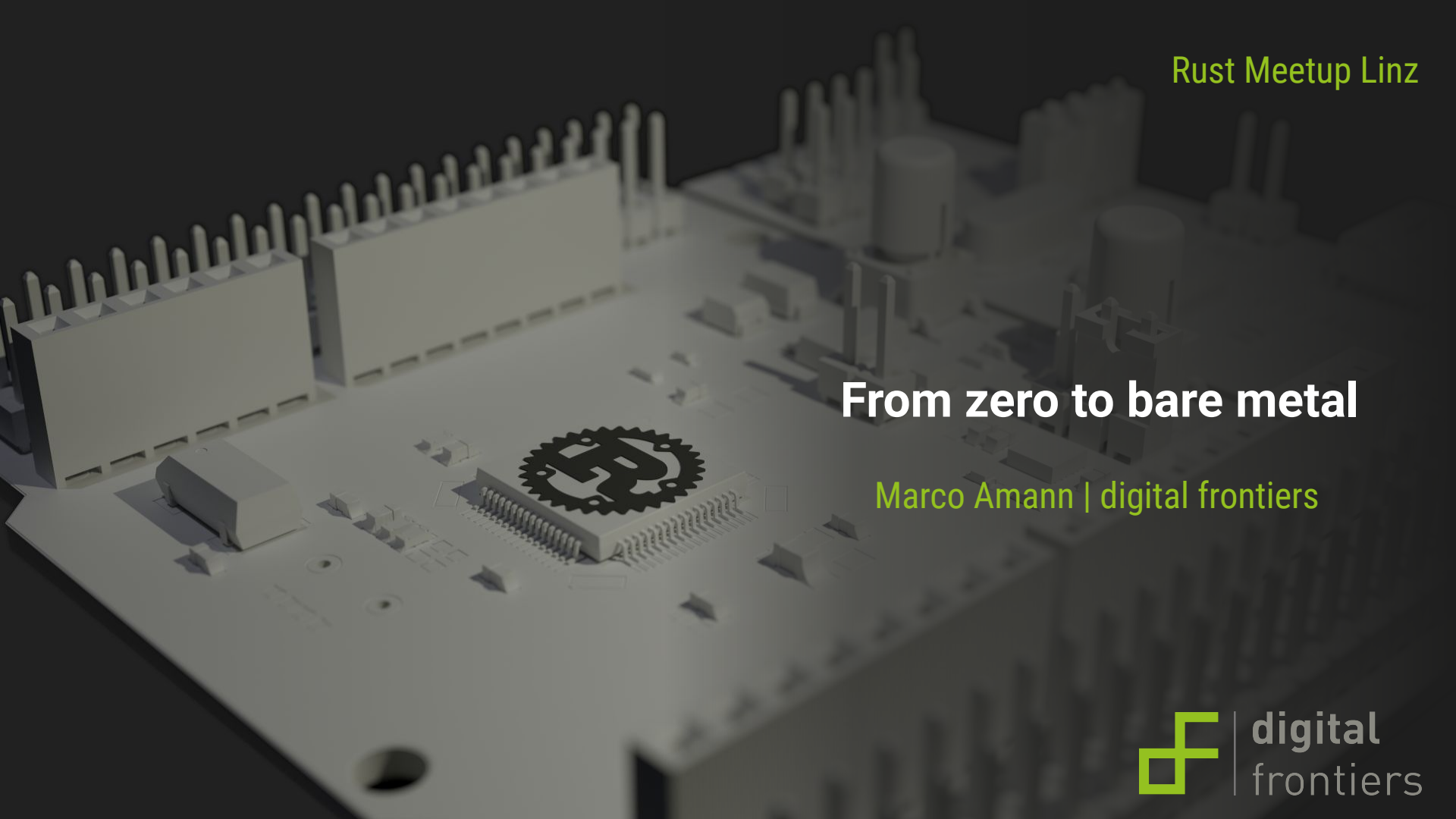


Rust Meetup Linz

From zero to bare metal

Marco Amann | digital frontiers

digital frontiers

About me

digitalfrontiers.de

marco@amann.dev

@amann_dev

rust-buch.de

#WeAreHiring

digital frontiers

Unsafe

Unsafe

Weniger Garantien

- raw pointer
- FFI
- safe wrapper

Photo by Vladyslav Cherkasenko on Unsplash

digital frontiers

```rust
let num = 42u16;
let ptr = &num as *const u16;
…
```

Pointer

Beispiele:

01_pointer.rs

02_bad_ideas_with_unsafe.rs

digital frontiers

FFI - Foreign Function Interface

(Vorschau)

Wie kann der Compiler Garantien für C-Code ausstellen?

