

A tale of Rust, the ESP32 and IoT

It can't be that hard...

Who am I?

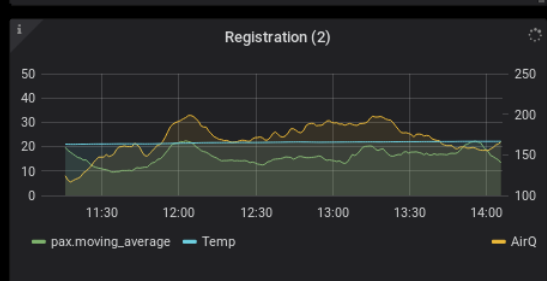
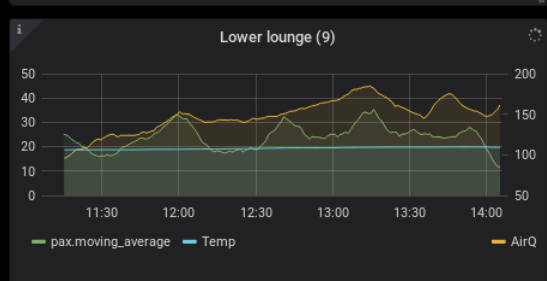
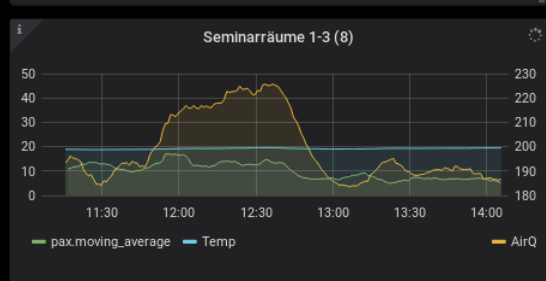
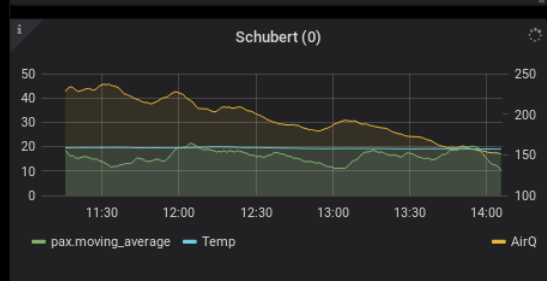
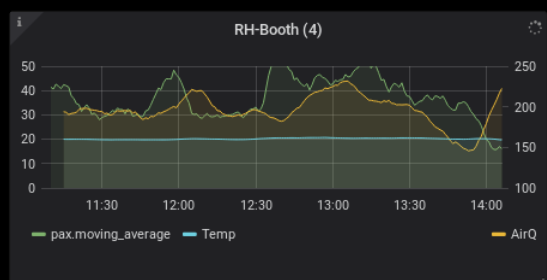
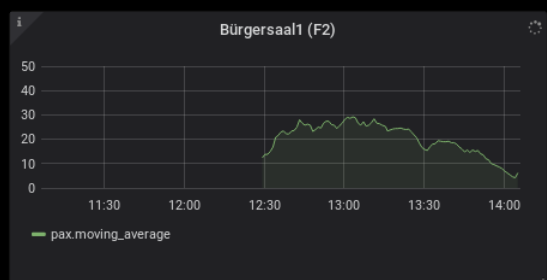
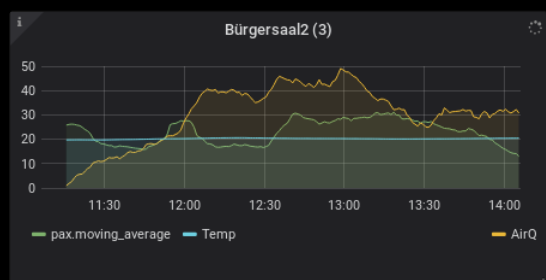
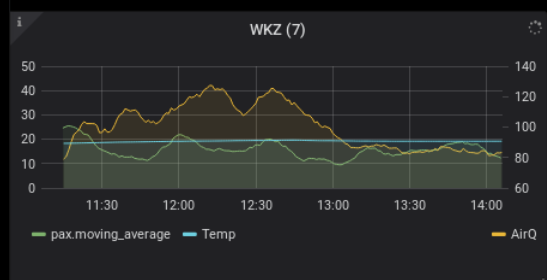
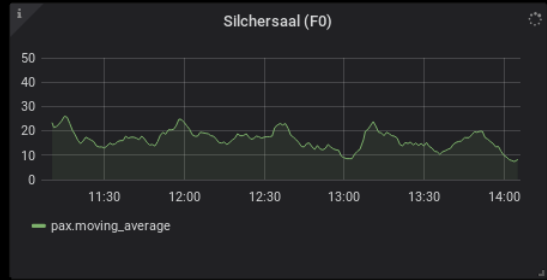
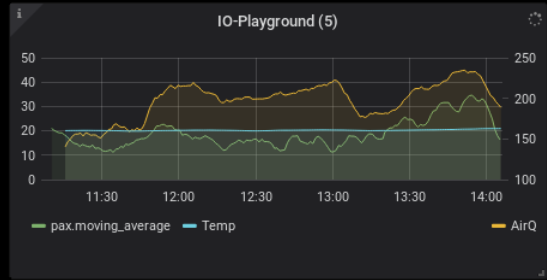
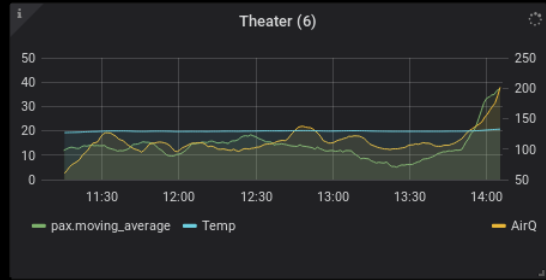
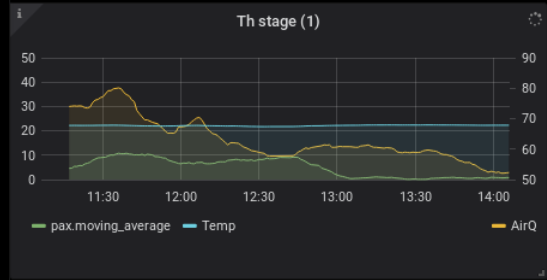
Jens Reimann

