

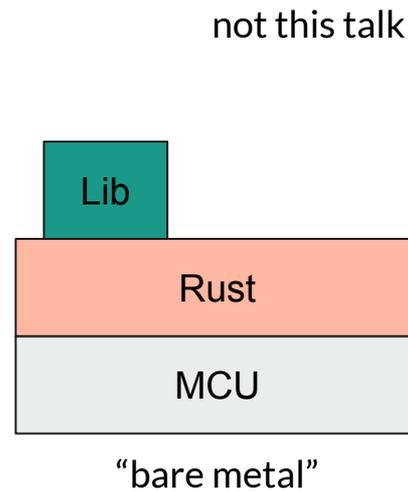
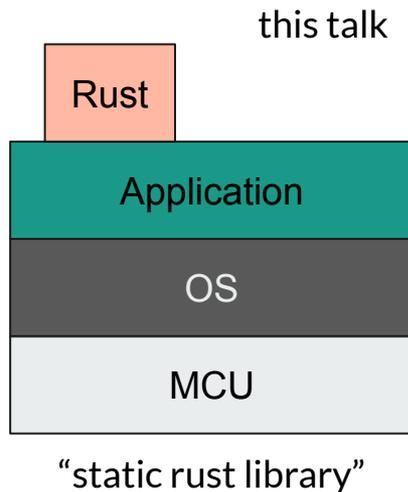


# Embedding Rust

Adding Rust to embedded projects

# Embedding Rust

- Calling Rust static library
- From existing C-ABI application





# Reasoning

There are good reasons why we not always choose “bare metal” Rust:

- Existing project
- Vendor environment (STM-IDE, NRF Connect SDK, ESP-IDF)
- OS (FreeRTOS, Zephyr, etc.)
- Certification (of RF stack)



# Reasoning

However we might still want to use Rust for some parts:

- Strict
- Safe
- Efficient
- Crates
- Async

### tm1637-gpio-driver v2.0.6

Generic GPIO driver for the TM1637 micro controller, primarily for educational purpose. Doesn't need std-lib and you can choose any GPIO interface/library you want.

[Homepage](#) [Documentation](#) [Repository](#)

↓ All-Time: 2,414

↓ Recent: 86

🕒 Updated: about 1 month ago

### aht20-driver v1.2.0

Rust embedded-hal driver for the AHT20 temperature and humidity sensor.

[Homepage](#) [Documentation](#) [Repository](#)

↓ All-Time: 383

↓ Recent: 35

🕒 Updated: 8 months ago

### ws2818-rgb-led-spi-driver v2.0.0

Simple, stripped down, educational, no\_std-compatible driver for WS28XX (WS2811/12) RGB LEDs. Use: SPI device for timing/clock, and works definitely on Linux/Raspberry Pi.

[Homepage](#) [Documentation](#) [Repository](#)

↓ All-Time: 2,542

↓ Recent: 178

🕒 Updated: over 1 year ago

### xca9548a v0.2.1

Platform-agnostic Rust driver for the TCA954xA and PCA954xA I2C switches/multiplexers.

[Homepage](#) [Documentation](#) [Repository](#)

↓ All-Time: 2,343

↓ Recent: 130

🕒 Updated: about 2 years ago

### embedded-graphics v0.7.1

Embedded graphics library for small hardware displays

[Documentation](#) [Repository](#)

↓ All-Time: 289,725

↓ Recent: 38,014

🕒 Updated: over 1 year ago

### stepper-driver v0.1.0

A49xx and DRV88xx stepper motor driver.

[Repository](#)

↓ All-Time: 432

↓ Recent: 9

🕒 Updated: over 4 years ago



# Difficulty

Rust makes it easy!

- Extern: just like a C-style function
- No-mangle: exact symbol names
- `#[repr(C)]`: memory layout is identical to C



# Difficulty

Rust makes it sometimes not so easy...