Syscall

```rust
fn main() {

    let mut t =  libc::timespec {
        tv_sec: 0,
        tv_nsec: 0
     } ;

    let _ = unsafe { libc::clock_gettime(99, &mut t) };
    println!("wrong: {} {}", t.tv_sec, t.tv_nsec);

}
```

Beispiele:

03_syscall_sloppy.rs

04_syscall_better.rs

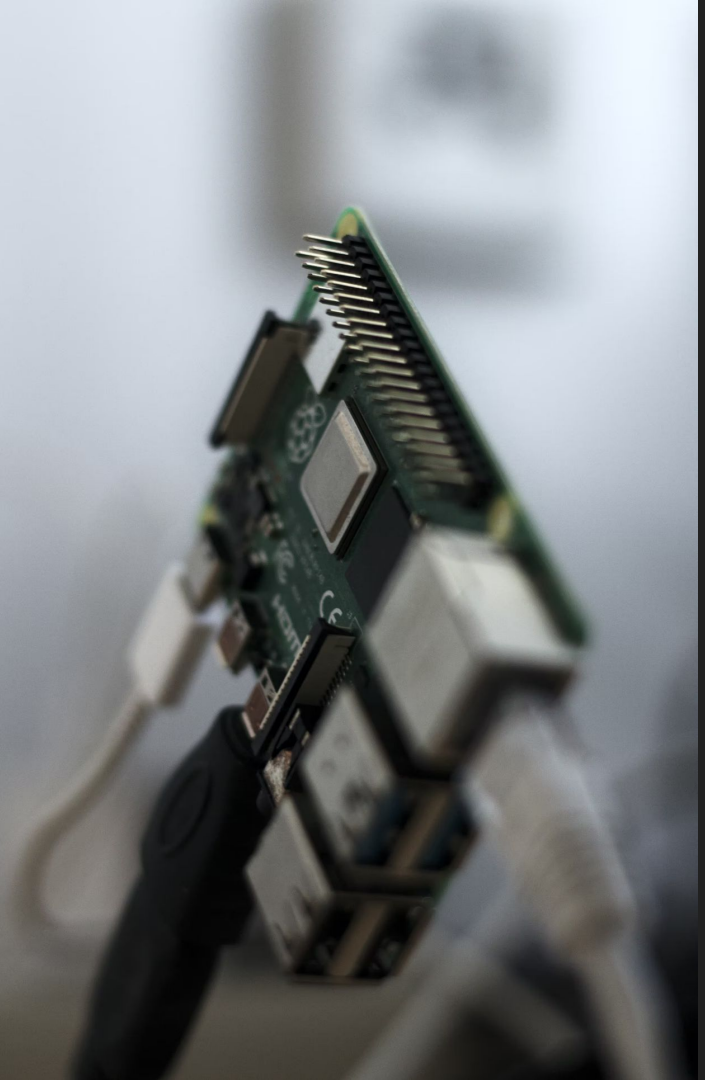
Unsafe

- Sichere Wrapper um unsafe

- unsafe ist nur so (un)sicher wie man damit umgeht

Hardware

Endlich....



LED Blink

```rust
const GPIO_LED: u8 = 23;     // phys 16

fn main() -> Result<()> {
    let mut pin = Gpio::new()?
        .get(GPIO_LED)?
        .into_output();
    loop {
        pin.toggle();
        thread::sleep(Duration::from_millis(200));
    }
}
```

