Using Rust in MCUboot

David Brown

Linaro Connect VIRTUAL 2020

Rust

- New programming language
  - By Graydon Hoare (and others) at Mozilla
  - 5 years ish initial development, 5 years since 2015 1.0 release
- Emphasis on memory safety
  - No SEGFAULT without using “unsafe”
  - Strong, compiler-enforced concept of ownership and borrowing

MCUboot

- Original bootloader for MyNewt
- Linaro (me) ported to Zephyr, now multi-platform
- Written in C (there is also support code in Python and Go)

What MCUboot Does

- Cryptographically strong root of trust
- Validate signatures of image before running it
- Validates signatures, verifies dependencies before installing updates
- One common update method is “swap”
  - Exchanges images between two slots
  - Attempts to recover from power loss at any time
  - Currently two implementations, soon a third
- There are other methods as well, such as an overwrite, direct XIP, and loading to RAM
- This upgrade is important to get right
  - Modifies persistent state (flash)
  - MCUboot itself is difficult/impossible to upgrade

How we are using Rust

- Simulator, to test validation and upgrade process
- Wrote the flash simulator in Rust, as well as the testing framework
    - Rust has a testing framework, which matched well
    - Safety, less worry about bugs in simulator
- Calling to C
    - A little bit of a mismatch, but not too bad
    - #[repr(C)] allows struct to match C conventions
    - Callbacks are unsafe, but carefully contained

Issues using Rust

- Testing is multi-threaded
  - MCUboot had lots of global state, could only be run once
  - Initially, we just used a mutex to make sure only one test ran at a time
  - Eventually, we moved the state into a struct. Embedded version uses a static struct, testing version is passed in
  - Overall, this made the MCUboot code better
- Not too many know Rust
  - Some contributions come in without simulator changes to match
  - Simulator testing of the features usually has to be done by core team

Future of Rust and MCUboot

- Embedded Rust. Still early, not many boards supported.
- One idea would be a Rust app, with a binding to the Zephyr API
- Numerous advantages to bootloader itself if written in Rust:
    - Memory safety
    - Easier to reason about code
    - “unsafety” more carefully managed
    - Growing popularity
- Not just “rewrite it in Rust”, but MCUboot isn't too large a body of code, and a really good fit (other than the embedded issue).

Thank you

Accelerating deployment in the Arm Ecosystem

Linaro Connect VIRTUAL 2020