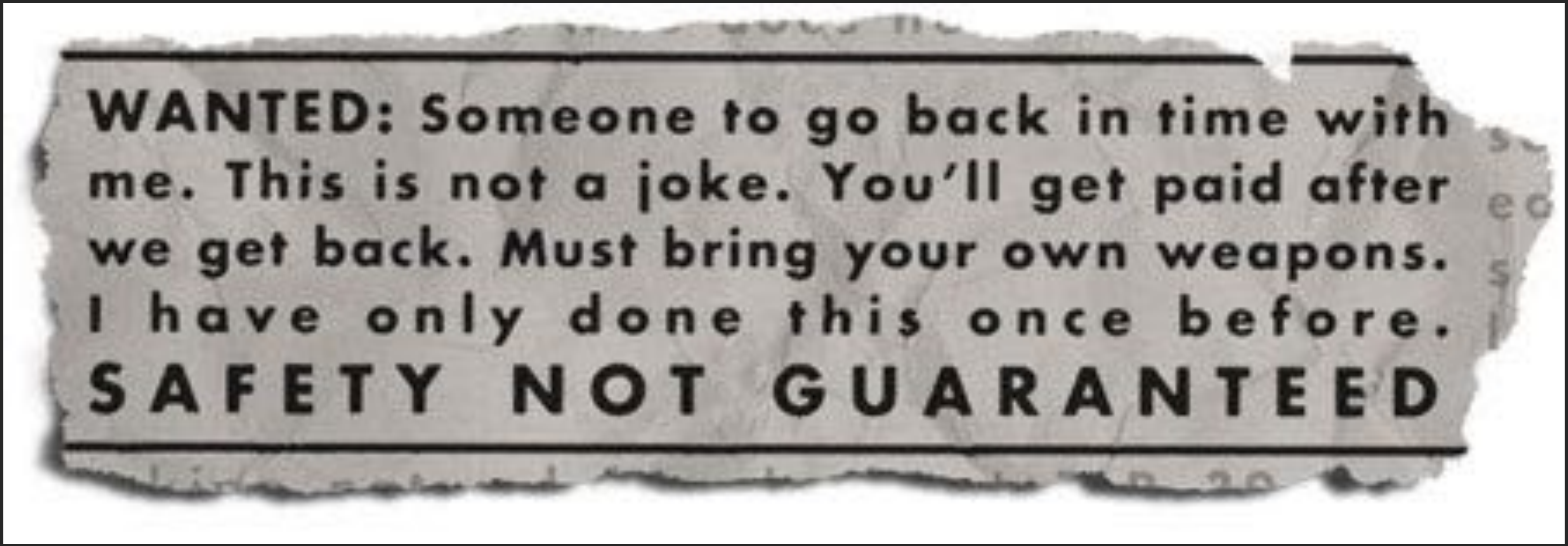


```
#[link_section=".isr_vector"]
#[allow(non_upper_case_globals)]
#[no_mangle]
pub static ISRVectors: [Option<unsafe extern fn()>; ISRCount] = [
  Some(__STACK_BASE),
  Some(main),             // Reset
  Some(isr_nmi),          // NMI
  Some(isr_hardfault),    // Hard Fault
  Some(isr_mmfault),      // CM3 Memory Management Fault
  Some(isr_busfault),     // CM3 Bus Fault
```

Goal 1
Blink some LEDs

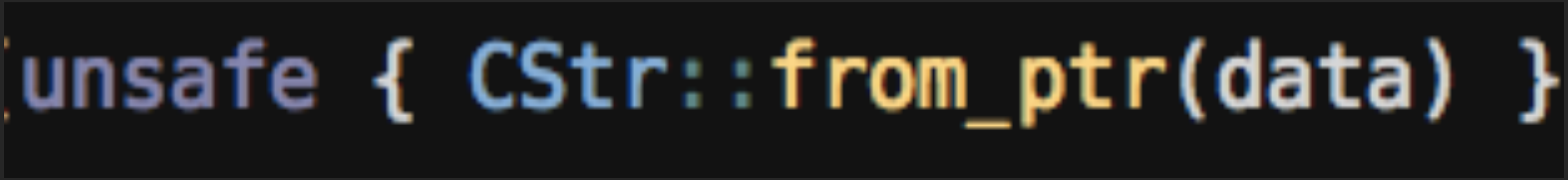
- Configu → (poke memory)
- Configur → (poke memory)
- lpc4330 has 8 gpio ports, each with 32 pins
  - greatfet package has 144 pins
  - not all can be used for GPIO
    - Selfishly, it needs power and stuff
- Set D → (poke memory)
- Write da → (poke memory)