- Principal Software Engineer
- Red Hat
 - Middleware, Messaging, IoT
- Programming languages
 - 90s: Basic, Pascal, C
 - 00s: C, C++, Java
 - 10s: Java, Go, Rust

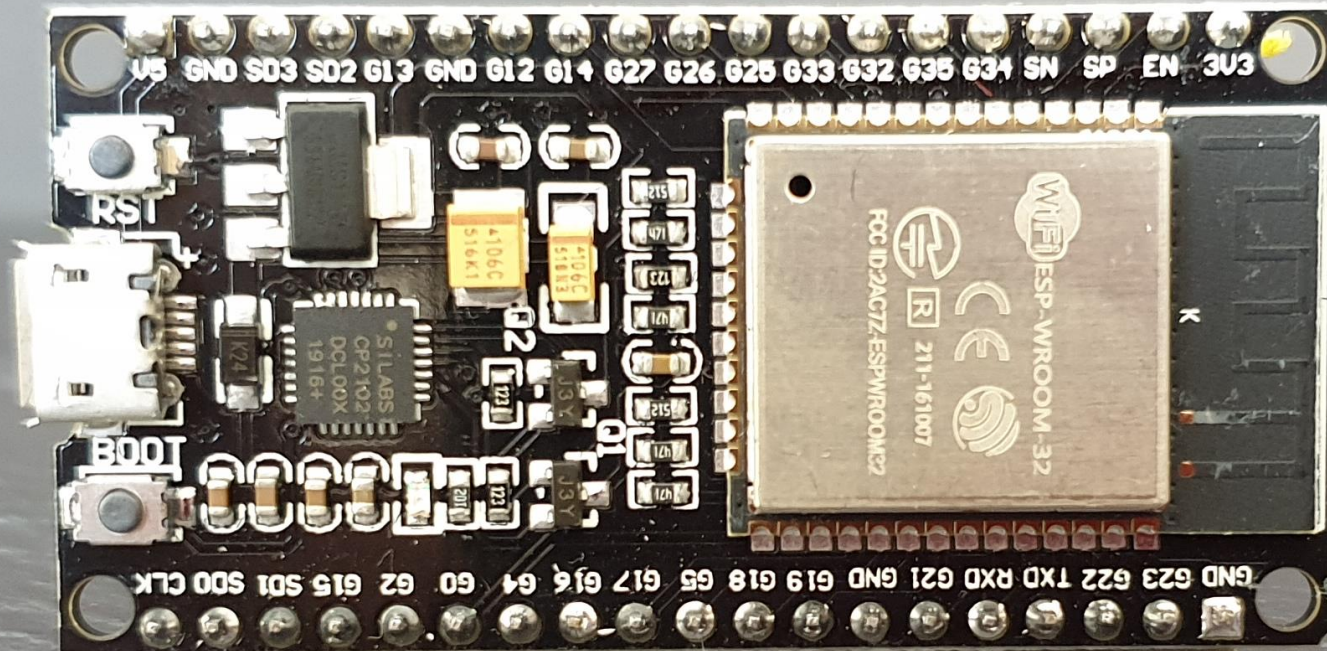


@ctr0n

<https://dentrassi.de>



Telemetry data for Eclipse Hono



Goal of this talk?

Get you excited about Rust!



Why Rust?

Do we really need another programming language?



Rust

"A language empowering everyone to build reliable and efficient software."

<https://www.rust-lang.org/>



Rust

"Rust is a language for systems programming."

Jim Blandy & Jason Orendorff, *Programming Rust*

"Systems programming is for:

...

- Code that runs in very cheap devices, or devices that must be extremely reliable

...

"



Rust is ...

"A safe, concurrent language with the performance of C and C++"

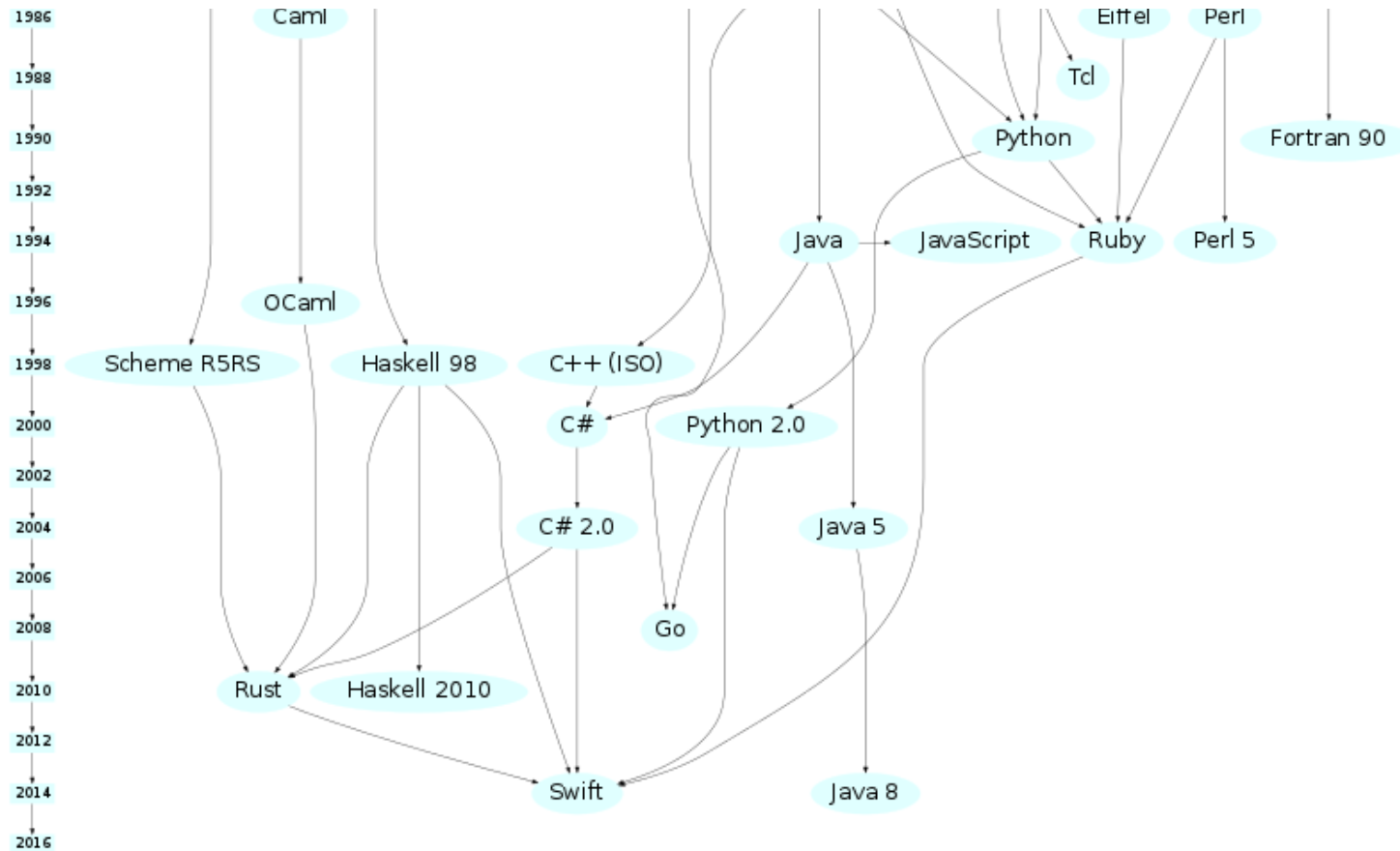
Jim Blandy & Jason Orendorff, *Programming Rust*

