My Open Source Story (so far!)

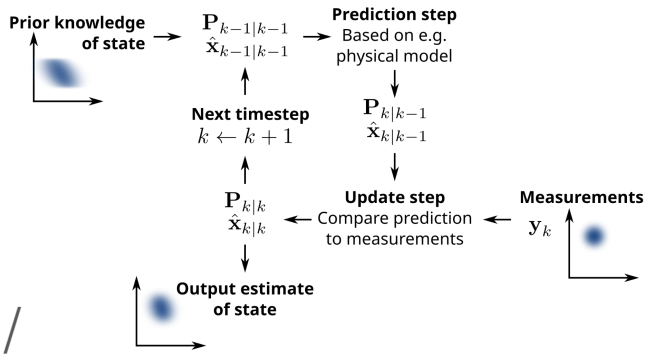
January 2025

Pete LeVasseur
Eclipse uProtocol Maintainer


Before OSS

In the before times…

- Software engineer in Automotive for ~10 years
- Primarily worked in the area of Automated Driving / Advanced Driver Assistance Systems (ADAS)
- Primarily worked in C, C++, some Python for scripting
- Became fairly frustrated with C and C++ around ~2016
  - Especially mentoring people new to the languages
- Found the Rust programming language in 2016
  - Brings memory-, thread-, and type-safety, making software written in it much more reliable
- Wrote a small demo using Rust for a side project at work in ~2017
- Wrote a bare-metal Kalman Filter library in Rust in ~2018



Kalman filter equations

Finding an open-source project to contribute to

- Project at GM I worked on for ~6 years got cancelled in late 2023 :(
- Find Eclipse uProtocol open source project getting staffed by GM folks :)
- There's an effort to write a Rust language library for uProtocol!

Eclipse uProtocol

- Site: https://eclipse-uprotocol.github.io/
- GitHub project: https://github.com/eclipse-uprotocol