unsafety



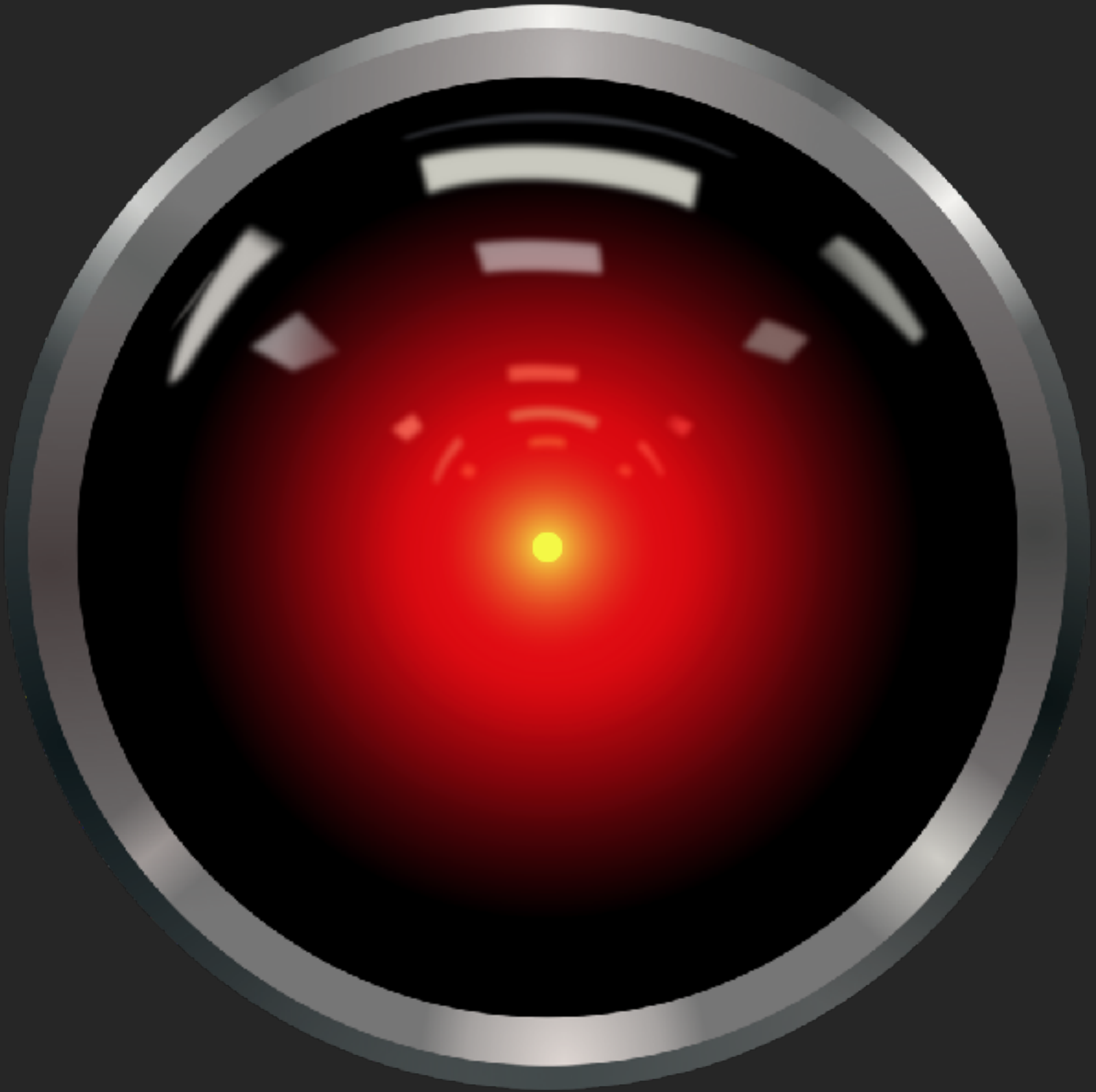
WANTED: Someone to go back in time with me. This is not a joke. You'll get paid after we get back. Must bring your own weapons. I have only done this once before.

SAFETY NOT GUARANTEED



```
unsafe { CStr::from_ptr(data) }
```

The HAL



Goal 1 revisited
Blink some LEDs

- Write Rust abstraction over GreatFETs GPIO
- Expose logical LEDs to userland code!

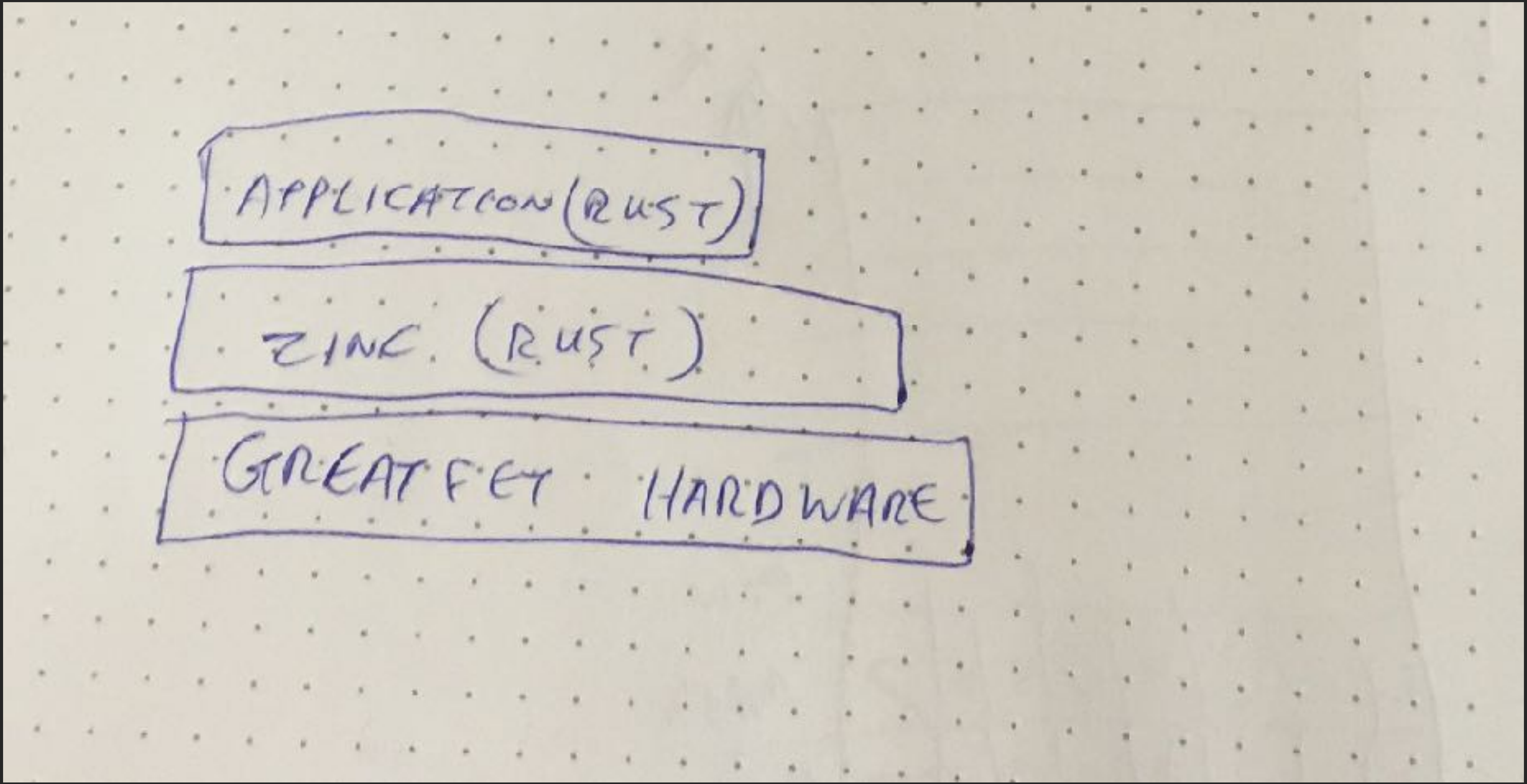
- Great success
- Once we had a "read to"/"write from" register abstraction, we can build anything