"...eliminate many classes of bugs at compile-time."

<https://www.rust-lang.org/>



History of languages

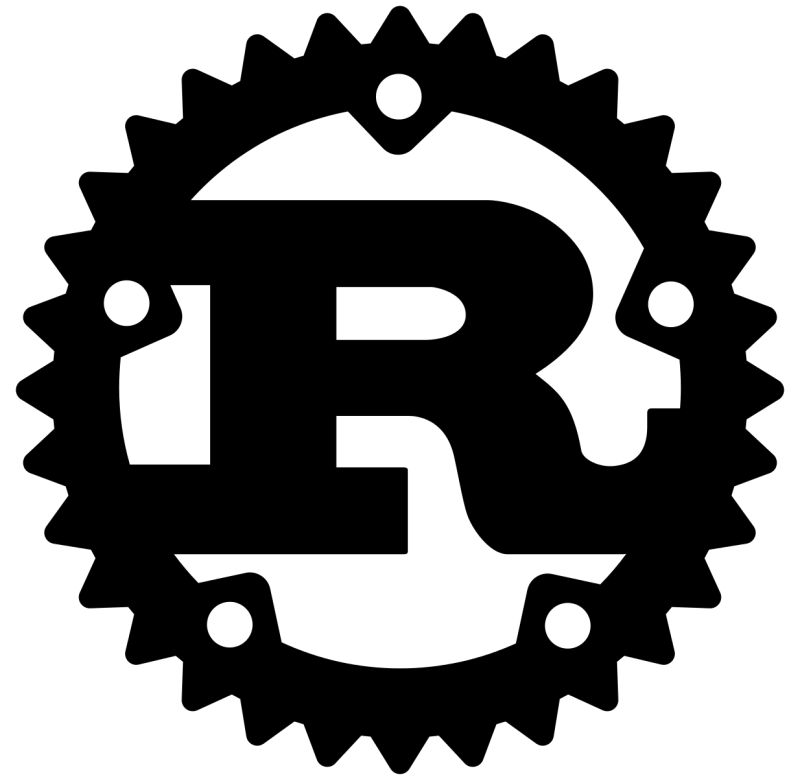


History of Rust

- Started 2006 by Mozilla employee Graydon Hoare
- Announced 2010 by the Mozilla Foundation
- Self-compiled 2011
- Getting things right, before moving on
- A community to grow the language, not only use it

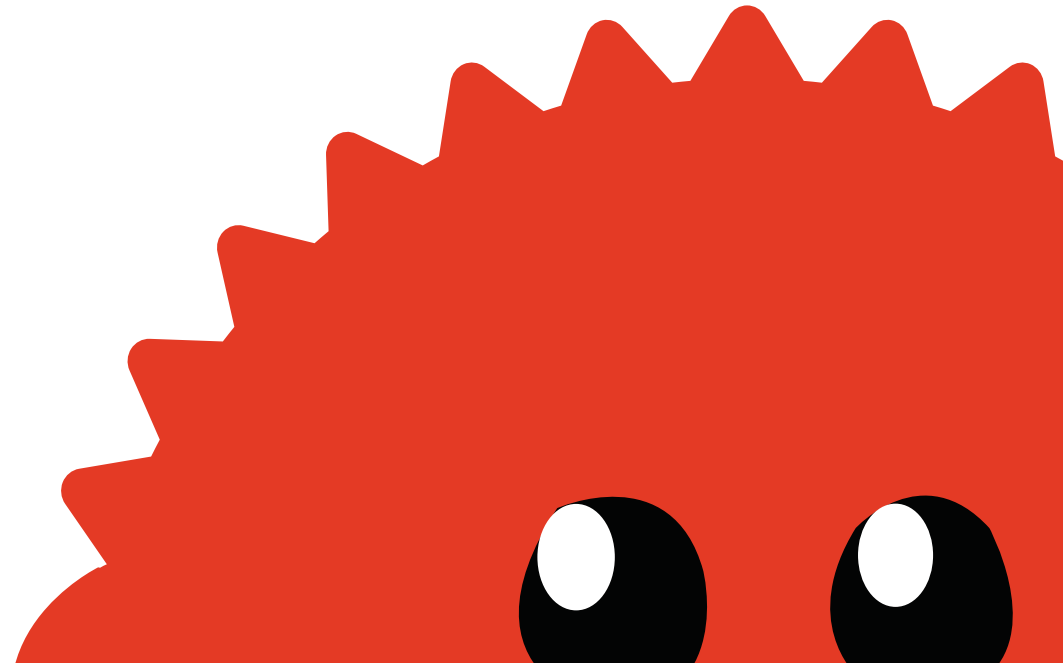


“Rust was the third-most-loved programming language in the 2015 Stack Overflow annual survey, **and took first place in 2016, 2017, 2018, and 2019.**” – Wikipedia



Fixing stuff at compile time

- Have a compiler which understands your code
- Have language rules which prevent bugs
- Eliminate “undefined behavior”
- Reduce “unexpected behavior”



