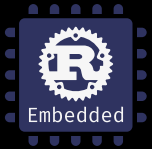
Using Rust on RIOT OS

Christian M. Amsüss <ca@etonomy.org>

2019-09-05
RIOT Summit, Helsinki

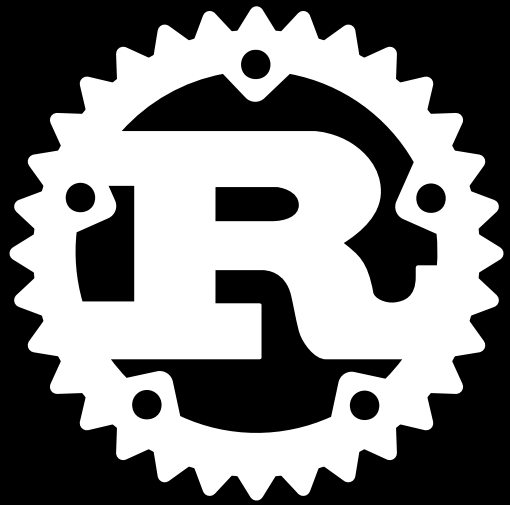


etonomy

RIOT

Outline

- The Rust language and ecosystem
- Rust on embedded systems
- Rust on RIOT



Simple things: functions and variables

```
fn fibonacci(n: u32) -> u32 {
    let mut a = 1;
    let mut b = 1;

    for _i in 2..n {
        let old_b = b;

        b = a + b;
        a = old_b;
    }

    b
}
```

Structs

```
struct Color {
    r: u8,
    g: u8,
    b: u8
}

let red = Color { r: 255, g: 0, b: 0 };
let h = red.get_hue();
```

Enums

```
enum ThreadStatus {
    Stopped,
    Sleeping,
    MutexBlocked,
    ReceiveBlocked,
    // ...
}

enum MaybeThreadStatus {
    Ok(ThreadStatus),
    NotFound
}
```

Ownership, Pointers & References: Ideas

- `x` / `mut x`: owned value

Ownership, Pointers & References: Ideas

- `x` / `mut x`: owned value
- `&mut x`: mutable reference
- `&x`: immutable reference

Ownership, Pointers & References: Ideas

- `x` / `mut x`: owned value
- `&mut x`: mutable reference
- `&x`: immutable reference
- `*const x` / `*mut x`: raw pointers

Ownership, Pointers & References: Example

```
fn receive() -> SharedPktsnip { ... }
fn read_data_from(r: &SharedPktsnip) -> &InternalData { ... }
fn process(input: &InternalData) -> Response { ... }
fn start_write(s: SharedPktsnip) -> WritablePktsnip { ... }
fn set_message(w: &mut WritablePktsnip, data: Response) { ... }

let received: SharedPktsnip = receive();
let data = read_data_from(&received);
let response = process(data);
let mut b = start_write(received);
set_message(&mut b, 42);
```

Ownership, Pointers & References: unsafe

```
let ptr = 0x40084000 as *mut u32;
unsafe {
  *ptr = 42;
}
```

. . . but you're on your own to know that is safe

Traits

```
trait OutputPin {
    fn set_low(&mut self);
    fn set_high(&mut self);
}

impl OutputPin for MyPeriph {
    fn set_low(&mut self) {
        self.write_command(SET_LEVEL, 0);
    }

    fn set_high(&mut self) {
        self.write_command(SET_LEVEL, 1);
    }
}
```

Generics

```
enum Option<T> {
    Some(T),
    None
}

struct SoftDebouncedButton<P: InputPin> {
    pin: P,
    laststate: bool,
    statecount: u16,
}
```

Et cetera ad infinitum

- Namespacing
- Visibility
- Closures
- Arrays and slices
- Standard library
- Unicode strings
- Futures / asynchronous programming

- Stable releases
- Cargo and the Crates
- The Rust Book

embedded-hal, *-hal, *-pac

```rust
trait OutputPin {
    fn set_low(&mut self);
    fn set_high(&mut self);
}
```

- embedded-hal

embedded-hal, *-hal, *-pac

```
impl OutputPin for A10 {
    fn set_low(&mut self) {
        self.reg.PA_doutclr.write(|w|
            w.p10().bit(true) });
    }

    fn set_high(&mut self) {
        self.reg.PA_doutset.write(|w|
            w.p10().bit(true) });
    }
}
```

- embedded-hal
- various HAL implementations

embedded-hal, *-hal, *-pac

```
...
<register>
  <name>PA_DOUT</name>
  <description>Port Data Out
      Set Register</description>
  <addressOffset>0x00C</addressOffset>
  <size>32</size>
  <access>read-write</access>
  <fields>
    <field>
      <name>P0</name>
      ...
```

- embedded-hal
- various HAL implementations
- various Peripheral Access Crates (PACs)

embedded-hal, *-hal, *-pac

```
struct RiotOutputPin {
    pin: gpio_t
}

impl OutputPin for RiotOutputPin {
    fn set_low(&mut self) {
        unsafe {
            gpio_set(self.pin);
        }
    }
}
```

- embedded-hal
- various HAL implementations
- various Peripheral Access Crates (PACs)

- riot-wrappers
- riot-sys

Rust on RIOT in practice: Implementation status

- I^2C
- SPI
- Threads
- Gcoap server
- shell
- GNRC (Pktsnip)

+ direct hardware access

Rust on RIOT in practice: Build system integration

```
$ make BOARD=stk3700 all flash term
make -C ./RIOT/boards/stk3700
...
make -C ./RIOT/cpu/cortexm_common/periph
RIOT_CFLAGS="-D... -I ..." cargo build --target arm-... --release
Compiling riot-sys v0.2.2
...
Compiling demo v0.1.0
    Finished release [optimized + debuginfo] target(s) in 1m 09s
arm-none-eabi-gcc .../*.o target/.../libdemo.a -o bin/demo.elf
### Flashing Target ###

main(): This is RIOT! (Version: ...)
Hello, world!
>
```

Rust on RIOT in practice

- in active use
- benefits on bug prevention and code reuse

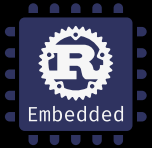
Future exploration

- More complete mappings
- Rust code in packages
- Const propagation and bindgen enhancements

Questions?

Thanks for your attention

Slides and more links on
`https://christian.amsuess.com/presentations/2019/rust-on-riot/`



etonomy

RIOT