demo time

Don't get excited, it's blinking LEDs

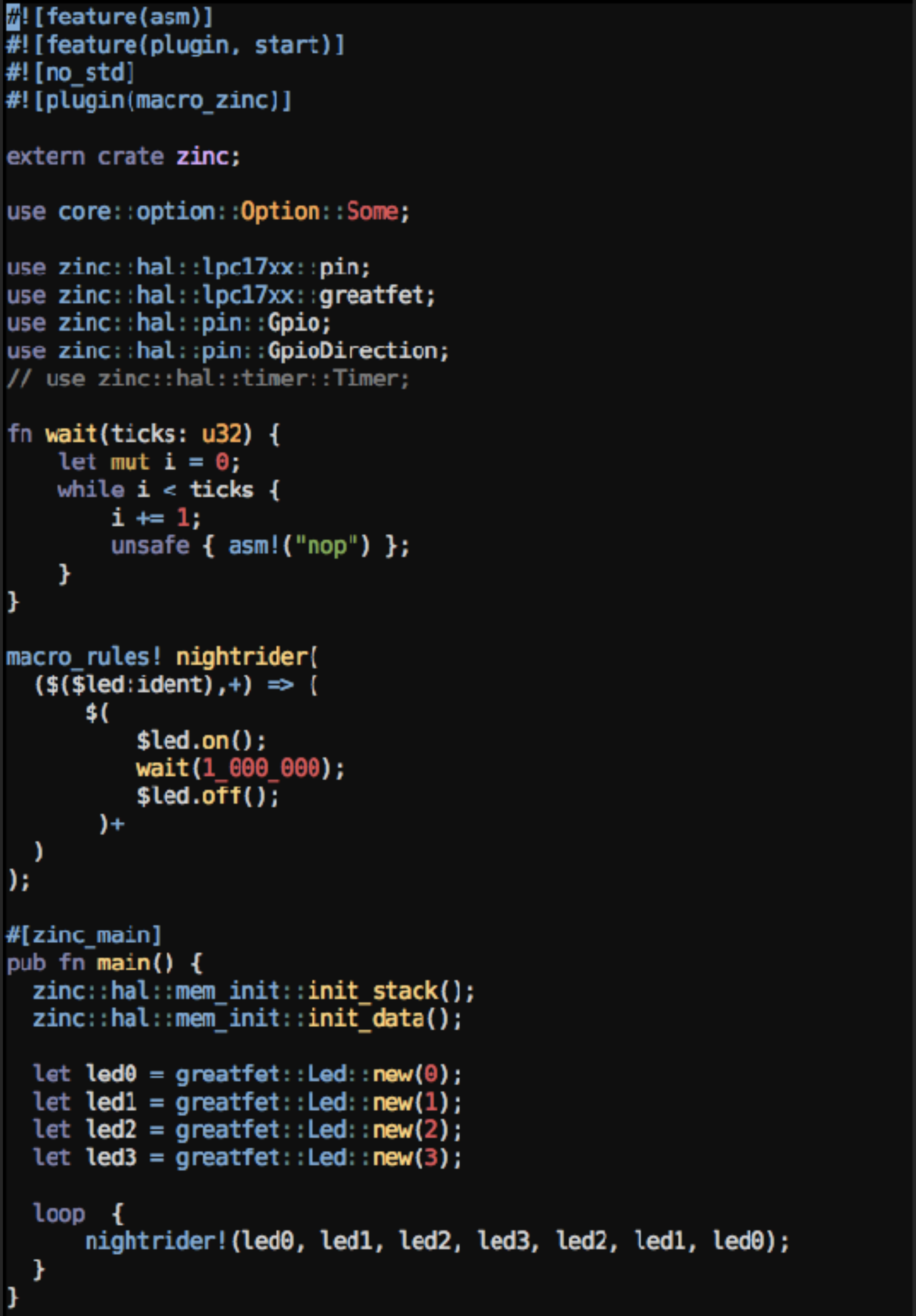
demo time

Rust on GreatFET



demo time

Rust on GreatFET



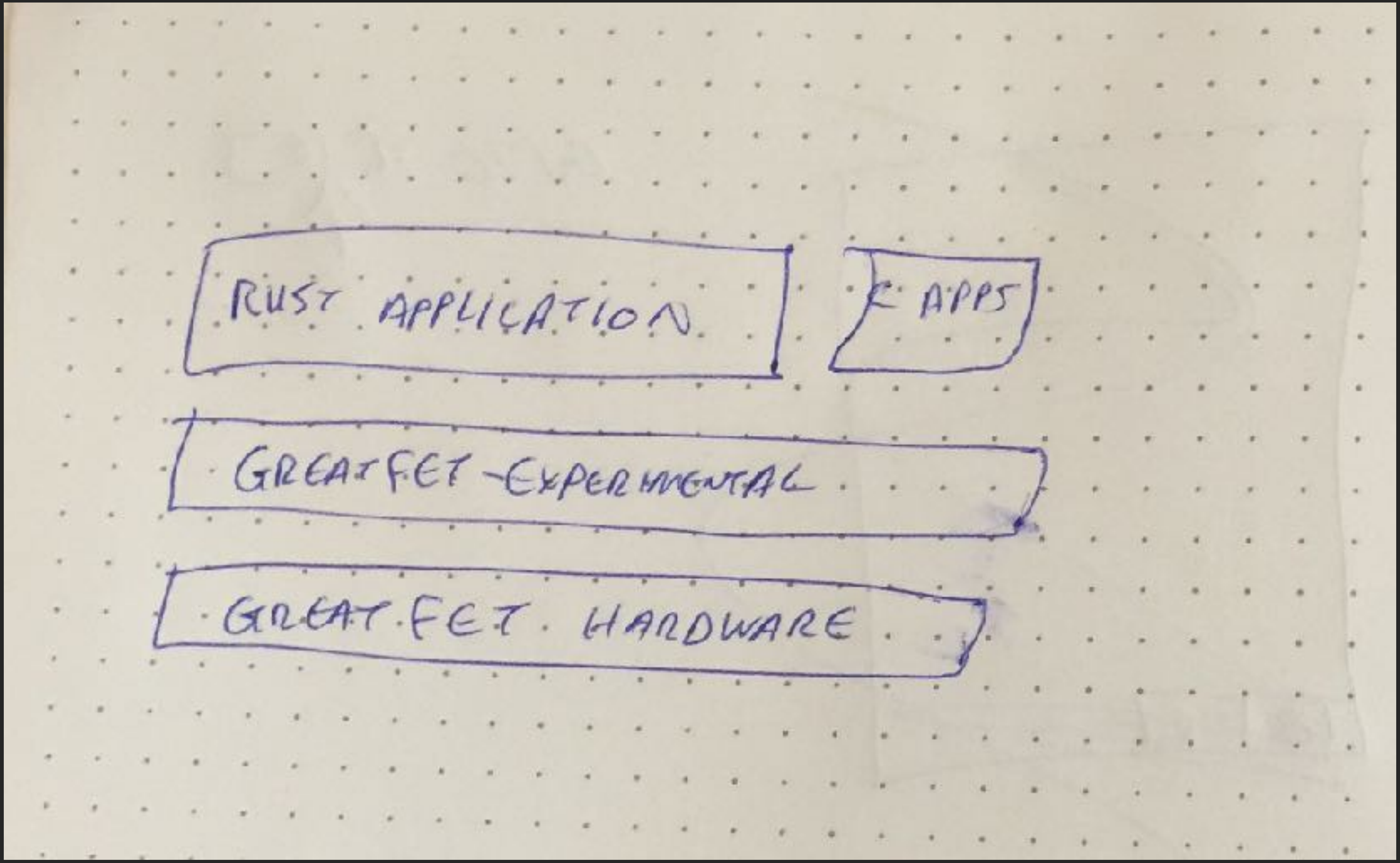
```
#![feature(asm)]
#![feature(plugin, start)]
#![no_std]
#![plugin(macro_zinc)]

extern crate zinc;

use core::option::Option::Some;

use zinc::hal::lpc17xx::pin;
use zinc::hal::lpc17xx::greatfet;
use zinc::hal::pin::Gpio;
use zinc::hal::pin::GpioDirection;
// use zinc::hal::timer::Timer;

fn wait(ticks: u32) {
    let mut i = 0;
    while i < ticks {
        i += 1;
        unsafe { asm!("nop") };
    }
}

macro_rules! nightrider(
  ($($led:ident),+) => (
      $(
          $led.on();
          wait(1_000_000);
          $led.off();
       )+
  )
);

#[zinc_main]
pub fn main() {
  zinc::hal::mem_init::init_stack();
  zinc::hal::mem_init::init_data();

  let led0 = greatfet::Led::new(0);
  let led1 = greatfet::Led::new(1);
  let led2 = greatfet::Led::new(2);
  let led3 = greatfet::Led::new(3);

  loop  {
      nightrider!(led0, led1, led2, led3, led2, led1, led0);
  }
}
```