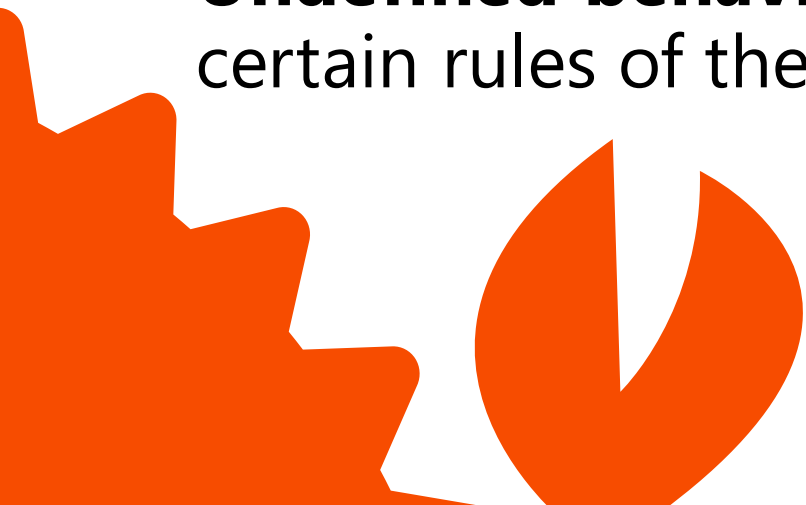
What is undefined behavior?

Undefined behavior: behavior, upon use of a nonportable or erroneous program construct or of erroneous data, for which this International Standard imposes no requirements

§3.4.3, Cx11

Undefined behavior: Renders the entire program meaningless if certain rules of the language are violated.

cppreference.com





```
panic(cpu 3 caller 0xfffff801ed92ad21): "Process 1 exec of /sbin/launchd failed, error: VM Swap Subsystem is ON"
Debugger called: <panic>
Backtrace (CPU 3), Frame : Return Address
0xfffff80b421bdf8 : 0xfffff801e92ad21
0xfffff80b421be78 : 0xfffff801ed92ad21
0xfffff80b421bef8 : 0xfffff801ed92ad21
0xfffff80b421bf18 : 0xfffff801edd9011
0xfffff80b421bf58 : 0xfffff801e927256
0xfffff80b421bf88 : 0xfffff801ea1756e
0xfffff80b421bfa8 : 0xfffff801ea33c6f

BSD process name corresponding to current thread: Init

Mac OS version:
Not yet set

Kernel Version:
Kernel Version 14.5.0: Wed Jul 29 02:26:53 PDT 2015; root:xnu-2782.40.9~1/RELEASE_ARM_T8020
ID: 50F06365-45C7-3CA7-B880-173AFD1A83C4
Model: MacBookPro6,2 (Mac-F22586C8)
Time in nanoseconds: 3512701434
```



Undefined / unexpected behavior

Go FAQ

Q: „Why are map operations not defined to be atomic? “

A: „...This was not an easy decision, however, since it means uncontrolled map access can crash the program....“

Java „CME“ - HashSet

“Fail-fast iterators throw ConcurrentModificationException on a best-effort basis. Therefore, it would be wrong to write a program that depended on this exception for its correctness: the fail-fast behavior of iterators should be used only to detect bugs.”





A helping compiler...

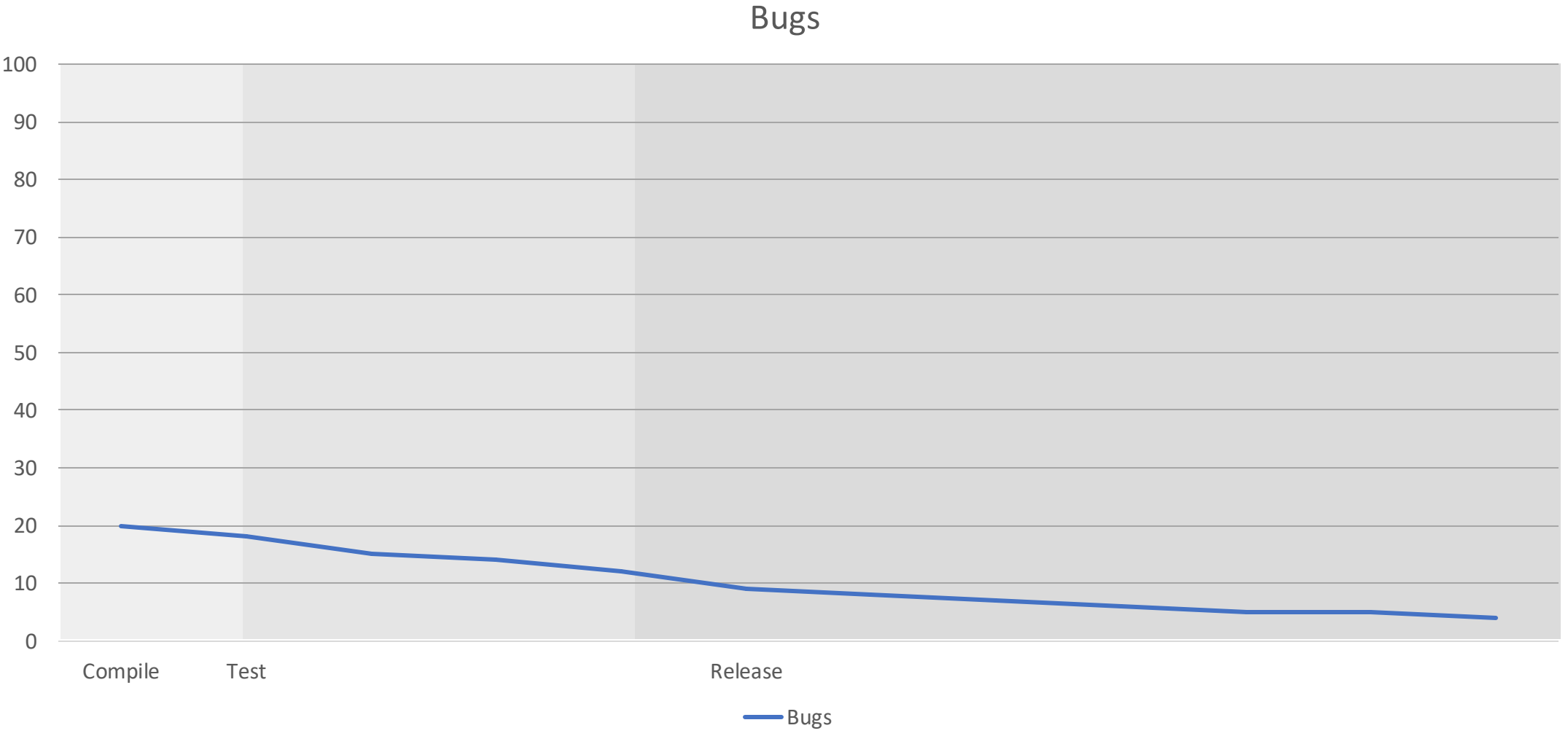
Code

```
fn main() {  
  
    let mut s1 = "Foo";  
    let mut x = &mut s1;  
    assert_eq!(*x, "Foo");  
  
    {  
        let mut s2 = "Bar";  
        x = &mut s2;  
    }  
  
    assert_eq!(*x, "Bar");  
  
}
```

