Embedded Rust

Intro and Ecosystem Overview

Levi Pearson

Overview

Background

I am:

- A very experienced embedded systems programmer
- A moderately experienced Rustacean
- Pretty new at the combination

Questions to answer

- What are embedded systems?
- What makes Rust interesting in that context?
- What resources are there right now?
- How do I get started?

Embedded Systems?

A (very general) definition

When a device has a computerized, software-controlled component as part of its mechanism, which supports its primary function rather than the computer being *its primary function, that component constitutes an embedded system.*

Examples

- A digital scale's control system
- The engine management computer in a car
- The disk controller in a hard drive
- A factory process control computer

They are everywhere

- Microcontrollers start in the ~$0.01 price range
- They can be as small as a grain of rice
- They are in credit cards and SIM cards
- They are also often off-the-shelf Windows boxes

Microcontrollers

- Single-chip computers specialized for embedded control
- Much wider variety of architecture
- Integrated peripherals:
  - Memory (RAM/Flash)
  - Timers
  - Sensors
  - Communication bus interfaces
  - Display controllers

Why Rust?

The landscape today

- Most embedded systems are still done in C
- C was a big step up from assembly
- C is still very error-prone
- We are putting more software in more things

Why not other safe languages?

- Real-time response requirements
- Resource (often RAM/Flash) constraints
- Memory layout and representation control
- Perception (no one thinks of Ada)

Big wins

- The big enabler: `#[no_std]` and powerful, alloc-free `core` library
- Static resource analysis via ownership and hiding
- Modularity without overhead via traits
- Flexible, easy-to-use builds including cross-compiling with cargo
- Industrial strength multi-arch compiler back-end via LLVM

Rusty resources

Compiler and tools

- `rustup target list | grep none` for no-OS embedded targets
- `cargo install cargo-binutils` for `cargo size`, `cargo nm`, etc.
- `rustup component add llvm-tools-preview` for LLVM-native binutils
- gdb built for the target arch for debugging
- `openocd` to drive the programmer/debugger module

Embedded-wg

- Embedded Rust Book
- Awesome / Not-Yet-Awesome Lists
- `svd2rust` and Peripheral Access Crates
- `embedded-hal` and HAL Crates
- Board Support Crates
- `rtfm` “Real-Time For The Masses” framework

Levels of maturity

- Compiler back-ends for ARM and some other architectures are solid
- Much work is now possible with stable Rust
- Some patterns for safety are well-defined, others are still experimental
- Patterns for modular, reusable code are still in progress
- Ferrous Systems - works with **embedded-wg** and provides training/support

Where to start?

Emulation!

- QEMU emulates a pair of ARM Cortex-M3 evaluation boards
- The `ml3s6965` board has a minimal peripheral support crate
- Both the Embedded Book and RTFM Book walk through starting with it

Development boards

- Vendor-developed boards can be expensive, but have nice components
- Many hobbyist-oriented boards are available, usually cheaper!
- Cheap knock-offs of open-hardware boards are *really* cheap
- Stay away from AVR-architecture Arduino-style boards for now!
- Risc-V and MSP-430 have llvm support, but not as much community support

Sensors, actuators, etc.

- Many boards have LEDs, buttons, capacitive touch, etc.
- Adafruit, Sparkfun, Pololu have interesting, well-designed break-outs
- Lots of cheap stuff from China, of varying quality!
- Salvage from old electronics junk

Time left?

Discussion/Demos

- General or rust-specific embedded Q&A
- Walk-through tool + qemu install and first program
- Look at some example code
- Interest in future workshop w/real hardware?