demo time

Our demo probably failed, have an otter

demo time

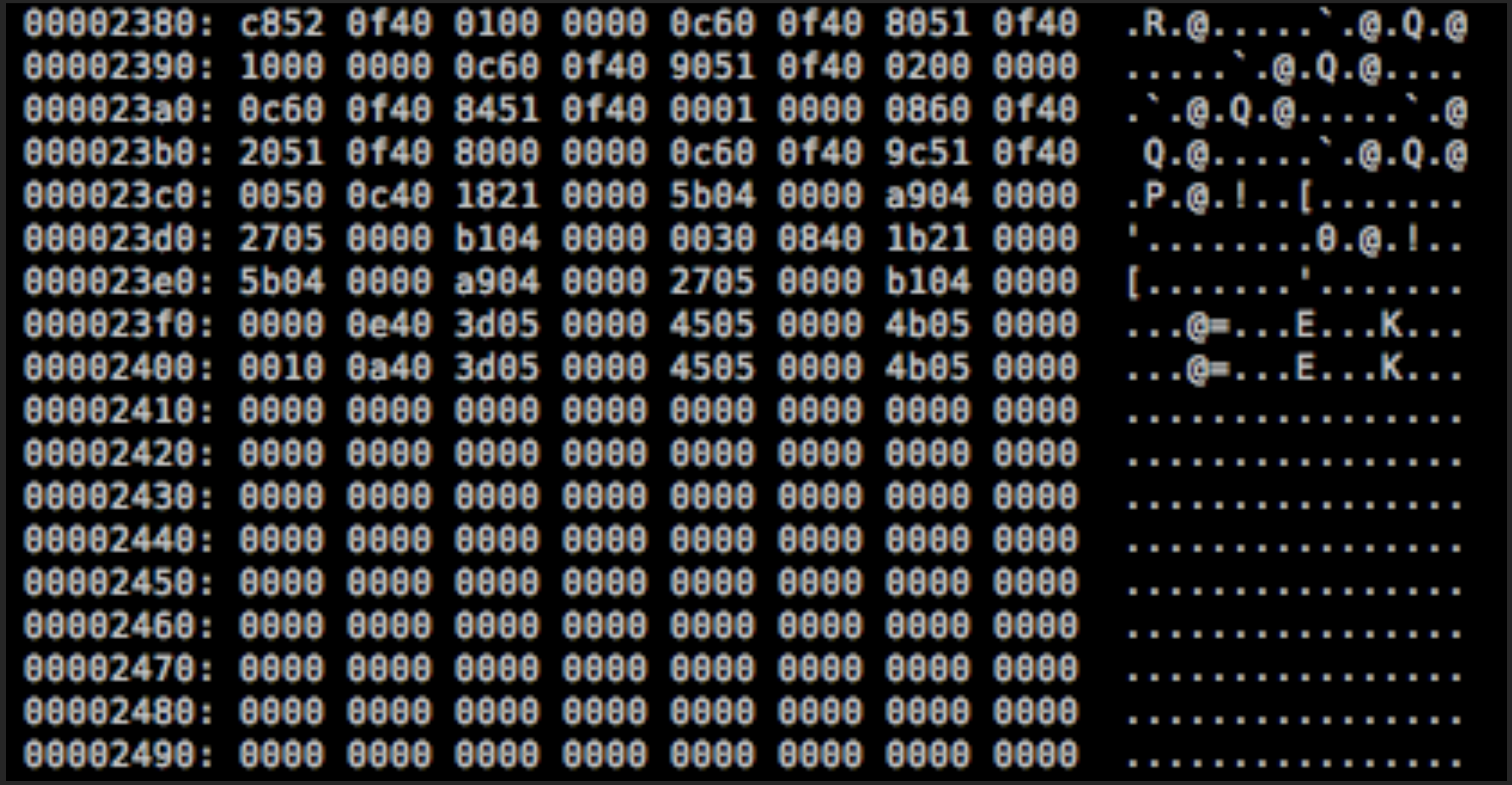
Rust on GreatFET on GreatFET



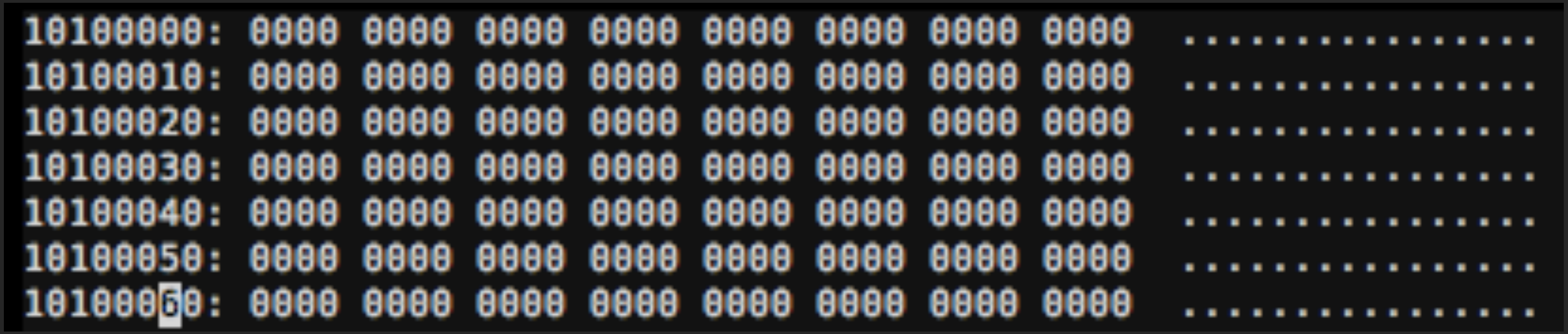
objcopy is bad software

- 337kb elf -> 257mb bin (WTF objcopy?)
- Probably some hilarious underflow.
    - .... Should have written it in rust

