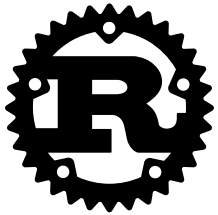
TencentOS Tiny meets Rust

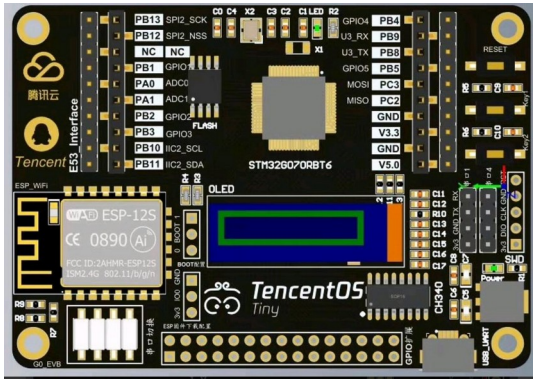


zhqli@tencent.com

Overview

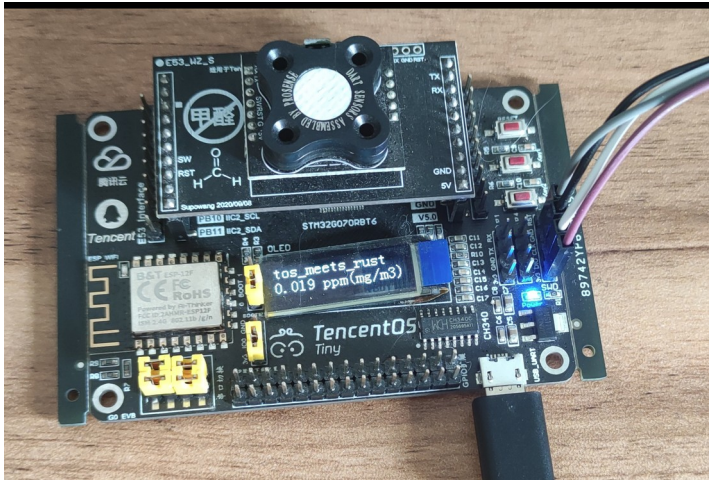
- Background
- Environment Setup
- Rust Integrated with tos
- Future
- References

Background



【源来很好玩01期】手把手带你打造智能家居环境监测终端 有奖 已结束

活动日期：2020年8月27日 00:00 ~ 2020年9月10日 00:00



Background

```rust
fn sum_odd_numbers(n: u64) -> u64 {
    let mut acc = 0;
    for element in 0.. {
        if element ≥ n {
            break;
        }
        if element.is_odd() {
            acc += element;
        }
    }
    acc
}
```

```rust
fn sum_odd_numbers(n: u64) -> u64 {
    (0..)
        .take_while(|element| element < &n)
        .filter(|n| n.is_odd())
        .fold(0, |sum, element| sum + element)
}
```

C is good, but Rust is better.

Feature	C	Rust
Hardware access	Y	Y
Pointers	Y	Y
Link with C binary	Y	Y
Garbage collection	N	N
Zero Cost Abstraction	Y	Y
LLVM Backend	Y	Y
Standard Library	Y	Y
Memory safety	N	Y

Feature	C	Rust
Type inference	N	Y
Parametric types	N	Y
Mono-morphisation	N	Y
Enumerated types	N	Y
Pattern matching	N	Y
Build and packaging	N	Y
Macros	N	Y
Closures	N	Y