- Need to check a lot more (but you should've probably checked that regardless)
- Memory ownership is complicated (but it always was)
- Rust can “panic”



# Difficulty

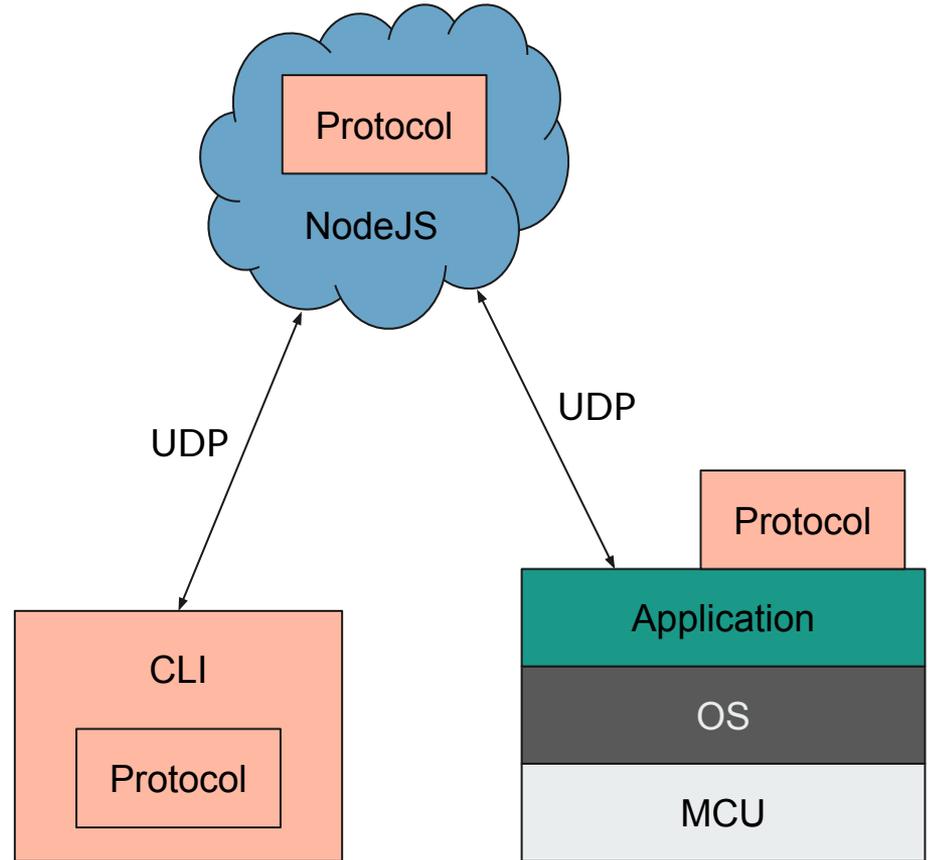
What makes Rust loved is laid bare when integrating into foreign systems (FFI).

Because now you have to do a lot of it manually.

# Cases - Protocol

Implementation of byte-packet protocol in Rust.

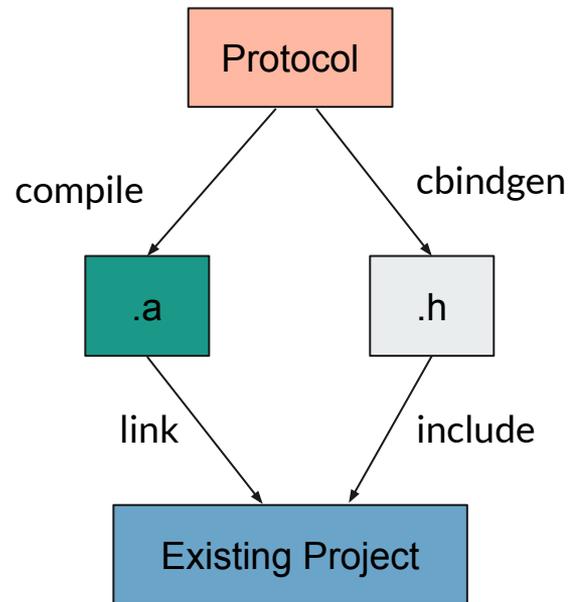
- Embedded in C
- Server in NodeJS
- CLI in Rust