objcopy is bad software



```
00002380: c852 0f40 0100 0000 0c60 0f40 8051 0f40  .R.@.....`.@.Q.@
00002390: 1000 0000 0c60 0f40 9051 0f40 0200 0000  .....`.@.Q.@....
000023a0: 0c60 0f40 8451 0f40 0001 0000 0860 0f40  .`.@.Q.@.....`.@
000023b0: 2051 0f40 8000 0000 0c60 0f40 9c51 0f40   Q.@.....`.@.Q.@
000023c0: 0050 0c40 1821 0000 5b04 0000 a904 0000  .P.@.!..[.......
000023d0: 2705 0000 b104 0000 0030 0840 1b21 0000  '........0.@.!..
000023e0: 5b04 0000 a904 0000 2705 0000 b104 0000  [.......'.......
000023f0: 0000 0e40 3d05 0000 4505 0000 4b05 0000  ...@=...E...K...
00002400: 0010 0a40 3d05 0000 4505 0000 4b05 0000  ...@=...E...K...
00002410: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00002420: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00002430: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00002440: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00002450: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00002460: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00002470: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00002480: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00002490: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```



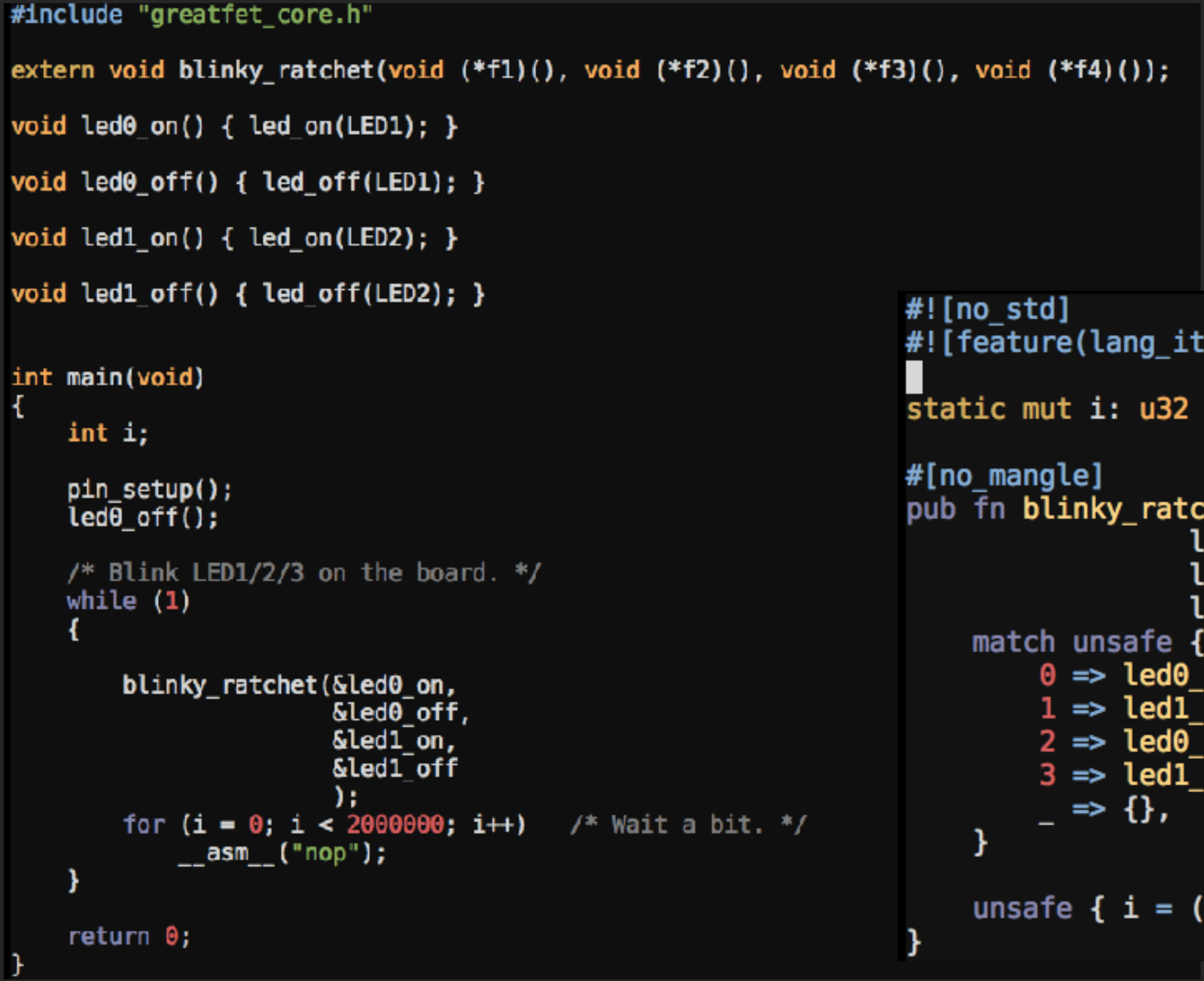
```
10100000: 0000 0000 0000 0000 0000 0000 0000 0000  ................
10100010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
10100020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
10100030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
10100040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
10100050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
10100060: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

