Arduino Software Development in Rust

Author: Bc. Michal Žatečka
Supervisor: RNDr. Marek Rychlý Ph.D.

Why do we need that?

- Rust offers memory safety features that C/C++ doesn't have
- Hard to configure manually
- A lot of steps in the building process
- No existing tool that would automate the process
- Low abstraction cost

Example Rust code

```
#![no_std]
#![no_main]

use panic_halt as _;

#[arduino_hal::entry]
fn main() -> ! {
    let dp = arduino_hal::Peripherals::take().unwrap();
    let pins = arduino_hal::pins!(dp);

    let mut led = pins.d13.into_output();
    loop {
        led.toggle();
        arduino_hal::delay_ms(1000);
    }
}
```

Tool architecture

- Support for only Rust or Rust with C/C++ libraries applications
- Resolving dependencies
- Automation from project initialization to uploading the binary
- CLI tool and the Visual Studio Code extension work as a thin client
- Cargo ensures building
- PlatformIO provides necessary tools and configuration data

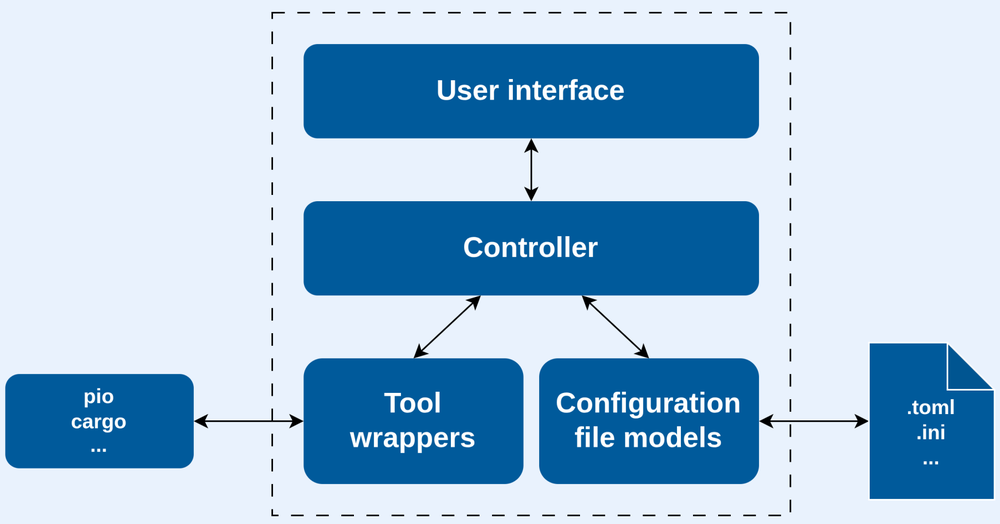

Figure 1: Architecture of the CLI tool.

Build pipeline

- C/C++ library linking only when it is needed (hybrid mode)

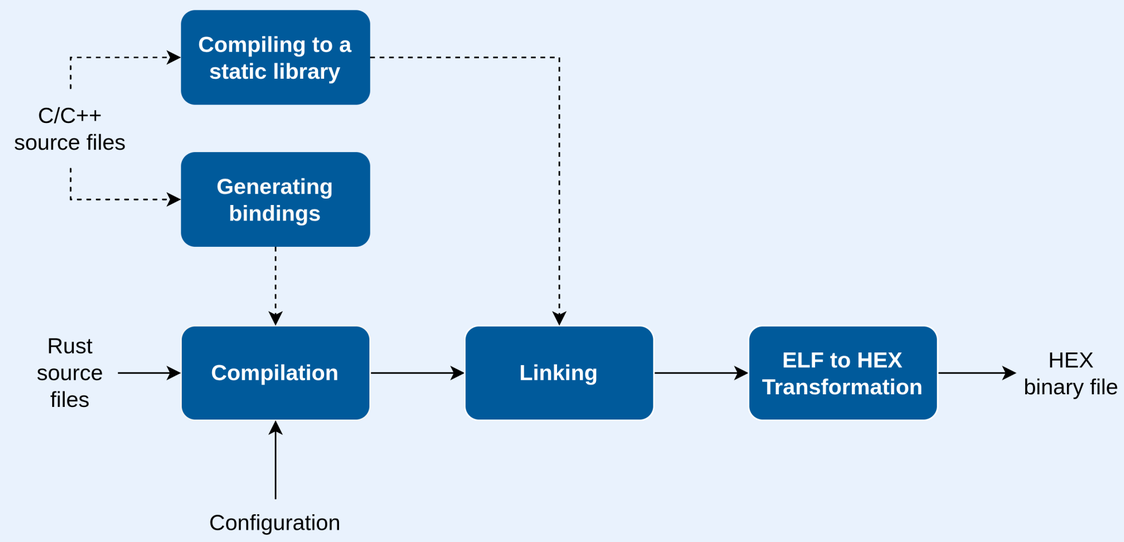

Figure 2: Rust or Rust with C/C++ embedded application build pipeline.