Photo by Quentin Schulz on Unsplash

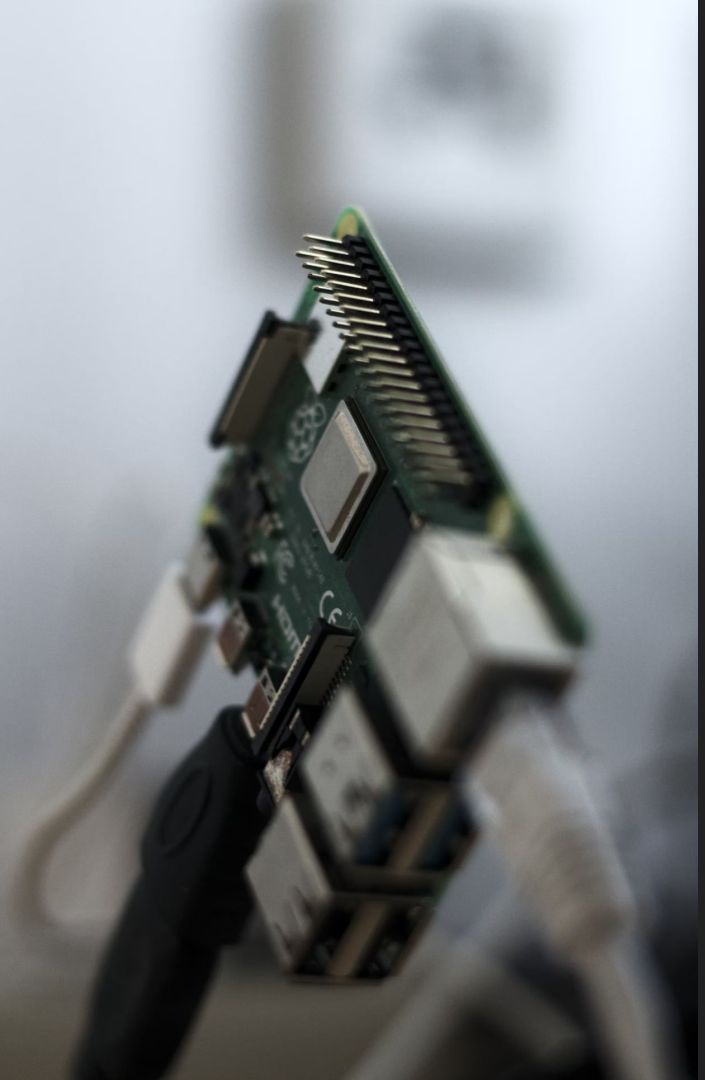
digital frontiers

Compile times

```
raspberrypi: $ cargo build --release
   Compiling libc v0.2.107
   Compiling anyhow v1.0.45
   Compiling lazy_static v1.4.0
   Compiling rppal v0.13.1
   Compiling blink v0.1.0 (...)
        Finished release [optimized] target(s) in 1m 31s
```

```
pc: $ cargo build --release
   Compiling libc v0.2.107
   Compiling anyhow v1.0.45
   Compiling lazy_static v1.4.0
   Compiling rppal v0.13.1
   Compiling blink v0.1.0 (...)
        Finished release [optimized] target(s) in 2.48s
```