objcopy is bad software

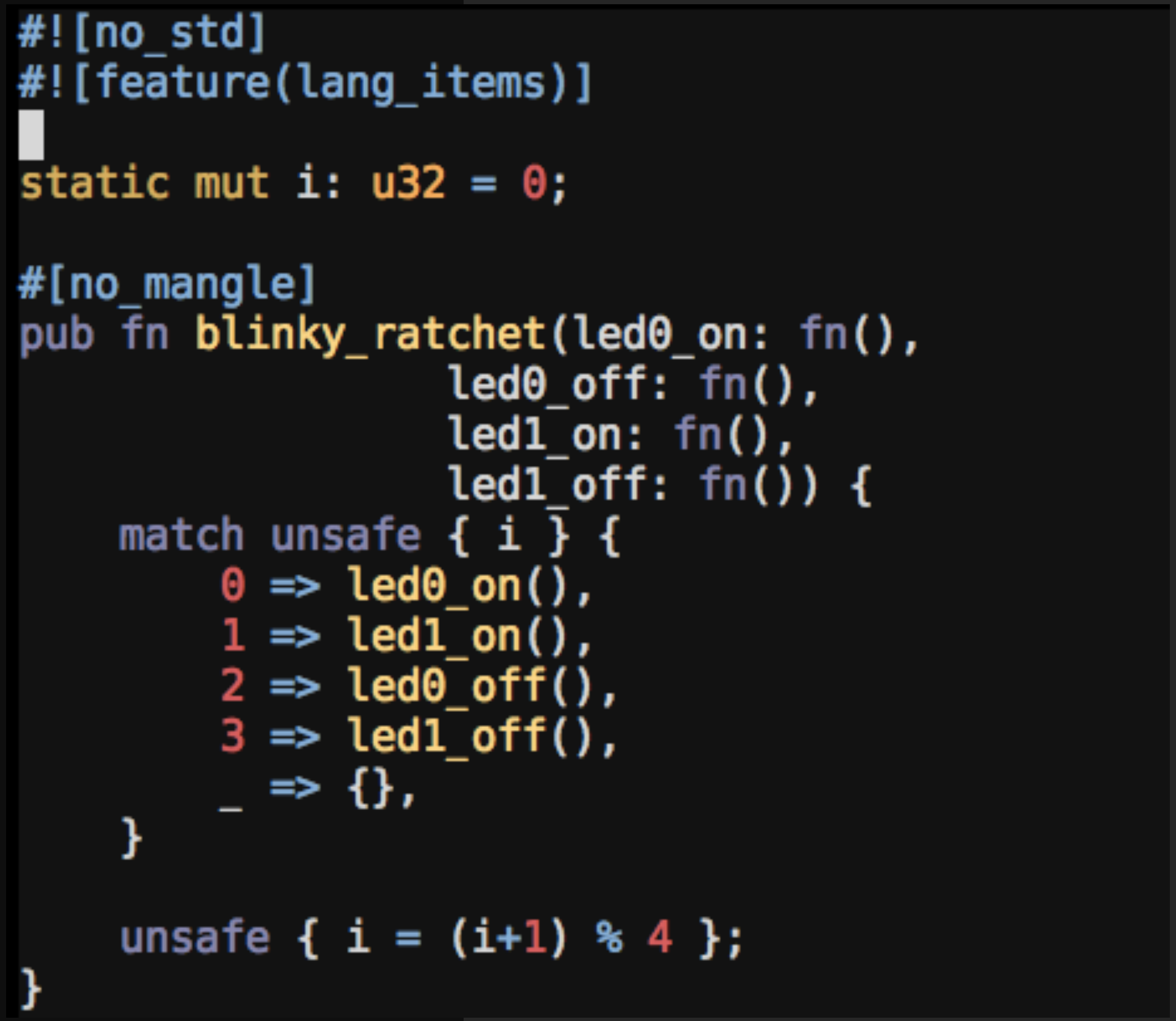
- Whatever all those zeros are probably not important
- ¯_(ツ)_/¯

demo time

Rust on GreatFET on GreatFET



```c
#include "greatfet_core.h"

extern void blinky_ratchet(void (*f1)(), void (*f2)(), void (*f3)(), void (*f4)());

void led0_on() { led_on(LED1); }

void led0_off() { led_off(LED1); }

void led1_on() { led_on(LED2); }

void led1_off() { led_off(LED2); }


int main(void)
{
    int i;

    pin_setup();
    led0_off();

    /* Blink LED1/2/3 on the board. */
    while (1)
    {

        blinky_ratchet(&led0_on,
                       &led0_off,
                       &led1_on,
                       &led1_off
                       );
        for (i = 0; i < 2000000; i++)    /* Wait a bit. */
            __asm__("nop");
    }

    return 0;
}
```



```rust
#![no_std]
#![feature(lang_items)]

static mut i: u32 = 0;

#[no_mangle]
pub fn blinky_ratchet(led0_on: fn(),
                      led0_off: fn(),
                      led1_on: fn(),
                      led1_off: fn()) {
    match unsafe { i } {
        0 => led0_on(),
        1 => led1_on(),
        2 => led0_off(),
        3 => led1_off(),
        _ => {},
    }

    unsafe { i = (i+1) % 4 };
}
```

demo time

.... yup. Otters.

demo time

But not yet

- Go to mike and dominic's talk on thursday 4pm

Where does that leave us?