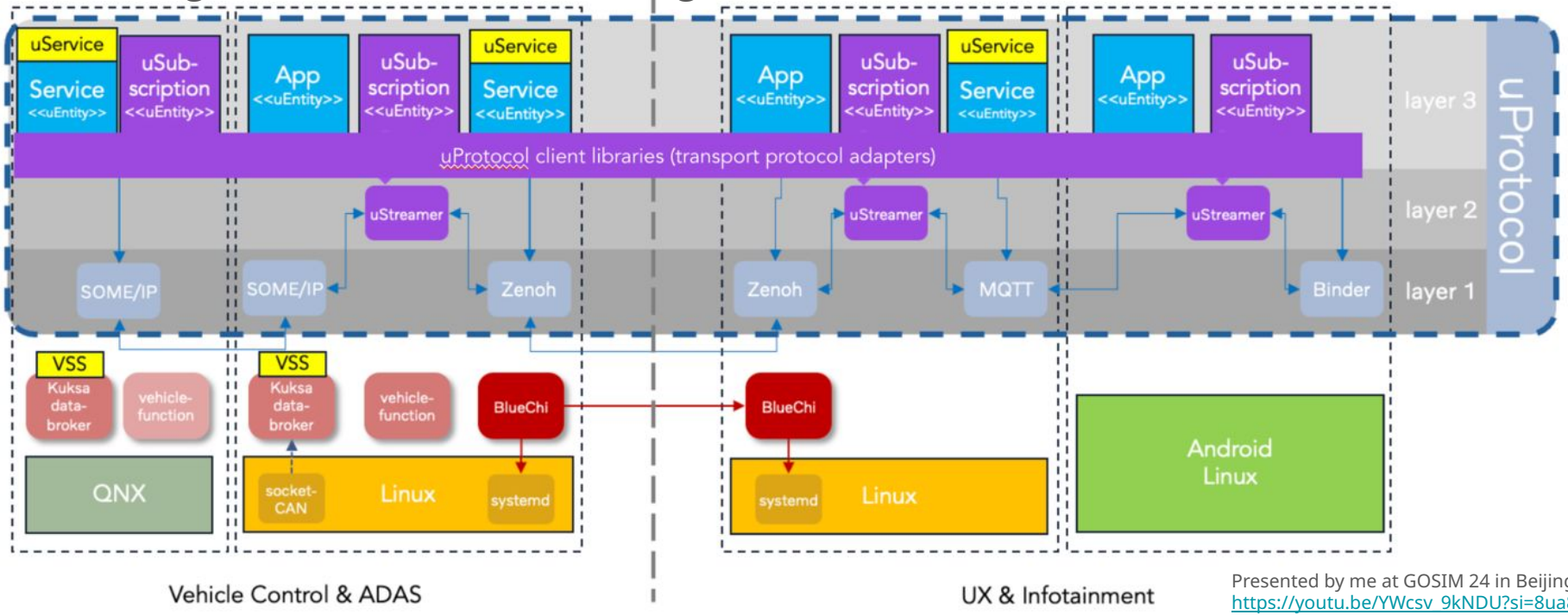
Pete's Open Source Timeline

I've been involved in OSS for ~1 year and been fortunate to contribute to several efforts!


uProtocol

Eclipse uProtocol – An SDV Service Mesh Framework

Key Takeaway: Eclipse uProtocol allows abstracting away underlying transport to let engineers focus on delivering customer value



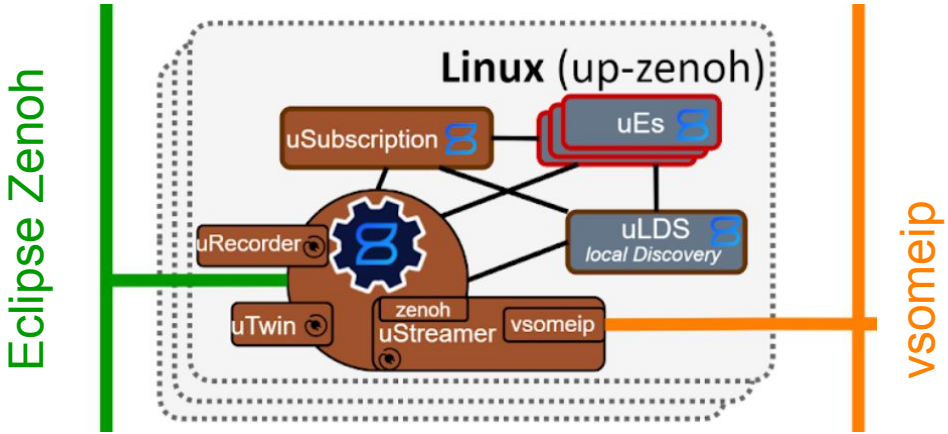
Presented by me at GOSIM 24 in Beijing: https://youtu.be/YWcsv_9kNDU?si=8ua827lXmLrnBYiO

Ramping up

- I spent 2-3 weeks reading over the specification (`up-spec`) to see if the Rust implementation abided by it
- I noticed a gap in the Rust language library implementation of uProtocol: `up-rust`
- Namely, there was not validation done on values when serializing the `UUri` to ensure that all the bits and bytes would fit into the on-the-wire format (Google Protocol Buffers)
- I wrote a first issue and pull request to address it
- It took a month for my first PR to get merged 😅
- I learned a lot though from this though!

`up-streamer-rust`: Concept

- Working with the project lead, we identified that the UStreamer component could have a proof-of-concept written in Rust
  - UStreamer: a transport-bridge, bridging between underlying transports Foo and Bar
- UStreamer is a stand-alone binary / Zenoh plugin so didn't need to interact with other modules written in C++, just reads config file