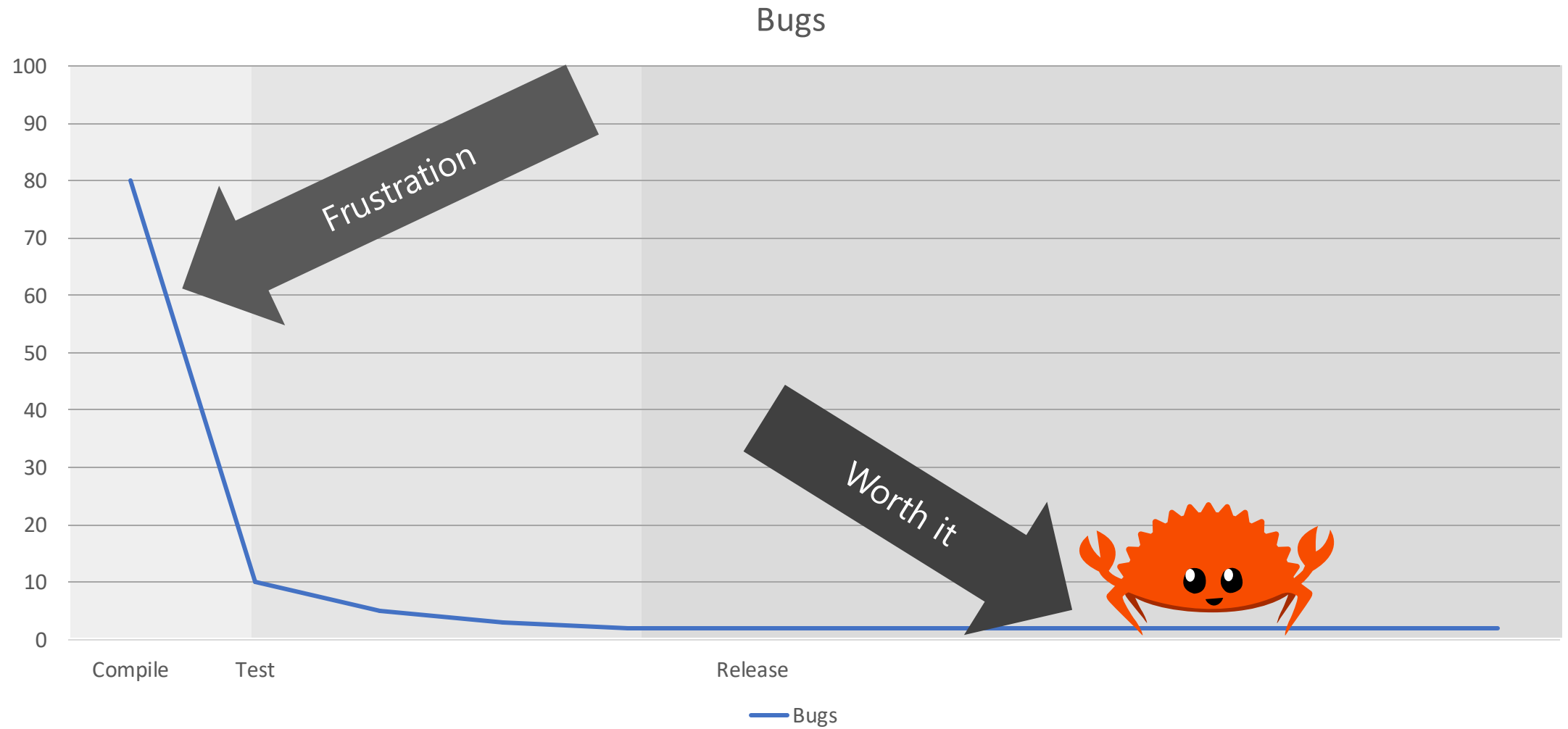
Output

```
Compiling playground v0.0.1 (/playground)  
error[E0597]: `s2` does not live long enough  
--> src/main.rs:8:13  
8 |         x = &mut s2;  
   |             ^^^^^^^ borrowed value does not live long enough  
9 |     }  
   | - `s2` dropped here while still borrowed  
10 |     assert_eq!(*x, "Bar");  
   | ----- borrow later used here  
  
= note: this error originates in a macro outside of the current crate (in  
Nightly builds, run with -Z external-macro-backtrace for more info)  
  
error: aborting due to previous error  
  
For more information about this error, try `rustc --explain E0597`.  
error: Could not compile `playground`.  
  
To learn more, run the command again with --verbose.
```

The cost of a bug over time...