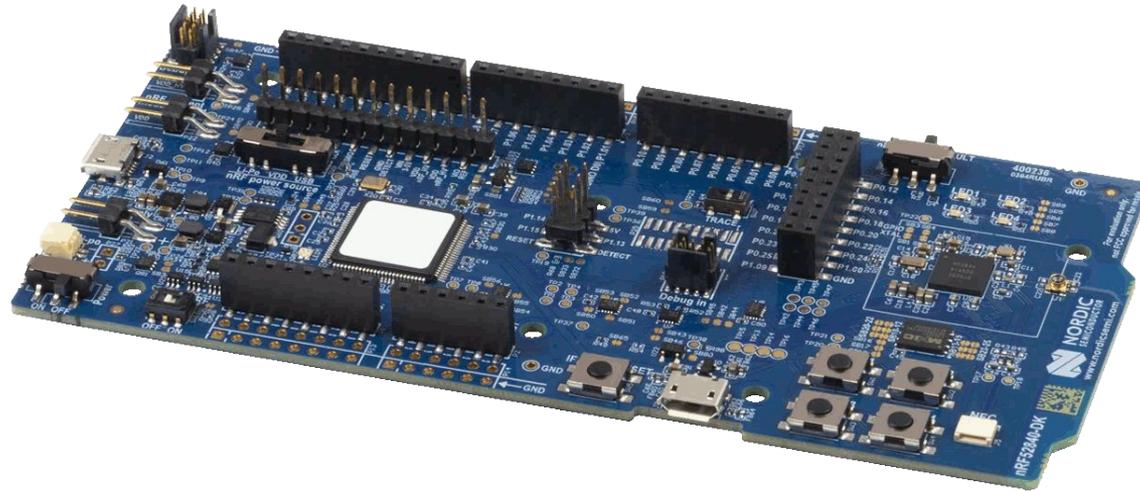
# Approach

Presentation of a simplified protocol.

- Define FFI boundary
- Compile static `.a` library
- Run `cbindgen`, get header file
- Include headers & link to library
- Call Rust code!



# Cases - Protocol





## Cases - Protocol

```
1 #![no_std]
2
3 use serde::Serialize;
4
5 #[derive(Serialize)]
6 #[repr(C)]
7 pub enum Level {
8     Info,
9     Debug,
10 }
11
12 #[derive(Serialize)]
13 #[repr(C)]
14 pub struct Message {
15     recipient: Recipient,
16     ttl: u16,
17     level: Level,
18 }
```

```
1 #include <stdarg.h>
2 #include <stdbool.h>
3 #include <stddef.h>
4 #include <stdint.h>
5 #include <stdlib.h>
6
7
8
9 typedef enum Level {
10     Info,
11     Debug,
12 } Level;
13
14 typedef struct Message {
15     struct Recipient recipient;
16     uint16_t ttl;
17     enum Level level;
18 } Message;
```

```
1 #[repr(C)]
2 pub struct Recipient {
3     // Starting pointer to a C-style string representing the name of the recipient.
4     ptr: *const c_uchar,
5     /// Length of the C-style string in bytes, including null character at end.
6     len: usize,
7 }
8
9 impl Serialize for Recipient {
10     fn serialize<S>(&self, serializer: S) -> Result<S::Ok, S::Error>
11     where
12         S: serde::Serializer,
13     {
14         let slice = unsafe { core::slice::from_raw_parts(self.ptr, self.len) };
15         if let Ok(Ok(s)) = core::ffi::CStr::from_bytes_with_nul(slice).map(|cstr| cstr.to_str()) {
16             serializer.serialize_str(s)
17         } else {
18             serializer.serialize_none()
19         }
20     }
21 }
```