`up-streamer-rust`: Roadblock on `UTransport`

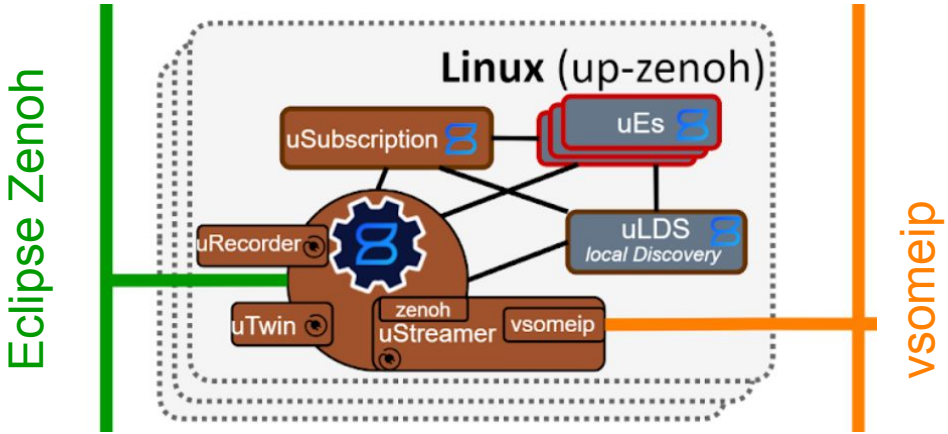
- Identified that the `UTransport` trait needed to be made thread-safe to support ease of use in implementing `up-streamer-rust`
  - Daniel Krippner put the PR through in `up-rust`
- Rationale: `UPStreamer` uses an async executor meaning that that calls to the async functions in `UTransport` would potentially be run on different threads each time

```rust
#[async_trait]
pub trait UTransport {
    async fn send(&self, message: UMessage) -> Result<(), UStatus>;
    async fn receive(&self, topic: UUri) -> Result<UMessage, UStatus>;
    async fn register_listener(
        &self,
        topic: UUri,
        listener: Arc<dyn UListener>,
    ) -> Result<(), UStatus>;
    async fn unregister_listener(
        &self,
        topic: UUri,
        listener: Arc<dyn UListener>,
    ) -> Result<(), UStatus>;
}
```

```rust
#[async_trait]
pub trait UTransport: Send + Sync {
    async fn send(&self, message: UMessage) -> Result<(), UStatus>;
    async fn receive(&self, topic: UUri) -> Result<UMessage, UStatus>;
    async fn register_listener(
        &self,
        topic: UUri,
        listener: Arc<dyn UListener>,
    ) -> Result<(), UStatus>;
    async fn unregister_listener(
        &self,
        topic: UUri,
        listener: Arc<dyn UListener>,
    ) -> Result<(), UStatus>;
}
```

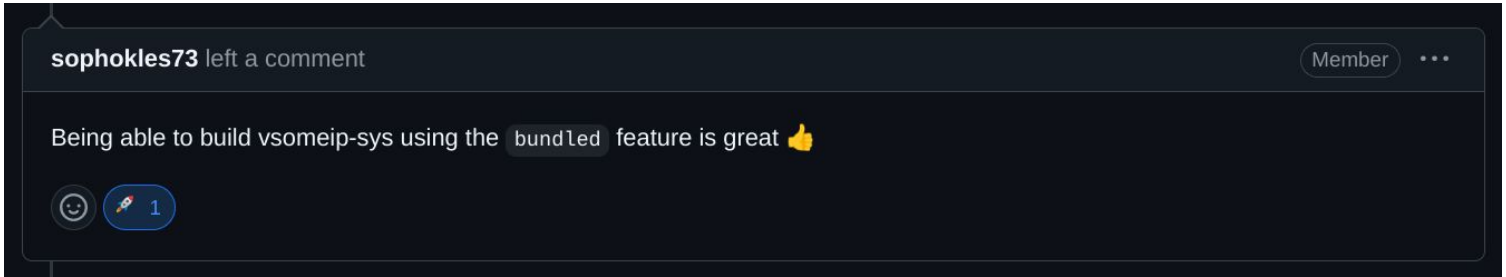
`up-transport-vsomeip-rust`: Necessity!

- An attempt made for approximately two months by other contributors to Eclipse uProtocol to write a Rust SOME/IP transport that did not succeed
- There existed a C++ UStreamer implementation that was not compliant with spec which would be used instead



`up-transport-vsomeip-rust`: Handling `unsafe` Rust

- I wrote a Rust wrapper around the C++ vsomeip library, making use of Rust's strong support for C++ interop (cxx, autocxx crates)
- Exposed the Rust vsomeip wrapper, `vsomeip-sys`, so that `up-transport-vsomeip-rust` could be written in a memory-safe way
- Unfortunately, no other contributors to Eclipse uProtocol knew how Rust's `unsafe` keyword worked nor had familiarity with Rust's FFI support
  - *Basically YOLO'ed all of `vsomeip-sys` in with no review!*
- Got some great feedback to push to improve to have both vsomeip bundled with and provided separately from the crate