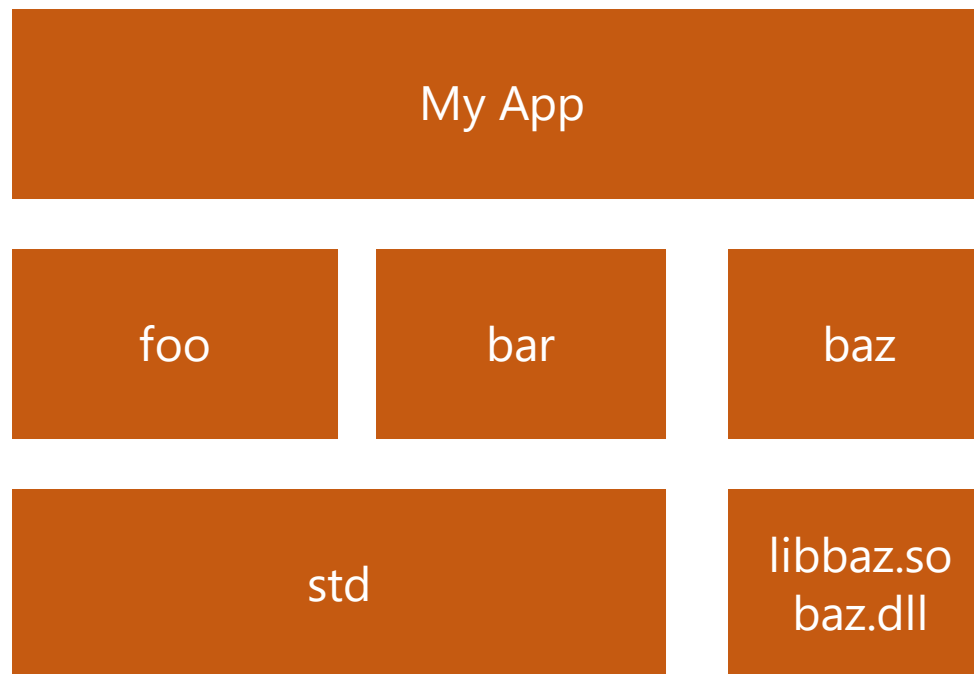


...with Rust



Dependencies

- “Crates” are the “JAR files of Rust” ... but contain code, not binaries!
- Then “crates.io” is “Maven Central for Rust”
- “cargo” manages dependencies, and orchestrates the build and test



The problem with „std“

- „std“ provides all kinds of functionality
 - Files, Streams, ...
 - Network, Sockets, ...
 - ...
- But also requires a POSIX-like operating system
- So, what about embedded systems? Like the ESP32?



#![no_std], "core" & "alloc"

- You can disable the usage of "std" and switch to "core" instead
- If you can provide an allocator, you can also use "alloc" for dynamic memory allocations (like String, Vec, ...)
- Some crates support this by using "features", which enable/disable features of the crate at compile time
 - e.g. "serde" with "serde-json-core"

<https://crates.io/keywords/nostd>



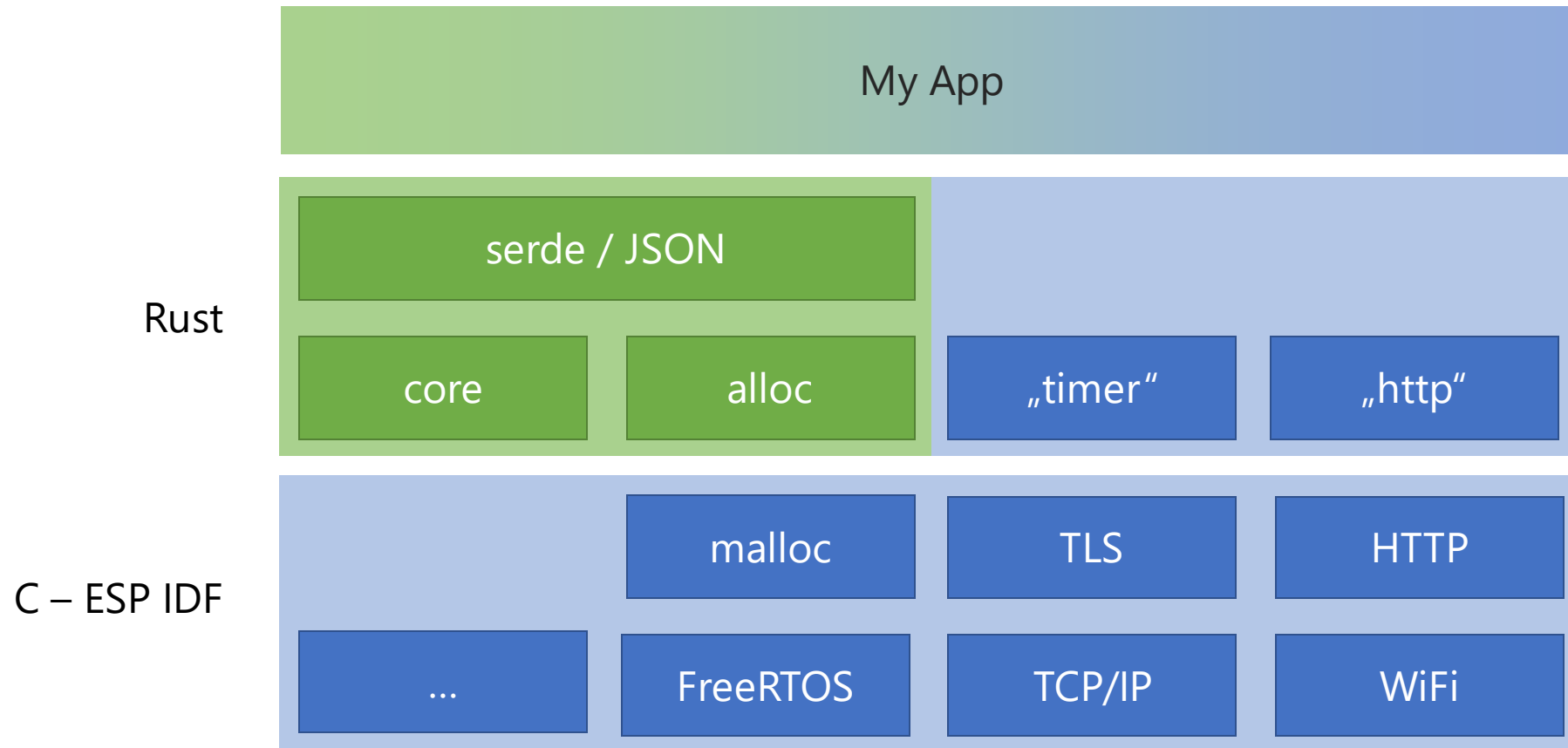
Unsafe Superpowers

- How?
 - "unsafe {}" block
 - "unsafe" keyword
- Why?
 - Call other „unsafe“ methods
 - Dereference raw pointers
 - ...

```
let base = Box::new(TimerBase {  
    callback: Box::new(callback),  
});  
  
unsafe {  
    let ptr = Box::into_raw(base) as *mut c_void;  
  
    let timer = xTimerCreate(  
        b"Timer\0".as_ptr() as *const i8,  
        period_ticks,  
        reload,  
        ptr,  
        Some(Timer::handler),  
    );  
  
    Timer { handle: timer }  
}
```