## Cases - Protocol

```
1 #[no_mangle]
2 pub extern "C" fn serialize_message(
3     msg: *const Message,
4     dst_ptr: *mut u8,
5     dst_len: usize,
6 ) -> usize {
7     let slice = unsafe { core::slice::from_raw_parts_mut(dst_ptr, dst_len) };
8     if let Ok(len) = serde_json_core::to_slice(unsafe { &*msg }, slice) {
9         len
10    } else {
11        0
12    }
13 }
```

```
1 size_t serialize_message(const struct Message *msg, uint8_t *dst_ptr, size_t dst_len);
```



## Cases - Protocol

```
1 [package]
2 name = "protocol"
3 version = "0.1.0"
4 edition = "2021"
5
6 [dependencies]
7 serde = { version = "1.0", features = ["derive"], default-features = false }
8 serde-json-core = { version = "0.4", default-features = false }
9
10 [build-dependencies]
11 cbindgen = "0.24"
12
13 [lib]
14 crate-type = ["staticlib", "lib"]
```



## Cases - Protocol

```
wouter@wgeraedts-sleipnir:~/sarif/embedded-integration/examples/protocol$ cargo build --release
  Compiling protocol v0.1.0 (/home/wouter/sarif/embedded-integration/examples/protocol)
error: `#[panic_handler]` function required, but not found

error: could not compile `protocol` due to previous error
```

```
1 #![no_std]
2
3 extern crate panic_halt;
```



## Cases - Protocol

```
wouter@wgeraedts-sleipnir:~/sarif/embedded-integration/examples/protocol$ ls -la ./protocol.h
-rw-rw-r-- 1 wouter wouter 732 nov  4 09:41 ./protocol.h
wouter@wgeraedts-sleipnir:~/sarif/embedded-integration/examples/protocol$ ls -la target/thumbv7em-none-eabi/release/libprotocol.a
-rw-rw-r-- 2 wouter wouter 6708386 nov  4 11:06 target/thumbv7em-none-eabi/release/libprotocol.a
```



# Cases - Protocol

```
1 #include <zephyr/zephyr.h>
2 #include <string.h>
3
4 #include <protocol.h>
5
6 static const size_t BUF_LEN = 128;
7
8 void main(void)
9 {
10     char *str = "recipient A";
11
12     Message message = {
13         .recipient = {
14             .ptr = str,
15             .len = strlen(str) + 1,
16         },
17         .ttl = 5,
18         .level = Debug,
19     };
20
21     uint8_t buf[BUF_LEN];
22     size_t len = serialize_message(&message, buf, BUF_LEN);
23
24     if (len == 0)
25     {
26         printk("Serialization failed\n");
27     }
28     else
29     {
30         buf[len] = 0;
31         printk("%s\n", buf);
32     }
33 }
```



# Cases - Protocol

```
wouter@wgeraeds-sleipnir: ~/sarif/embedded-integration/examples/integrated/thread-node
wouter@wgeraeds-sleipnir:~/sarif/embedded-integration/examples/integrated/thread-node$ west flash
-- west flash: rebuilding
ninja: no work to do.
-- west flash: using runner nrfjprog
Using board 683958415
-- runners.nrfjprog: Flashing file: /home/wouter/sarif/embedded-integration/examples/integrated/thread-node/build/zephyr/zephyr.hex
[ ##### ] 0.927s | Erase file - Done erasing
[ ##### ] 0.399s | Program file - Done programming
[ ##### ] 0.388s | Verify file - Done verifying
Enabling pin reset.
Applying pin reset.
-- runners.nrfjprog: Board with serial number 683958415 flashed successfully.
wouter@wgeraeds-sleipnir:~/sarif/embedded-integration/examples/integrated/thread-node$ █

wouter@wgeraeds-sleipnir: ~
*** Booting Zephyr OS build v3.1.99-ncs1 ***
{"recipient":"recipient A","ttl":5,"level":"Debug"}
█
```



# Conclusions

- Calling Rust from your embedded C is easy.
- But there are some catches and caveats, so be careful.
- Rust can be a great addition to your project.