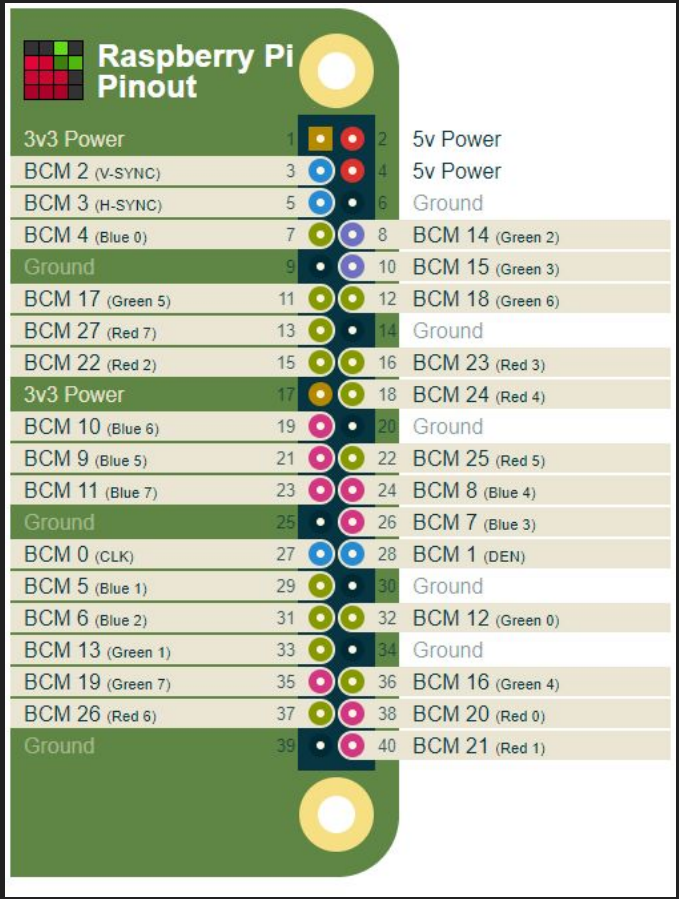
digital frontiers

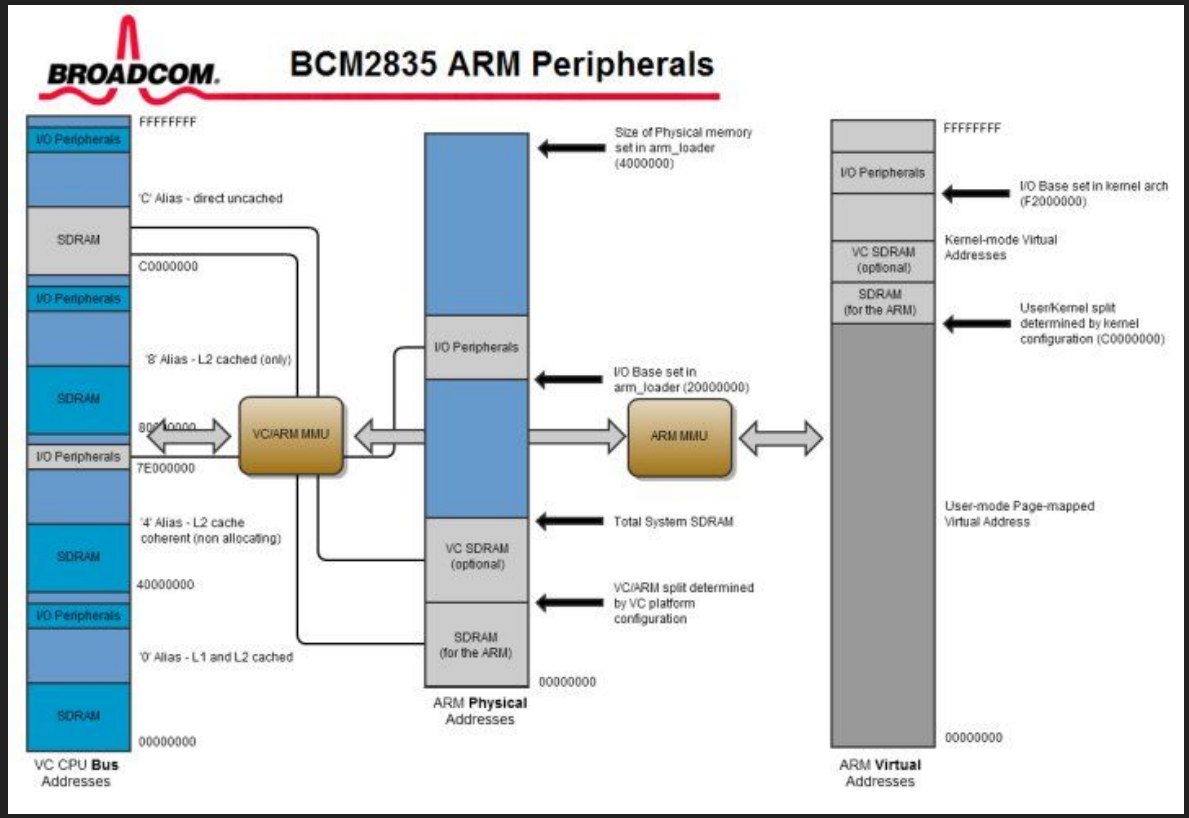


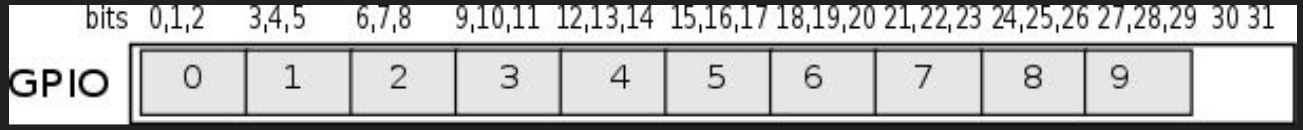
Led blink mit /dev/mem

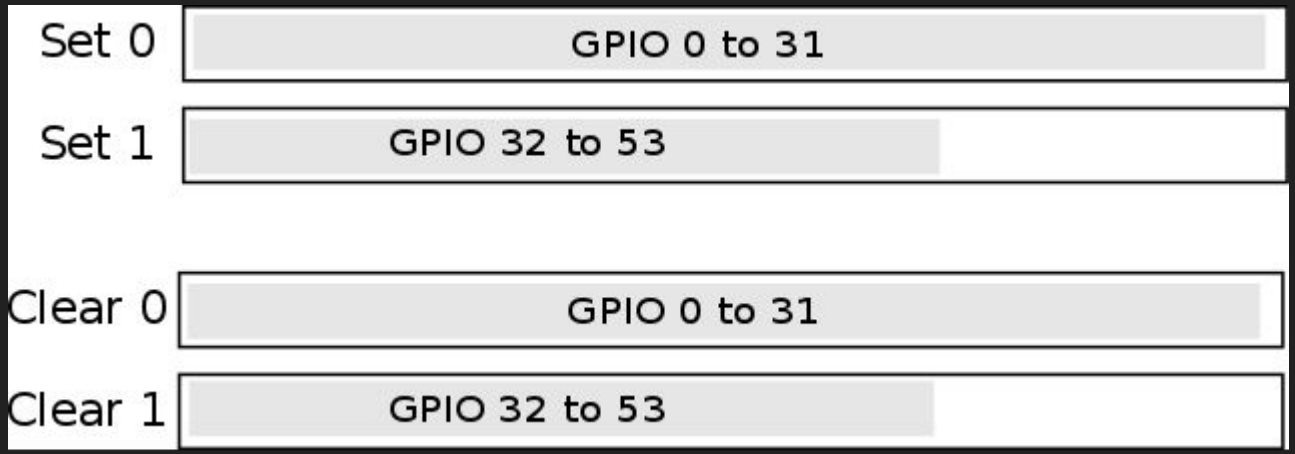
Photo by Quentin Schulz on Unsplash

digital frontiers


Raspberry Pi Pinout
3v3 Power 1 2 5v Power
BCM 2 (V-SYNC) 3 4 5v Power
BCM 3 (H-SYNC) 5 6 Ground
BCM 4 (Blue 0) 7 8 BCM 14 (Green 2)
Ground 9 10 BCM 15 (Green 3)
BCM 17 (Green 5) 11 12 BCM 18 (Green 6)
BCM 27 (Red 7) 13 14 Ground
BCM 22 (Red 2) 15 16 BCM 23 (Red 3)
3v3 Power 17 18 BCM 24 (Red 4)
BCM 10 (Blue 6) 19 20 Ground
BCM 9 (Blue 5) 21 22 BCM 25 (Red 5)
BCM 11 (Blue 7) 23 24 BCM 8 (Blue 4)
Ground 25 26 BCM 7 (Blue 3)
BCM 0 (CLK) 27 28 BCM 1 (DEN)
BCM 5 (Blue 1) 29 30 Ground
BCM 6 (Blue 2) 31 32 BCM 12 (Green 0)
BCM 13 (Green 1) 33 34 Ground
BCM 19 (Green 7) 35 36 BCM 16 (Green 4)
BCM 26 (Red 6) 37 38 BCM 20 (Red 0)
Ground 39 40 BCM 21 (Red 1)