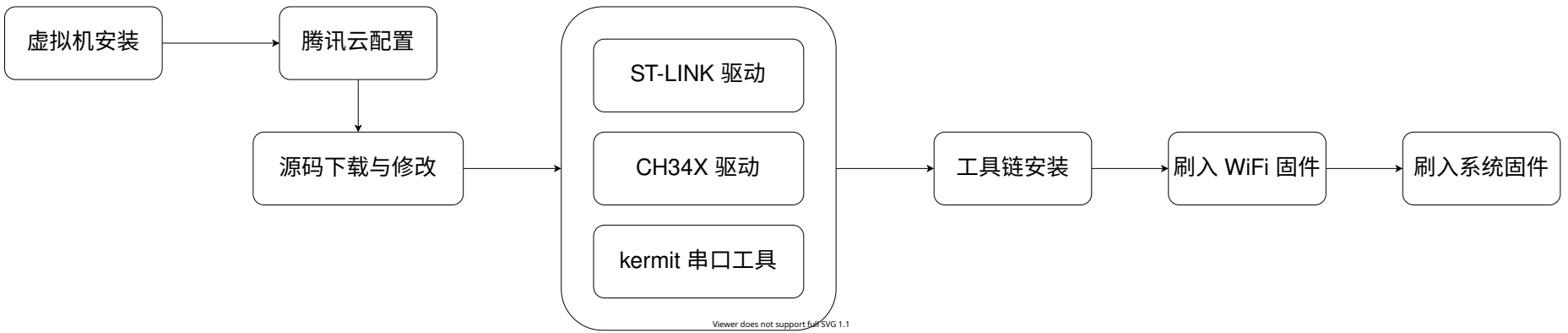
C vs Rust (source)

```rust
let _a = "hello";

let mut vec = Vec::new();
vec.push(5u8);

let _x: Vec<_> = (0..10).collect();
```

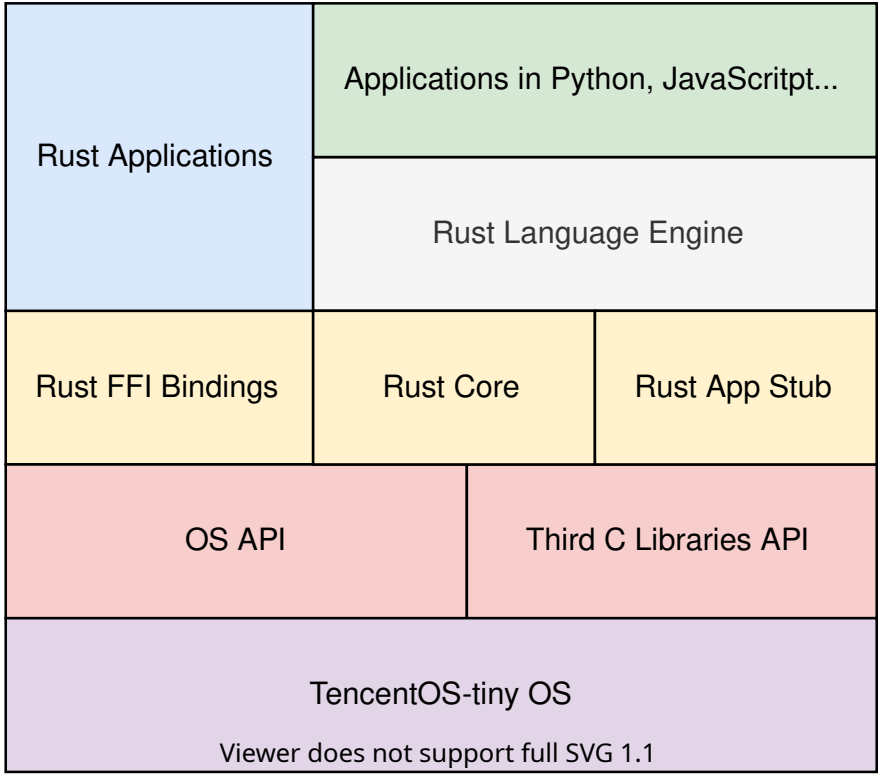
```rust
#[derive(Debug)]
pub enum WikitError {
    Info(String),
    Io(std::io::Error),
    Nom(String),
    ParseInt(std::num::ParseIntError),
    Utf8(std::string::FromUtf8Error),
    Utf16(std::string::FromUtf16Error),
    Regex(nom::regex::Error),
    TryFromSliceError(std::array::TryFromSliceError),
    Sqlx(sqlx::Error),
    Rocket(rocket::error::Error),
}
```

```rust
let p = Point { x: 0, y: 7 };
match p {
    Point { x, y: 0 } => println!("On the x axis at {}", x),
    Point { x: 0, y } => println!("On the y axis at {}", y),
    Point { x, y } => println!("On neither axis: ({}, {})", x, y),
}
```