Learning the open-source ropes

- You may get fairly direct feedback! This is totally normal
- Issues you open may have a fair bit of back and forth till resolution
- Issues/PRs may never get addressed at all even if you contribute. It happens
- Working code > theoretically perfect code that never arrives
- Generally the bar is *higher* on code quality and practices
  - The code is public! Not something sitting in an internal company repo
- Not top-down mandates, but rather ground-up percolation of ideas
- Automotive open-source still a bit new, OEMs and suppliers warming to the idea
  - Eclipse SDV putting energy into badges to show level of maturity of software practices for projects
  - Eclipse SCORE project aims make more OSS safety-qualifiable for automotive


Rust SIG

Eclipse SDV: Rust Special Interest Group

- Florian Gilcher kickstarted the Rust Special Interest Group based out of the Eclipse SDV
- Great chance to chat with and learn from someone that had been in the Rust space for over a decade
  - I’ve learned more from Florian about leadership and community building than Rust tho!
- I became co-lead of the SIG, so I now:
  - Formulate agendas, gathering input
  - Hold meetings, guide discussion
  - Coordinate and contribute to creation of artifacts


A New Road

Forcibly put on a new road!

- GM executed a 1000 person restructuring in August 2024, I was restructured
- Started a blog, writing about the adventures in Eclipse uProtocol
- Started a consulting firm for Rust: Oxidation Partners
- Began actively posting to social media about Rust and Eclipse uProtocol
- Decided together with wife that I would attempt to get another role where I can develop in Rust, ideally involved in the open-source community
- Made a gamble, paying out of pocket to go to RustConf in Montreal just a few weeks after being laid off
    - Booking so late I had to get a hotel room with three beds to be close to the event!

RustConf 2024 in Montreal

- Have dinner with Eclipse uProtocol project lead Steven Hartley
- Got invited to the Safety-Critical Rust Consortium
- Attend kickoff meeting of the Safety-Critical Rust Consortium
- Watch some great talks, meet and get to know some fellow Rustaceans
- Lead an Unconference session about Rust and Industry: Where they Meet
- Attend Zed Hackathon, get to hack on some Vim mode bugs in Zed
- Get invited to give a talk about Eclipse uProtocol in the Software Defined Vehicle track at GOSIM, an open source conference in Beijing in October

GOSIM 2024 in Beijing

- Present a talk about Eclipse uProtocol, its value proposition as a service mesh to abstract over underlying transports
- Meet folks doing cool work with Rust in plenty of different domains
- Run into one of the contributors, Allen Wyma, to the Rustacean Station Podcast, chat about the problems Eclipse uProtocol is trying to solve, and he invites me on the podcast.

Safety-Critical Rust Consortium: Coding Guidelines

- I kept asking questions about the Coding Guidelines Subcommittee, as I was very bothered by my earlier experience of no one being able to review my `unsafe` code in uProtocol. It was suggested I could chair the group!
- Was suggested by Florian Gilcher that we're lacking good, practical guidelines on `unsafe` in Rust, so this idea would be valuable to both safety-critical and non
- We were able to meet up with some contacts Florian had, finding out about the Learn unsafe Rust book as a place to contribute work.
- Run for a few months now, we're starting to break the work down of the different pieces of `unsafe` for folks to investigate.