BROADCOM.
BCM2835 ARM Peripherals
FFFFFFFF
I/O Peripherals
'C' Alias - direct uncached
SDRAM
C0000000
I/O Peripherals
'8' Alias - L2 cached (only)
SDRAM
80000000
I/O Peripherals
7E000000
'4' Alias - L2 cache coherent (non allocating)
SDRAM
40000000
I/O Peripherals
'0' Alias - L1 and L2 cached
SDRAM
00000000
VC CPU Bus Addresses
VC/ARM MMU
I/O Peripherals
VC SDRAM (optional)
SDRAM (for the ARM)
00000000
ARM Physical Addresses
Size of Physical memory set in arm_loader (4000000)
I/O Base set in arm_loader (20000000)
Total System SDRAM
VC/ARM split determined by VC platform configuration
ARM MMU
FFFFFFFF
I/O Peripherals
I/O Base set in kernel arch (F2000000)
VC SDRAM (optional)
Kernel-mode Virtual Addresses
SDRAM (for the ARM)
User/Kernel split determined by kernel configuration (C0000000)
User-mode Page-mapped Virtual Address
00000000
ARM Virtual Addresses





digital frontiers

no_std

digital frontiers

Rusts core und std Crates

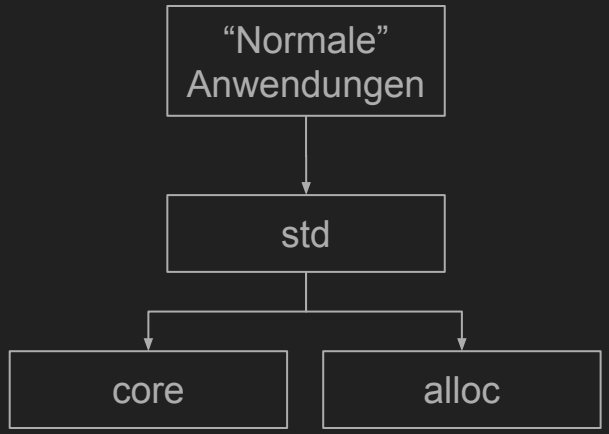
core:

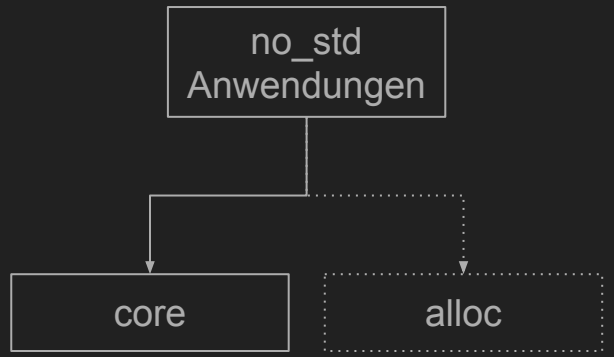
“The Rust Core Library is the dependency-free foundation of The Rust Standard Library.”

std:

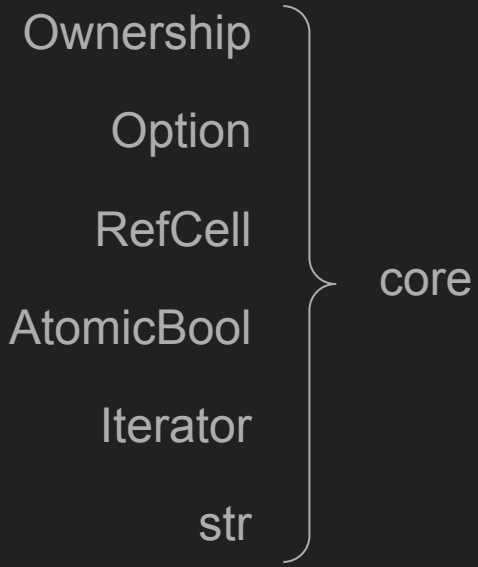
“The Rust Standard Library is the foundation of portable Rust software.”

Rusts core und std Crates

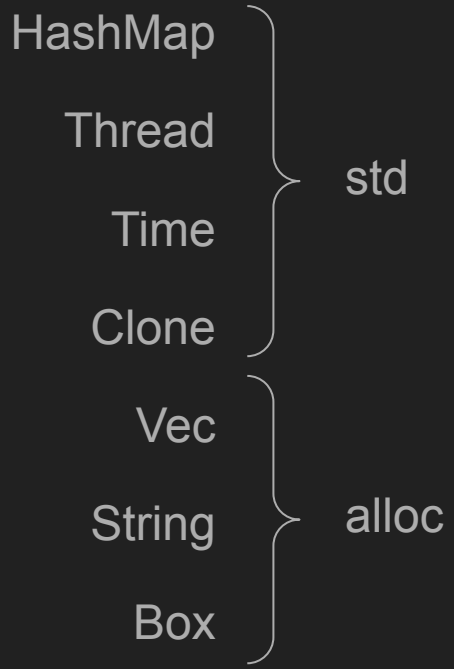




In std oder core ?