Environment Setup

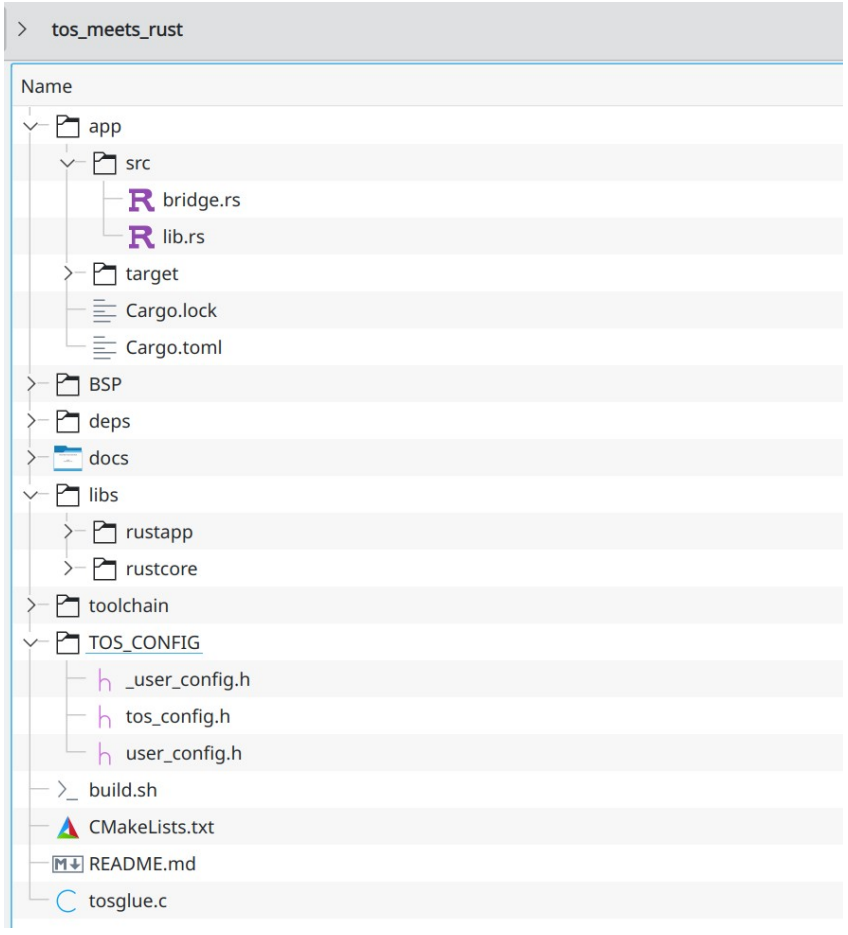


Rust Integrated with tos



Rust 集成架构图

Rust Integrated with tos



Future

- Python or Javascript interpreter
- WiFi module with RUST
- Any more?

References

- The Pain Of Real Linear Types in Rust: https://gankra.github.io/blah/linear-rust/

  介绍了什么是 liner type 和 affine type

- An introduction to Data Oriented Design with Rust: http://jamesmcm.github.io/blog/2020/07/25/intro-dod/#en

  讲解了许多深层的语言设计问题, 比如 Dynamic Dispatch vs. Monomorphisation

- Macros in Rust: A tutorial with examples: https://blog.logrocket.com/macros-in-rust-a-tutorial-with-examples/

  详细说明了 Rust 中的声明宏和过程宏

- Why you should learn the Rust programming language: https://developer.ibm.com/technologies/web-development/articles/os-developers-know-rust/

  讲了一下 Rust 语言起源

- Rust Influences: https://doc.rust-lang.org/stable/reference/influences.html

  Rust 语言灵感来源

- Rust over C: https://prataprc.github.io/rust-over-c.html

  Rust 和 C 的对比

- CS 242: Programming Languages, Fall 2019: https://stanford-cs242.github.io/f19/

  斯坦福的编程语言课程

References

- https://carette.xyz/posts/zero_cost_abstraction/：讲述了什么是零抽象能力