ESP-IDF & Rust



End-to-end example

```
110 #[no_mangle]
111 pub fn app_main() {
112     log::log(Level::INFO, &TAG, format_args!("Hello World"));
113
114     init_global_ca_store();
115
116     let config = http::HttpClientConfig {
117         url: HONO_HTTP_ADAPTER_URL,
118         authentication_type: Some(http::AuthenticationType::BASIC),
119         authentication_header: Some(HONO_AUTH_HEADER),
120         method: http::Method::POST,
121         ..Default::default()
122     };
123
124     let mut http = http::HttpClient::new(&config).expect("Failed to init HTTP client");
125
126     let mut app = app::Application::new(WIFI_SSID, WIFI_PASSWORD, move || {
127         let temp = temperate_sensor_read();
128         log::log(Level::INFO, &TAG, format_args!("Ticked: {}", temp));
129         if let Err(err) = publish_telemetry(&mut http, temp) {
130             log::log(
131                 Level::ERROR,
132                 &TAG,
133                 format_args!("Failed to execute HTTP upload: {}", err),
134             );
135         }
136     });
137
138     app.run();
139 }
```



End-to-end example



```
90 #[derive(Serialize, Deserialize, Debug)]
91 pub struct TelemetryPayload {
92     temp: f32,
93 }
94
95 fn publish_telemetry(client: &mut http::HttpClient, temp: f32) -> http::Result<http::Response> {
96     let payload = TelemetryPayload { temp: temp };
97     client.send_json::<heapless::consts::U128, TelemetryPayload>(&payload)
98 }
99
```

```
http::HttpClient, temp: f32) -> http::Result<http::Response> {
    { temp: temp };
    consts::U128
```

⚠ redundant field names in struct initialization

help: for further information visit https://rust-lang.github.io/rust-clippy/master/index.html#redundant_field_names
help: replace it with: `temp`

🔗 [Change to `temp`](#)

```
temp: f32
```

IDE Integration

```
1
2
3 fn default_value() -> String {
4     ... String::from("foo")
5 }
6
7 fn main() {
8     ... let s = Some(String::from("foo"));
9     ... let s = s.unwrap_or(default_value());
10 }
11
```

⚠ use of `unwrap_or` followed by a function call

note: `#[warn(clippy::or_fun_call)]` on by default

help: for further information visit https://rust-lang.github.io/rust-clippy/master/index.html#or_fun_call

help: try this: `unwrap_or_else(default_value)`

➡ [Change to `unwrap_or_else\(default_value\)`](#)

```
fn default_value() -> String
```



Eclipse Corrosion

Eclipse Corrosion: Rust edition and debug



☆ 29 💬 1

📄 Install



Details

Screenshots

Metrics

Errors

External Install Button

Corrosion enables **Rust** application development in the Eclipse IDE.

!!! A standalone [Eclipse IDE for Rust Developers](#) is also available for download !!!

Corrosion provides a **rich and smart Rust editor** with: - Syntax highlighting (using TextMate grammar) and Error reporting, Hover. Content assist. Jump to references, Code Outline, Formatting... provided by the Rust Language Server

Corrosion also integrates various operations of the `~cargo~` command-line (**New Project, Build, Run, Debug, Package**) as typical Eclipse IDE wizards and workflows.

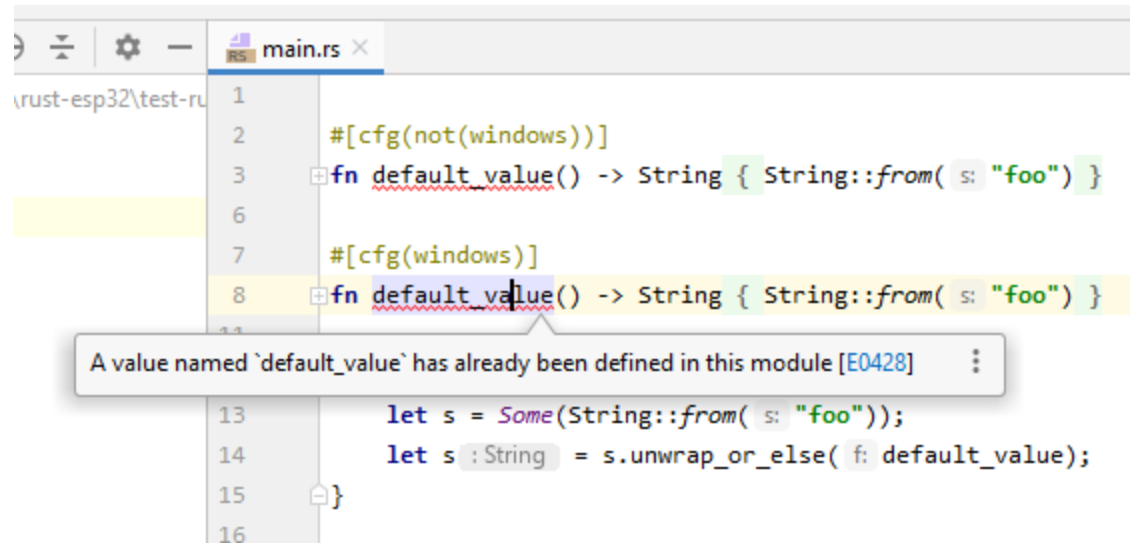
Corrosion contains a **rich debugger** for Rust applications, allowing to set breakpoints, jump in/over an instruction, view and edit structured variables...

Categories: Editor, IDE, Languages

Tags: rust, rustlang, cargo, fileExtension_rs, IDE, redox, corrosion



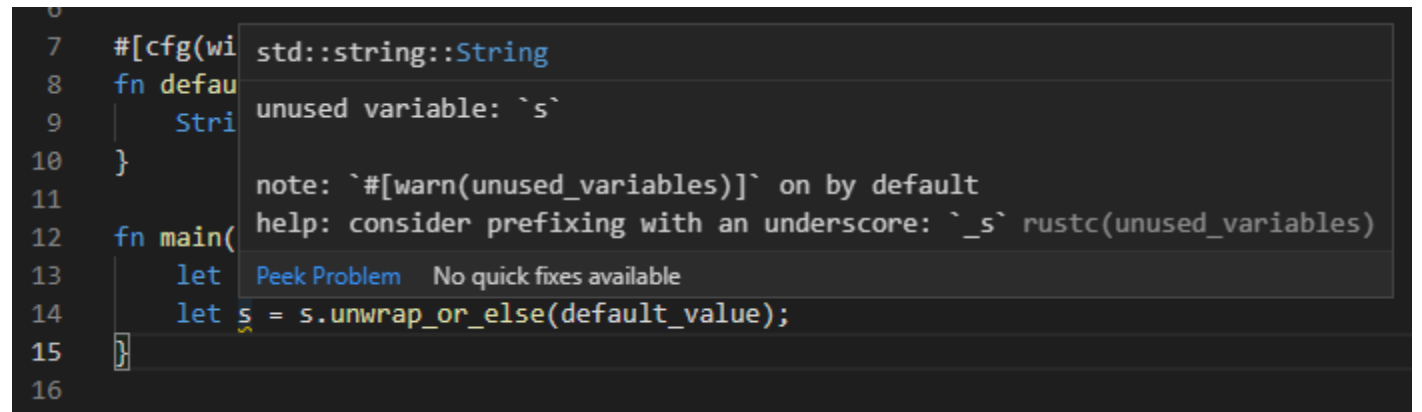
IntelliJ + Rust



```
rust-esp32\test-ru
1
2 #[cfg(not(windows))]
3 fn default_value() -> String { String::from( s: "foo") }
6
7 #[cfg(windows)]
8 fn default_value() -> String { String::from( s: "foo") }
11
13 let s = Some(String::from( s: "foo"));
14 let s :String = s.unwrap_or_else( f: default_value);
15 }
16
```