In std oder core ?



no_std Hello World

Wie startet eigentlich die Anwendung?

Photo by Malcolm Lightbody on Unsplash

digital frontiers

```
→  no_std_hello_world git:(master) ✗ readelf -h ./target/debug/hello
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Shared object file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x8520
  Start of program headers:          64 (bytes into file)
  Start of section headers:          3664256 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         14
  Size of section headers:           64 (bytes)
  Number of section headers:         43
  Section header string table index: 42
```

```
→  no_std_hello_world git:(master) ✗ nm ./target/debug/hello|grep 8520
0000000000008520 T _start
```

```
  ;-- _start:
  ;-- rip:
47: entry0 (int64_t arg3);
; arg int64_t arg3 @ rdx
0x00008520      endbr64
0x00008524      xor     ebp, ebp
0x00008526      mov     r9, rdx                         ; arg3
0x00008529      pop     rsi
0x0000852a      mov     rdx, rsp
0x0000852d      and     rsp, 0xfffffffffffffff0
0x00008531      push    rax
0x00008532      push    rsp
0x00008533      lea     r8, [__libc_csu_fini]           ; 0x3b2c0 ; sym.__libc_csu_fini
0x0000853a      lea     rcx, [__libc_csu_init]          ; 0x3b250 ; sym.__libc_csu_init
0x00008541      lea     rdi, [main]                     ; 0x8660 ; sym.main
0x00008548      call    qword [reloc.__libc_start_main] ; 0x4cc48 ; reloc.__libc_start_main(
```

digital frontiers

Aufruf

ELF -> EntryPoint: _start

_start -> __libc_start_main(main)

main
    -> std::rt::lang_start
    -> dbg.lang_start_internal
    ...
    -> dbg.main

```
50: int dbg.main (int argc, char **argv);
0x00008620      sub     rsp, 0x38                       ; hello.rs:1 ; void main();
0x00008624      mov     rdi, rsp                        ; hello.rs:2 ; int64_t arg1
0x00008627      lea     rsi, section..data.rel.ro       ; 0x4a2e8 ; int64_t arg2 ; (pstr 0x0003c000) "Hello\n"
0x0000862e      mov     edx, 1                          ; uint32_t arg3
0x00008633      lea     rcx, str.invalid_args_rustc_b416e3892d9526709f3a248f5ed3a43a970f795e_library_core_src_fmt_mod.rs ;
0x0000863a      xor     eax, eax
0x0000863c      mov     r8d, eax                        ; int64_t arg5
0x0000863f      call    core::fmt::Arguments::new_v1::h5699bbe6f50cc41c ; dbg.new_v1 ; dbg.new_v1 ; dbg.new_v1(0xfffffffff
0x00008644      mov     rdi, rsp
0x00008647      call    qword [dbg._print]              ; 0x4ce58 ; 0x4ce58(0xfffffffffffff50, 0x4a2e8, 0x1, 0x3c008)
0x0000864d      add     rsp, 0x38                       ; hello.rs:3
0x00008651      ret
0x00008652      nop     word cs:[rax + rax]
0x0000865c      nop     dword [rax]
```

Credit: Amos/fasterthanlime

```
#![no_std]
```

```
#![feature(lang_items)]
#[lang = "eh_personality"]
extern fn eh_personality() {}
```

```
#![feature(start)]

#[no_mangle]
#[start]
pub fn _start(_nargs: isize, _args: *const *const u8)-> isize {
  …
}
```

build.rs

```
fn main() {
    println!("cargo:rustc-link-arg=-nostartfiles");
}
```