## Further Reading

Interoperability: “a little Rust with your C”:

<https://docs.rust-embedded.org/book/interoperability/rust-with-c.html>

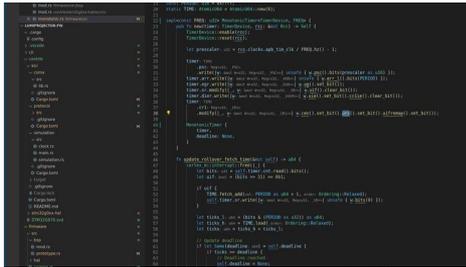
Tips for C developers:

<https://docs.rust-embedded.org/book/c-tips/index.html>

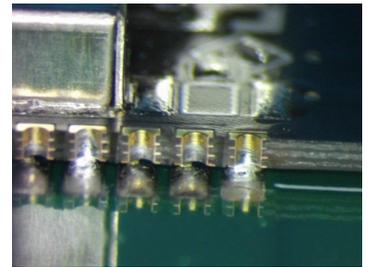
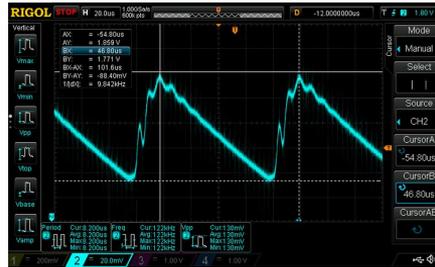
# Need help?