Binary size

Tested on 3 applications:

- Simple blinking LED
- Configuring the color of the RGB LED using an analog input
- A monitor of temperature and humidity that sends data using UART

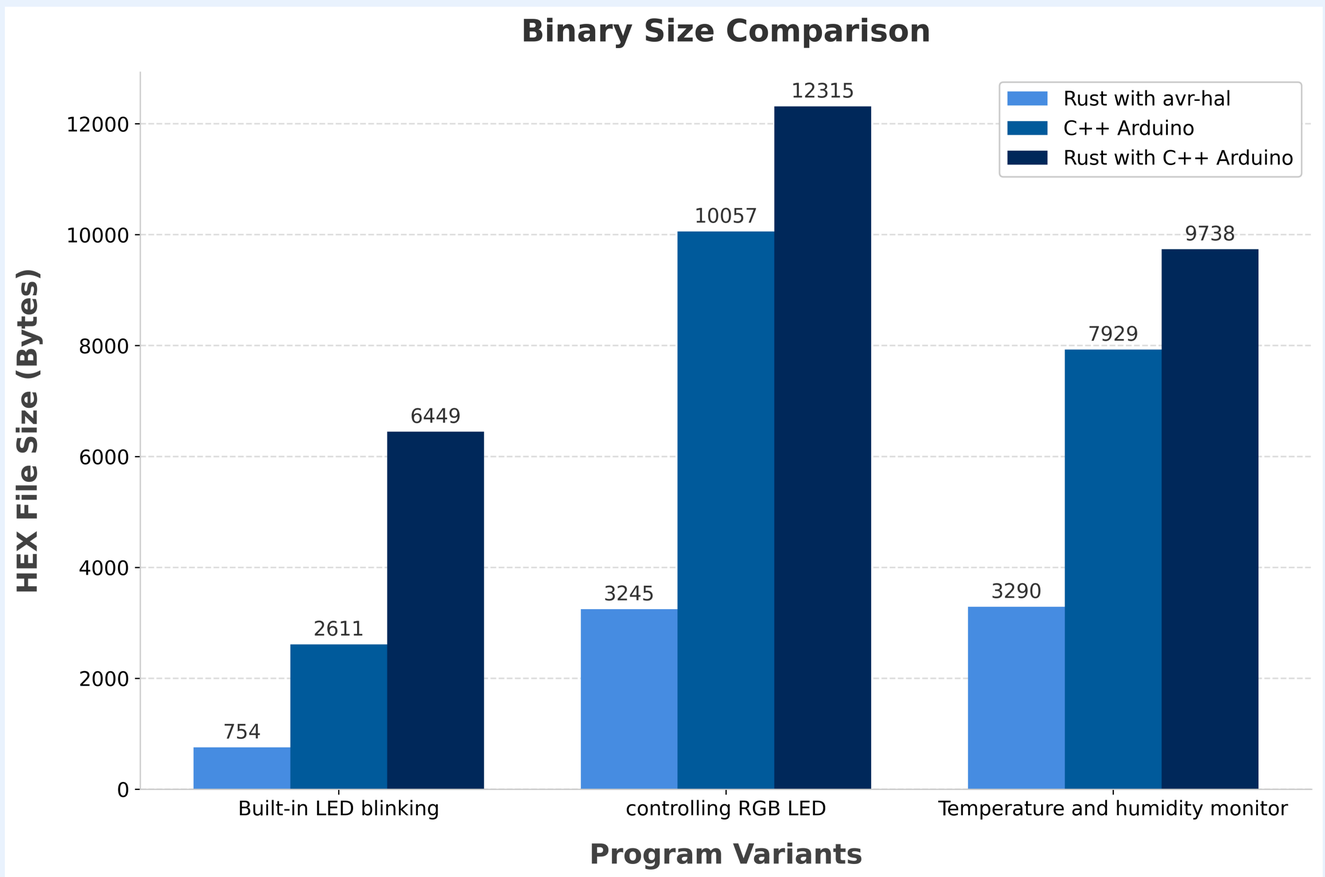

Figure 3: Comparison of binary file sizes.

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY
xzatec02@stud.fit.vut.cz

Excel@FIT 2026