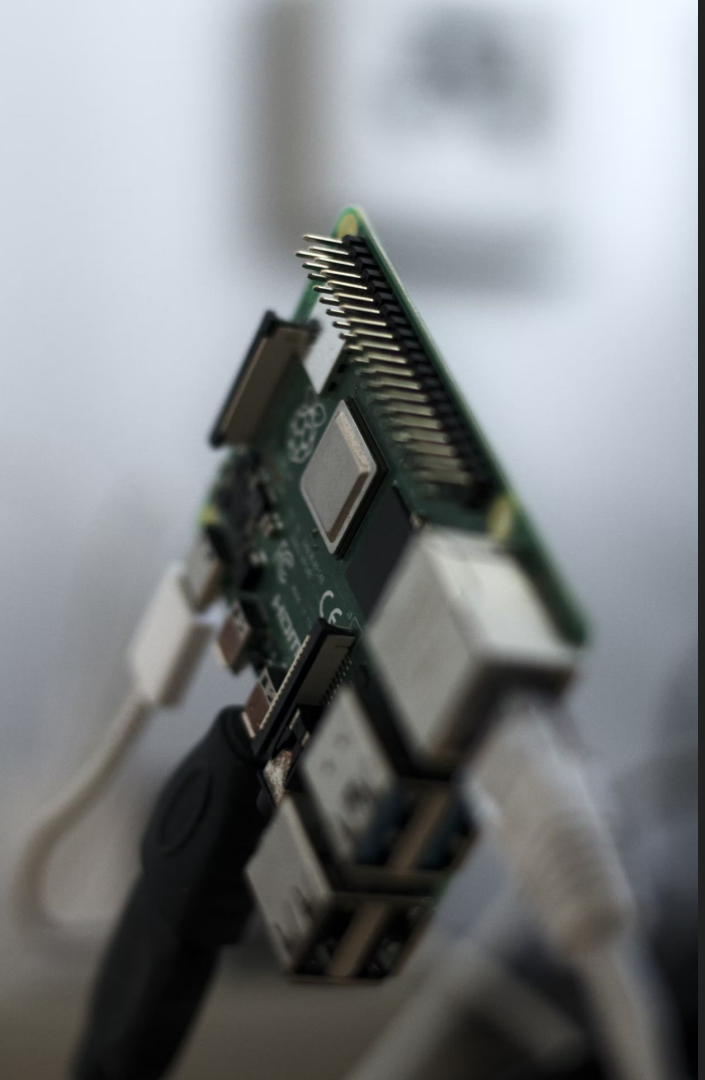
```
cargo +nightly run
```

```
→  no_std_hello_world git:(master) ✗ readelf -h target/debug/main
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Shared object file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x1050
  Start of program headers:          64 (bytes into file)
  Start of section headers:          17400 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         11
  Size of section headers:           64 (bytes)
  Number of section headers:         22
  Section header string table index: 21
```

```
45: dbg._start (int64_t arg1, int64_t arg2);
; var isize _nargs @ rsp+0x8
; var u8 **_args @ rsp+0x10
; arg int64_t arg1 @ rdi
; arg int64_t arg2 @ rsi
0x00001050      sub     rsp, 0x18                       ; main.rs:38 pub fn _start(_nargs: isize, _args: *const *const u8) -> isize {
0x00001054      mov     qword [_nargs], rdi             ; arg1
0x00001059      mov     qword [_args], rsi              ; arg2
0x0000105e      mov     edi, 1                          ; main.rs:39     write(1, b"Hello, World!\n" as *const u8, 14); ; int64_t arg1
0x00001063      lea     rsi, str.Hello__World__n        ; segment.LOAD2
                                                        ; 0x2000 ; int64_t arg2
0x0000106a      mov     edx, 0xe                        ; 14 ; int64_t arg3
0x0000106f      call    main::write::hc3760ee46272244d  ; dbg.write ; ssize_t write(int fd, const char *ptr, size_t nbytes)
0x00001074      xor     edi, edi                        ; main.rs:0 ; int64_t arg1
0x00001076      call    main::exit::h86b61fd33ee87bbd   ; main.rs:40     exit(0); ; dbg.exit
```