- 100% rust code
  - two interrupt handlers written in inline asm
- Still uses linker scripts to describe memory mapped registers to native Rust code
- Uses some unfortunate tricks to abstract over unsafe memory access
- Cargo works natively!
  - Want to terminate TLS on your greatfet for some reason?

Why do you care

Subtitle Text

- Embedded stuff sucks
    - Lol how do I pointers
        - Lifetimes! Borrow Checker!
    - Updates are hard
        - Cargo!
    - Operability
        - hella static analysis
    - Compile times
        - Incremental compilation, coherent module system
    - Generalisable code

Challenges for adoption

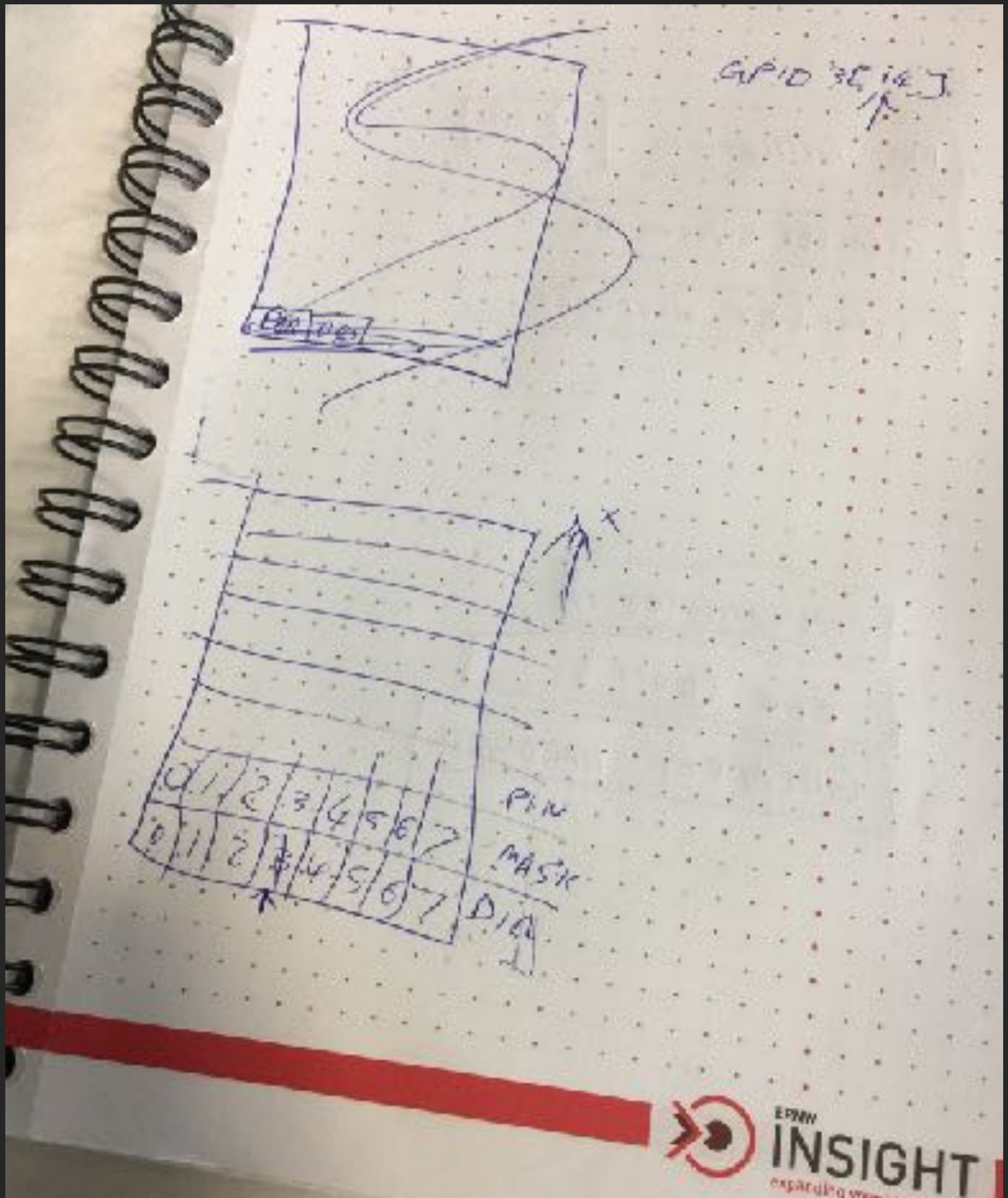
- Unwillingness to rewrite your whole codebase in Rust
    - Incremental rewrites now possible
- Rust learning curve
- Support doesn't magically port existing software

things don't always go well

- zinc has some serious tooling problems
- rust error messages are great
    - ... unless the bug is in a compiler plugin
        - Zinc is made of compiler plugins
- richö isn't very good at comprehension
    - so we might have wasted 20% of the development time on writing randomly across memory mapped registers

things don't always go well

- But seriously, do you read this and immediately know how to interact with GPIO on greatfet?



Questions?

Resources

Feel free to take pictures

- github.com/richo/zinc
  - The zinc fork with support for greatfet
- https://github.com/dominicgs/GreatFET-experimental/tree/rust/firmware
  - GreatFET firmware with support for embedded rust
- speakerdeck.com/richo/rust-greatfet
  - The slides for this talk
- We're on twitter
  - `@dominicgs @rich0H`
- We'll release a docker image