Freelance embedded software & hardware engineer

<https://sarif.nl>



```
1 // ...
2 #include <Arduino.h>
3 #include <SPI.h>
4 #include <Wire.h>
5 #include <EEPROM.h>
6 #include <SD.h>
7 #include <RTClib.h>
8 #include <Adafruit_GFX.h>
9 #include <Adafruit_ILI9341.h>
10 #include <Adafruit_BMP280.h>
11 #include <Adafruit_BMP280_I2C.h>
12 #include <Adafruit_BMP280_SPI.h>
13 #include <Adafruit_BMP280_I2C.h>
14 #include <Adafruit_BMP280_SPI.h>
15 #include <Adafruit_BMP280_I2C.h>
16 #include <Adafruit_BMP280_SPI.h>
17 #include <Adafruit_BMP280_I2C.h>
18 #include <Adafruit_BMP280_SPI.h>
19 #include <Adafruit_BMP280_I2C.h>
20 #include <Adafruit_BMP280_SPI.h>
21 #include <Adafruit_BMP280_I2C.h>
22 #include <Adafruit_BMP280_SPI.h>
23 #include <Adafruit_BMP280_I2C.h>
24 #include <Adafruit_BMP280_SPI.h>
25 #include <Adafruit_BMP280_I2C.h>
26 #include <Adafruit_BMP280_SPI.h>
27 #include <Adafruit_BMP280_I2C.h>
28 #include <Adafruit_BMP280_SPI.h>
29 #include <Adafruit_BMP280_I2C.h>
30 #include <Adafruit_BMP280_SPI.h>
31 #include <Adafruit_BMP280_I2C.h>
32 #include <Adafruit_BMP280_SPI.h>
33 #include <Adafruit_BMP280_I2C.h>
34 #include <Adafruit_BMP280_SPI.h>
35 #include <Adafruit_BMP280_I2C.h>
36 #include <Adafruit_BMP280_SPI.h>
37 #include <Adafruit_BMP280_I2C.h>
38 #include <Adafruit_BMP280_SPI.h>
39 #include <Adafruit_BMP280_I2C.h>
40 #include <Adafruit_BMP280_SPI.h>
41 #include <Adafruit_BMP280_I2C.h>
42 #include <Adafruit_BMP280_SPI.h>
43 #include <Adafruit_BMP280_I2C.h>
44 #include <Adafruit_BMP280_SPI.h>
45 #include <Adafruit_BMP280_I2C.h>
46 #include <Adafruit_BMP280_SPI.h>
47 #include <Adafruit_BMP280_I2C.h>
48 #include <Adafruit_BMP280_SPI.h>
49 #include <Adafruit_BMP280_I2C.h>
50 #include <Adafruit_BMP280_SPI.h>
51 #include <Adafruit_BMP280_I2C.h>
52 #include <Adafruit_BMP280_SPI.h>
53 #include <Adafruit_BMP280_I2C.h>
54 #include <Adafruit_BMP280_SPI.h>
55 #include <Adafruit_BMP280_I2C.h>
56 #include <Adafruit_BMP280_SPI.h>
57 #include <Adafruit_BMP280_I2C.h>
58 #include <Adafruit_BMP280_SPI.h>
59 #include <Adafruit_BMP280_I2C.h>
60 #include <Adafruit_BMP280_SPI.h>
61 #include <Adafruit_BMP280_I2C.h>
62 #include <Adafruit_BMP280_SPI.h>
63 #include <Adafruit_BMP280_I2C.h>
64 #include <Adafruit_BMP280_SPI.h>
65 #include <Adafruit_BMP280_I2C.h>
66 #include <Adafruit_BMP280_SPI.h>
67 #include <Adafruit_BMP280_I2C.h>
68 #include <Adafruit_BMP280_SPI.h>
69 #include <Adafruit_BMP280_I2C.h>
70 #include <Adafruit_BMP280_SPI.h>
71 #include <Adafruit_BMP280_I2C.h>
72 #include <Adafruit_BMP280_SPI.h>
73 #include <Adafruit_BMP280_I2C.h>
74 #include <Adafruit_BMP280_SPI.h>
75 #include <Adafruit_BMP280_I2C.h>
76 #include <Adafruit_BMP280_SPI.h>
77 #include <Adafruit_BMP280_I2C.h>
78 #include <Adafruit_BMP280_SPI.h>
79 #include <Adafruit_BMP280_I2C.h>
80 #include <Adafruit_BMP280_SPI.h>
81 #include <Adafruit_BMP280_I2C.h>
82 #include <Adafruit_BMP280_SPI.h>
83 #include <Adafruit_BMP280_I2C.h>
84 #include <Adafruit_BMP280_SPI.h>
85 #include <Adafruit_BMP280_I2C.h>
86 #include <Adafruit_BMP280_SPI.h>
87 #include <Adafruit_BMP280_I2C.h>
88 #include <Adafruit_BMP280_SPI.h>
89 #include <Adafruit_BMP280_I2C.h>
90 #include <Adafruit_BMP280_SPI.h>
91 #include <Adafruit_BMP280_I2C.h>
92 #include <Adafruit_BMP280_SPI.h>
93 #include <Adafruit_BMP280_I2C.h>
94 #include <Adafruit_BMP280_SPI.h>
95 #include <Adafruit_BMP280_I2C.h>
96 #include <Adafruit_BMP280_SPI.h>
97 #include <Adafruit_BMP280_I2C.h>
98 #include <Adafruit_BMP280_SPI.h>
99 #include <Adafruit_BMP280_I2C.h>
100 #include <Adafruit_BMP280_SPI.h>
```





```
1 type TransmitFunction = unsafe extern "C" fn(*const u8, usize) -> usize;
2
3 #[repr(C)]
4 pub struct Parameters {
5     transmit: TransmitFunction,
6 }
7
8 pub struct Context {
9     parameters: Parameters,
10    state: State,
11 }
12
13 #[no_mangle]
14 pub extern "C" fn context_size() -> usize {
15     core::mem::size_of::<Context>()
16 }
17
18 #[no_mangle]
19 pub extern "C" fn create_context(parameters: Parameters, context: *mut Context) {
20     let context: *mut MaybeUninit<Context> = unsafe { core::mem::transmute(context) };
21     let context: &mut MaybeUninit<Context> = unsafe { context.as_mut() }.unwrap();
22
23     let state: State = State::default();
24     context.write(Context {
25         parameters,
26         state
27     });
28 }
```