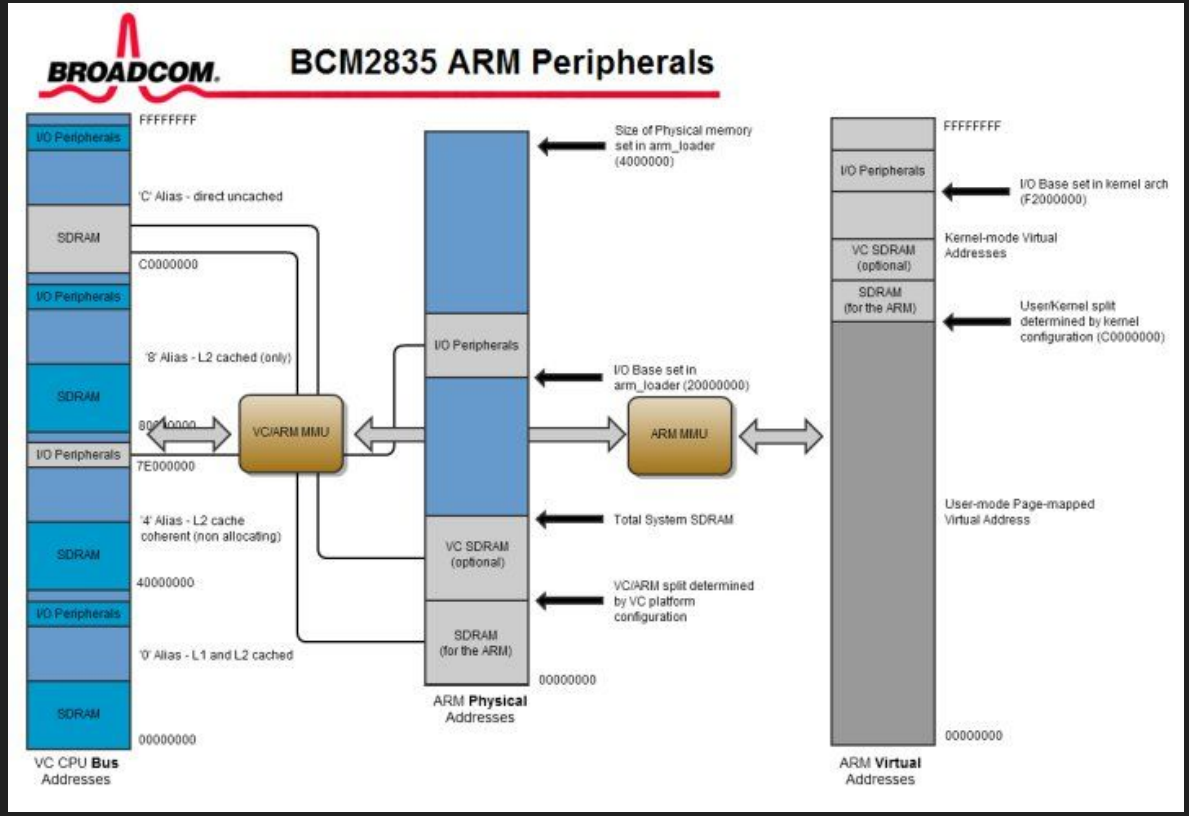
no_std Zusammenfassung

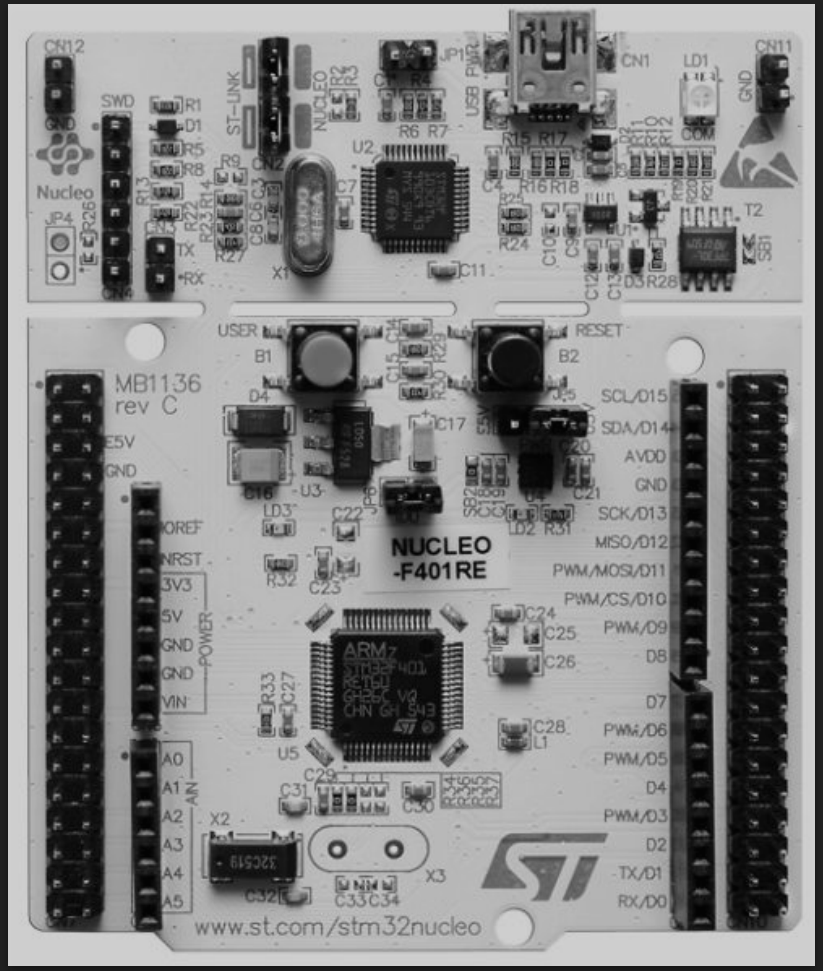
“Sicherheit von Rust” nicht in std sondern core



Led blink ohne OS

Photo by Quentin Schulz on Unsplash


BROADCOM.
BCM2835 ARM Peripherals
FFFFFFFF
I/O Peripherals
'C' Alias - direct uncached
SDRAM
C0000000
I/O Peripherals
'8' Alias - L2 cached (only)
SDRAM
80000000
I/O Peripherals
7E000000
'4' Alias - L2 cache coherent (non allocating)
SDRAM
40000000
I/O Peripherals
'0' Alias - L1 and L2 cached
SDRAM
00000000
VC CPU Bus Addresses
VC/ARM MMU
I/O Peripherals
VC SDRAM (optional)
SDRAM (for the ARM)
00000000
ARM Physical Addresses
Size of Physical memory set in arm_loader (4000000)
I/O Base set in arm_loader (20000000)
Total System SDRAM
VC/ARM split determined by VC platform configuration
ARM MMU
FFFFFFFF
I/O Peripherals
I/O Base set in kernel arch (F2000000)
VC SDRAM (optional)
Kernel-mode Virtual Addresses
SDRAM (for the ARM)
User/Kernel split determined by kernel configuration (C0000000)
User-mode Page-mapped Virtual Address
00000000
ARM Virtual Addresses



Board Support Crates
HAL/PAC

Alles was wir von Hand gemacht haben steht in crates bereit!

digital frontiers

RTIC: Real-Time Interrupt-driven Concurrency

Rust Framework

Sehr einfache Handhabung

Nicht so mächtig/überwältigend wie FreeRTOS

Habt ihr noch Fragen?

marco@amann.dev

@amann_dev

rust-buch.de

digitalfrontiers.de

#WeAreHiring

digital frontiers