Satellite Software Development Framework with Rust that Improves Developer Enablement

SSC23-P5-24

Yuta Sakamoto, Hidekazu Kobayashi, Ryo Suzumoto, Toshifumi Akima,
Shumon Fujita, Masato Hoshika, Yoshiki Iwasa, Kanta Yanagida

ArkEdgeSpace

sksat@arkedgespace.com

Our challenge: developing various satellites with a small team in a short period

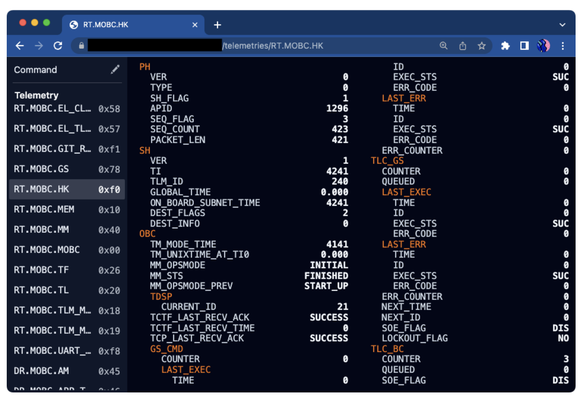
C-language: not able to scale	Rust ecosystem: high productivity
No standard toolchain • Difficult to setup a reproducible environment in all environments	**rustup: Rust toolchain installer** • Install every Rust toolchain with one command • Also install standard formatter, linter, etc
No standard build-system **No common package-manager** • Almost the only way to reuse libraries is to simply copy them	**Cargo: Rust package manager** • Integrated build-system • Custom build-script (It can be share build logic as package)
No package-registry • Downloading methods have to be provided (e.g., Git-submodule)	**crates.io** • Many useful packages: serde, embedded-hal, zerocopy, etc.
Low reusability of libraries • We are using C2A[1], as an existing FSW (flight software): written in C • Lack of language features: prevents proper partitioning of libraries according to responsibilities	**Many support for reuse implementation** • Some no_std crates can use in bare-metal environment • Language features: module system, strong type-system, etc

Rust ecosystem reinforces C2A

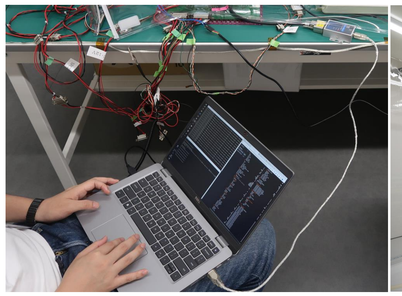
- **Integrating all build tools with Cargo**
- **Packaging C library as Rust library (We can reuse it by writing 1-line in Cargo.toml)**
- **Automatically generated glue code (bindgen)**
- **High functionality SILS (Software In The Loop Simulation) environment**

```
# Install whole toolchain
❯ curl -sSf https://sh.rustup.rs | sh
info: downloading installer
...(collapsed)...
# Clone your FSW repository
❯ git clone https://github.com/arkedge/c2a-sam
# Run FSW
❯ cd c2a-sample; cargo run
-- C2A SAMPLE Flight S/W (H-ON, F-ON) --
BUILD: Jul 11 2023 20:03:57
Git rev: CORE 0x772e7ca, USER 0x4148014
CYCLE: TOTAL 00000021, MODE 00000021
MODE: STAT 1, PREV 0, CURR 1
CMD: GS   0, RT   0, Ack  0, ID 0x00, Sts 0, E
```

Starting FSW development instantly
SILS runtime enables run actual FSW on your PC



C2A DevTools
Web-based human friendly telecommand/telemetry interface



Testing the OBC using C2A DevTools

Assembling our satellite

Conclusion

We have been challenged to develop a variety of satellites in a short period of time with a small team. However, our development structure based on C2A (our C language asset) was not productive enough: there are no common toolchain, difficulty to reuse code for similar satellites/components, etc.

To solve these problems, we have introduced the Rust ecosystem into the C2A development structure. Rust's high degree of interoperability with C allows us to significantly improve productivity without throwing away existing assets.

As a result, our **developer enablement** has been unified and improved across the entire team, allowing us to develop software for multiple satellites with a small team in a short period. Even when developing software for our new satellites, we can get started immediately by adding a few lines to Cargo.toml.

References

[1] Ryo Suzumoto, *et al.*, Open-source Software Suite for Small Satellites: C2A (Command Centric Architecture), S2E (Spacecraft Simulation Environment), and WINGS (Web-based Interface Ground-station Software), 36th smallsat, 2022.
[2] Rust: https://www.rust-lang.org/
[3] rustup: https://rustup.rs/
[4] Cargo: https://github.com/rust-lang/cargo
[5] bindgen: https://github.com/rust-lang/rust-bindgen