Woven

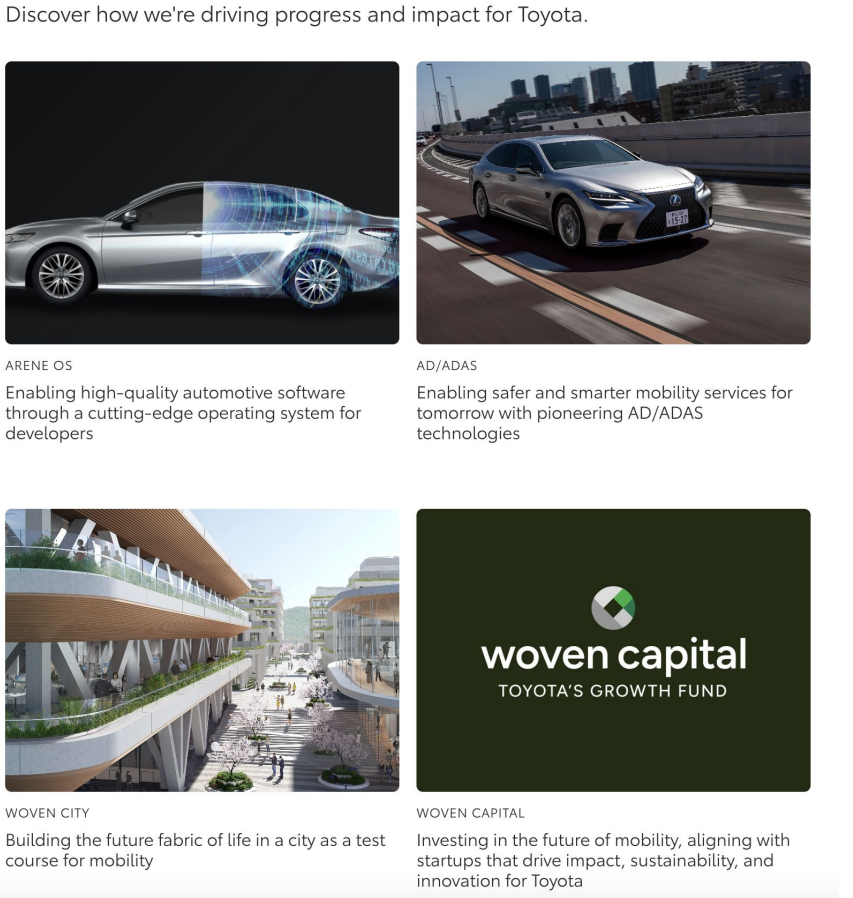
The Right Fit at Woven by Toyota

- From talking with folks from Woven by Toyota I met at RustConf and the following interviews it sounded like I checked a number of boxes: SDV, open-source, Rust
- Folks were aware of the work I was putting in within the open-source community for Rust, so I took this as a strong signal for me to continue my Rust advocacy at Woven
- *"Safety is our foremost priority in vehicle software development. Traditionally, achieving the highest levels of safety has been a complex and lengthy endeavor, requiring the use of specialized tools and processes beyond the programming language. We are therefore pleased to collaborate with leading experts in the safety industry to integrate new tools such as* ***Rust*** *into our safety-critical systems,"* said JF Bastien, Distinguished Engineer at Woven by Toyota.

Reference: https://foundation.rust-lang.org/news/announcing-the-safety-critical-rust-consortium/

Woven by Toyota: The Mission

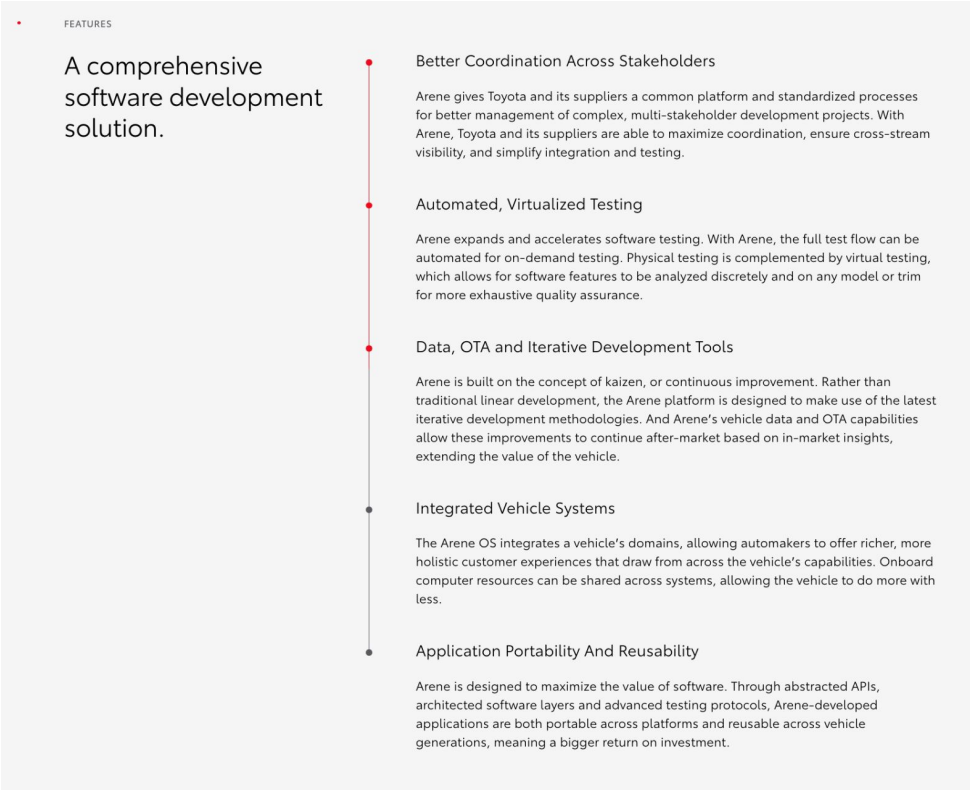
“Woven by Toyota will help Toyota to develop next-generation cars and to realize a mobility society in which everyone can move freely, happily and safely.”



