Pasadena Nights
The Legend of Rust vs Zephyr
March 11, 2023

Frédéric Desbiens — Program Manager and Evangelist, IoT and Edge Computing
@BlueberryCoder
About Me

Frédéric Desbiens
Program Manager and Evangelist — IoT and Edge Computing

B.Ed., B.Sc.A, MBA

Developer, Architect, Product Manager...

Oracle, Cisco, Pivotal...

Published author; Frequent Speaker

@BlueberryCoder
https://ca.linkedin.com/in/fredericdesbiens
December 2022

A comprehensive overview of the open-source IoT and Edge Computing platforms available at the Eclipse Foundation

ISBN: 978-1484288818
Come See Us!

Booth 215
Agenda

01 Do We Need RTOSes?
02 Zephyr
03 Rust
04 SHOWDOWN!!!
Do We Need RTOSes?
Do We Need RTOSes?
What OSes provide

- APIs
- Hardware Support
- Memory Management
- Multitasking
- Real-Time requirements
- Shared Services

They prevent you from reinventing the wheel
Types of Operating Systems

- **Time-Sharing**
  - Maximize hardware utilization
- **Real-Time**
  - Guarantee latency

Photo by Leonardo Yip on Unsplash
IT vs OT

Information Technology
- Off-the-shelf
- Replaceable
- Frequent updates

Operational Technology
- Purpose-built
- Controls critical infrastructure
- Infrequent updates
How to Decide?

- What kind of hardware platform do you have?
- What are the requirements for the software?
- What peripherals and busses do you need to leverage?
Zephyr: Main Features

- Written in C
- Kernel services
- Multiple scheduling algorithms
- Configurable and modular
- Memory protection
- Native networking stack
- Bluetooth Low Energy 5.0 support
- Non-volatile storage
Zephyr Architecture

Applications
Smart Objects / High Level APIs / Data Models
- LwM2M
- MQTT
- HTTP
- CoAP
- DTLS
- TLS

Device Management
- Thread
  - RPL
  - 6LoWPAN
  - IPv6 / IPv4
- 15.4
- BLE
- Wi-Fi
- NFC
- ...

Low Level API
- I2C
- SPI
- UART
- GPIO
- ...
- File system
- Logging / Debug
- Database / Properties
- Crypto
- IPC
- Sensors
- ...

Kernel Services / Schedulers

Power Management

Platform
- Radios
- Sensors
- Crypto HW
- Flash

COPYRIGHT (C) 2023, ECLIPSE FOUNDATION | THIS WORK IS LICENSED UNDER A CREATIVE COMMONS ATTRIBUTION 4.0 INTERNATIONAL LICENSE (CC BY 4.0)
Kernel Services

- Multi-threading
  - Cooperative, priority-based, non-preemptive, and preemptive threads

- Interrupt
  - Compile-time registration of interrupt handlers

- Memory Allocation
  - Dynamic allocation and freeing of fixed and variable-size memory blocks

- Inter-thread Synchronization
  - Binary semaphores, counting semaphores, and mutex semaphores

- Inter-thread Data Passing Services
  - basic message queues, enhanced message queues, and byte streams

- Power Management
  - Tickless idle and an advanced idling infrastructure
Scheduling Algorithms

> Cooperative and Preemptive Scheduling
> Earliest Deadline First (EDF)
> Meta IRQ scheduling (“interrupt bottom half” or “tasklet” behavior)
> Timeslicing (between preemptible threads of equal priority)
> Available queuing strategies:
  • Simple linked-list ready queue
  • Red/black tree ready queue
  • Traditional multi-queue ready queue
Why Zephyr and Not <Insert Your Favorite>?

**Pros**
- Modularity
- Maturity
- Vendor-neutral governance
- Growing ecosystem

**Cons**
- C is susceptible to buffer overflows, segfaults and race conditions,
- Some alternatives have broader hardware support (but this is improving)

Photo by Emily Mortar on Unsplash
Rust
Rust and Constrained (Embedded) Devices

- Rust is attractive for constrained devices
  - Memory and thread safety
- Supports bare metal development
  - Bootloaders
  - Operating Systems
  - Applications
- Thriving embedded ecosystem; no mature RTOS
- Libraries (crates) → Cargo package manager
Language Characteristics

> Variables immutable by default
> Statically typed
> Memory safety without a garbage collector
  • Each value has an owner
  • Only one owner at a time
  • Variable is dropped when owner out of scope
> Inspired by functional and object-oriented programming languages but is neither
Two possible runtimes

- **Libstd (std crate)**
  - Sets up stack overflow protection
  - Spawns the main thread before the main function is invoked
  - Platform-specific (OS integration)

- **Libcore (core crate or no_std)**
  - Platform-agnostic subset of libstd
  - No heap (dynamic allocation) or collections unless additional crates added
My god, it's full of stars crates
Why Rust

Pros
- Memory and thread safety
- Momentum
- Vendor-neutral governance
- Growing ecosystem

Cons
- Language is still a moving target (new features are introduced frequently)
- More limited hardware support (but this is improving)
SHOWDOWN!!!!!
Which One Should You Pick?

Photo by sippakorn yamkasikorn on Unsplash
Which One Should You Pick?

It depends! A few points to consider

> Your skills... and those of people you could hire
  • Programming languages, frameworks, protocols

> Hardware support
  • Microcontroller, board, external sensors and actuators

> Ecosystem
  • Stability or velocity? Formal spec and multiple implementations (C) or open innovation (Rust)?
The True Winner: One Punch Man
About Eclipse IoT
IoT Architecture

CONstrained DEVICES
- Sensors
- Actuators
- Device Update Management
- Hardware Abstraction Layer
- Operating System
  - Real-Time Operating System

DEvelopment TOOLS

DEVELOPMENT TOOLS

Protocols
- Connectivity

SecuRity

HARDWARE

IoT EDGE GATEWAYS
- Remote Device Management
- Field Connectivity
- Edge Applications
- Edge Orchestration
- Operating System

EDGE SERVERS
- Machine Learning
- Edge Analytics

IOT PLATFORM
- APPLICATIONS
  - EdgeOps
  - API Management
  - Application Runtime
- INTEGRATION
  - Device Management
  - Device Connectivity
  - Data integration
- DATA
  - Data Management
  - Analytics
  - Machine Learning
Where Eclipse Projects Fit

CONSTRAINED DEVICES

- Sensors
- Actuators
   - Device Update Management
   - Hardware Abstraction Layer
   - Operating System / RTOS

EDGE

- Remote Device Management
- Field Connectivity
- Edge Applications
- Edge Orchestration
- Operating System
- Machine Learning
- Edge Analytics

IOT PLATFORM

APPLICATIONS
- EdgeOps
- API Management
- Runtimes

INTEGRATION
- Device Management
- Device Connectivity
- Data Integration

DATA
- Data Management
- Analytics
- Machine Learning

Connectivity

Security

Hardware

Protocols

COPYRIGHT (C) 2023, ECLIPSE FOUNDATION | THIS WORK IS LICENSED UNDER A CREATIVE COMMONS ATTRIBUTION 4.0 INTERNATIONAL LICENSE (CC BY 4.0)
Towards a Comprehensive Open Source IoT RISC-V stack
Thank you!

Questions?

Frédéric Desbiens
@BlueberryCoder

iot.eclipse.org
edgenative.eclipse.org
sparkplug.eclipse.org