A value named `default_value` has already been defined in this module [E0428]

```
1
2 #[cfg(not(windows))]
3 fn default_value() -> String {
4     ...String::from("foo")
5 }
6
7 #[cfg(windows)]
8 fn default_value() -> String {
9     ...String::from("foo")
10 }
11
12 fn main() {
13     ...let s = Some(String::from("foo"));
14     ...let s = s.unwrap_or_else(default_value);
15 }
```

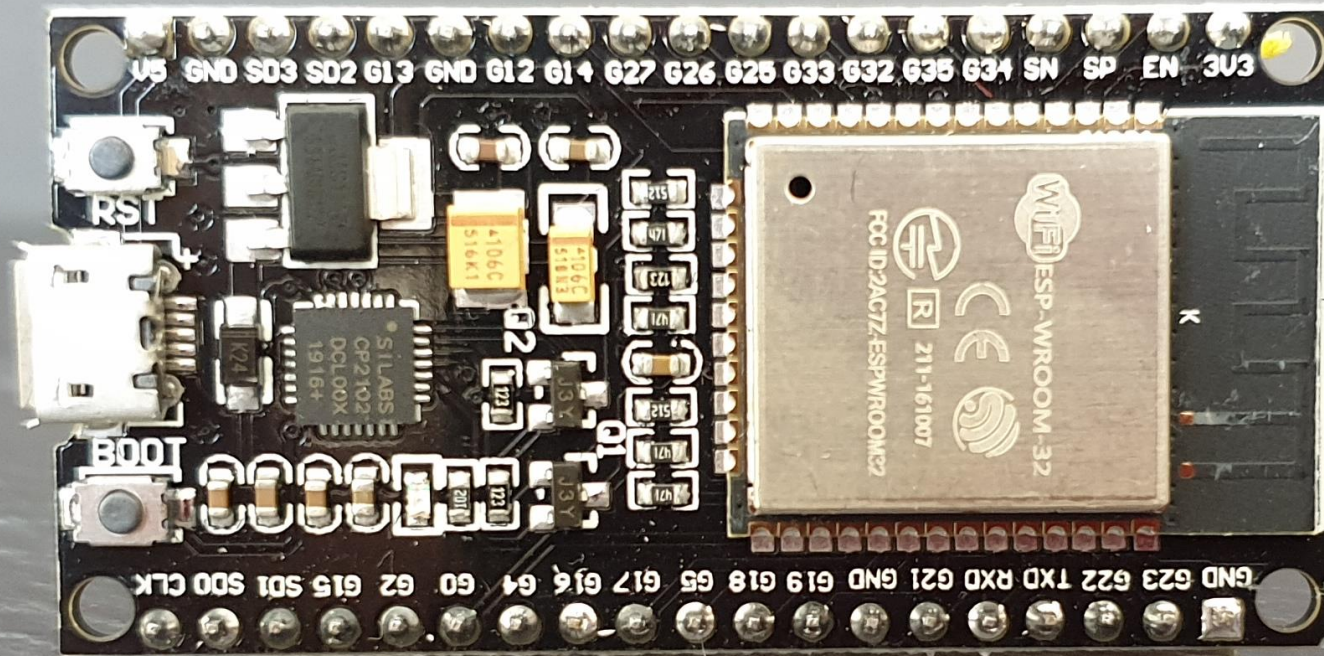


```
0
7 #[cfg(wi std::string::String
8 fn defau unused variable: `s`
9     Stri
10 }
11 note: `#[warn(unused_variables)]` on by default
12 help: consider prefixing with an underscore: `_s` rustc(unused_variables)
13 fn main( Peek Problem No quick fixes available
14     let s = s.unwrap_or_else(default_value);
15 }
16
```

One more thing...



How to compile for the Xtensa architecture?



Forked LLVM, forked rustc and a bunch of scripts

- Execute ~50 different commands
- Hope you have the right OS, in the right version, with the right packages
- Wait ~3½ hours
- Enjoy ... or try again

Containerized

- `docker run quay.io/ctrn/rust-esp`
 - `-ti -v $PWD:/home/project`
- Runs on Windows, Linux, (and should on Mac OS)

GV

What lies ahead?

- Rust Runtime for AWS Lambda
 - <https://aws.amazon.com/blogs/opensource/rust-runtime-for-aws-lambda/>
- Rust Embedded Book
 - <https://rust-embedded.github.io/book/>
- Linux kernel experiments with Rust
 - <https://lwn.net/Articles/797828/>
- Microsoft
 - <https://msrc-blog.microsoft.com/2019/07/16/a-proactive-approach-to-more-secure-code/>
 - „~70% of the vulnerabilities Microsoft assigns a CVE each year continue to be memory safety issues“



Questions?

... and answers!



A few links

- Rust
 - <https://www.rust-lang.org/>
- Rust Embedded Book
 - <https://rust-embedded.github.io/book/>
- Programming Rust
 - O'Reilly Media
- Eclipse Corrosion
 - <https://marketplace.eclipse.org/content/corrosion-rust-edition-eclipse-ide>
- Rust for ESP32
 - <https://github.com/ctron/rust-esp-container/>
- Rust, ESP32, ESP-IDF, Hono
 - <https://github.com/ctron/rust-esp32-hono>
- LLVM for Xtensa
 - <https://github.com/espressif/llvm-xtensa>
- Rust fork for Xtensa
 - <https://github.com/MabezDev/rust-xtensa>



Thank you!