Reference: https://woven.toyota/en/

Woven by Toyota: Arene OS

“The Arene OS platform draws on Toyota’s long history of automotive know-how and operational expertise. A combination of build tools and on-vehicle software services, Arene enables the development of integrated, software-driven experiences, all with the quality and efficiency that is synonymous with Toyota.”



FEATURES

A comprehensive software development solution.

Better Coordination Across Stakeholders

Arene gives Toyota and its suppliers a common platform and standardized processes for better management of complex, multi-stakeholder development projects. With Arene, Toyota and its suppliers are able to maximize coordination, ensure cross-stream visibility, and simplify integration and testing.

Automated, Virtualized Testing

Arene expands and accelerates software testing. With Arene, the full test flow can be automated for on-demand testing. Physical testing is complemented by virtual testing, which allows for software features to be analyzed discretely and on any model or trim for more exhaustive quality assurance.

Data, OTA and Iterative Development Tools

Arene is built on the concept of kaizen, or continuous improvement. Rather than traditional linear development, the Arene platform is designed to make use of the latest iterative development methodologies. And Arene’s vehicle data and OTA capabilities allow these improvements to continue after-market based on in-market insights, extending the value of the vehicle.

Integrated Vehicle Systems

The Arene OS integrates a vehicle’s domains, allowing automakers to offer richer, more holistic customer experiences that draw from across the vehicle’s capabilities. Onboard computer resources can be shared across systems, allowing the vehicle to do more with less.

Application Portability And Reusability

Arene is designed to maximize the value of software. Through abstracted APIs, architected software layers and advanced testing protocols, Arene-developed applications are both portable across platforms and reusable across vehicle generations, meaning a bigger return on investment.

Reference: https://woven.toyota/en/arene/

Conclusion

Open Source: Pursue passions, create awesome projects

- Do you have a programming language, a technology, a domain you find cool?
  - Jump in and learn more. Where are the gaps? Can you contribute docs? A test?
- Find people who care deeply about doing the right thing and pushing forward
  - It's possible in your day job you'd like to do things "the right way", but business objectives get in the way!
- Is open source right for everyone? Maybe, maybe not
  - Spam contributions! Drive-by PRs!
- Do open-source =/=> get a job
  - Can it lead to one? Sure! But I did Rust for several years just for fun before having a chance to use it professionally
- Bar is surprisingly low to contributing to many projects. Follow contribution guidelines and jump in!

Thank You!

JOIN US ON GITHUB!

Eclipse uProtocol GitHub Project:
https://github.com/eclipse-uprotocol

Eclipse SDV Blueprint: Service-to-Signal:
https://github.com/eclipse-sdv-blueprints/service-to-signal

Eclipse uProtocol Website:
https://eclipse-uprotocol.github.io

References

References

- Background image: https://www.flickr.com/photos/spodzone/33302660672/