



Zephyr Meetup, EW NA 2025, Anaheim, 2025-11-05

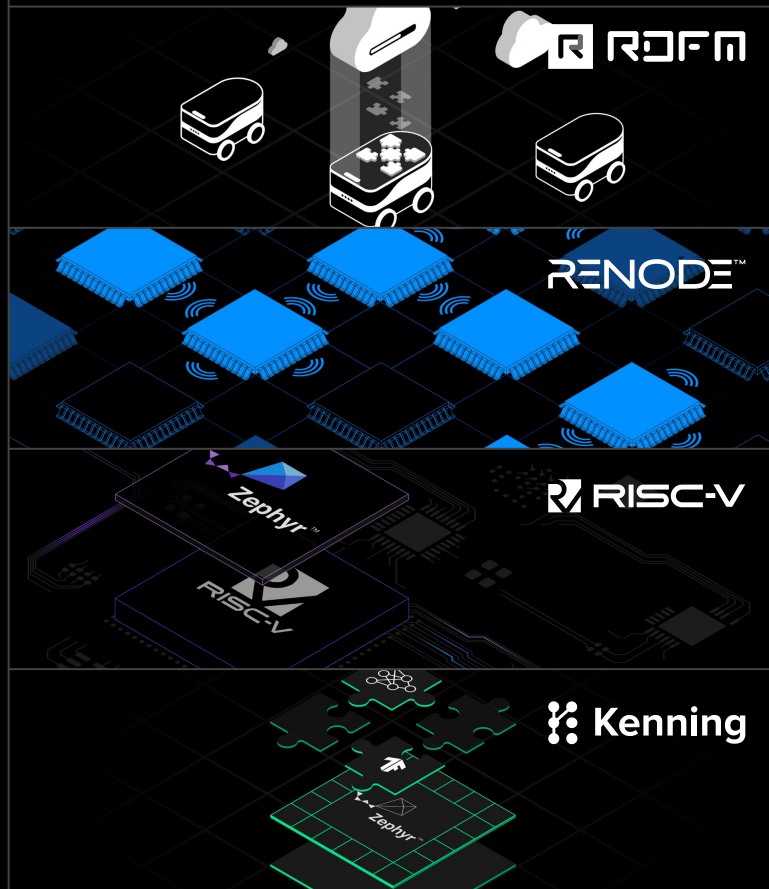
Making Zephyr AI development easier: Zephelin, Trace Viewer and AutoML



MICHAEL GIELDA
mgielda@antmicro.com

Antmicro work in Zephyr RTOS

- Over-The-Air updates and fleet management with RDFM
- Simulation and testing with Renode
- Maintaining RISC-V architecture
- AI model benchmarking, optimization and deployment with Kenning and Zephelin



Deploying AI models on embedded platforms

- Several things to take care of:
 - Selecting or designing an AI model architecture that have a chance to fit into the device in the first place
 - Optimizing model architecture and storage - quantization, pruning, etc.
 - Selecting an inference library that supports the given platform and meet applications constraints
 - Finding bottlenecks in the application at runtime to see what could be improved, or comparing several runtimes
- We also need to properly test the complete application with the model
- Nice to have some means to update the model or application when needed



Finding the right model and runtime

- There are at least several AI inference libraries that can run models on Zephyr platforms:
 - TFLite Micro/LiteRT | microTVM | IREE | ExecuTorch | ...
- Sparse support matrix, and even if multiple runtimes support your platform, finding which one performs best is not trivial
- Creating a small but decent quality model can be difficult on MCUs
- The open source Kenning framework helps with this process:
 - Allows seamless combination of various optimization algorithms and deployment using many runtimes
 - Supports a wide range of platforms, including Zephyr platforms
 - Provides a unified API allowing to easily test out various models and runtimes without the need to change a single line of code
 - Allows evaluation of models on-device



Repository

github.com/antmicro/kenning



Documentation

antmicro.github.io/kenning

Deploying models with Kenning Zephyr Runtime

- <https://github.com/antmicro/kenning-zephyr-runtime> allows evaluation and production-ready model deployment on Zephyr platforms
- Provides:
 - Unified API for various inference libraries (TFLite Micro, microTVM, ...)
 - Communication with host devices through simple protocols like UART
 - Evaluation app allowing to receive input data and deliver predictions along with additional statistics to host device for on-device evaluation
 - LLEXT-based support for updating models and entire inference libraries without the need to reflash the module
- Allows to verify:
 - Models, including models generated during AutoML search
 - Runtimes, allowing developers to see the performance of their solutions with various models and platforms
 - Hardware, e.g. when developing AI accelerators or when trying out possible solutions for users' applications

```
// ...
status_t status = STATUS_OK;
uint8_t *model_output = NULL;
size_t model_output_size = 0;

// initialize model
status = model_init();
RETURN_ON_ERROR(status, status);
// load model structure
status = model_load_struct((uint8_t *)&model_struct, sizeof(MlModel));
RETURN_ON_ERROR(status, status);
// load model weights
status = model_load_weights(model_data, model_data_len);
RETURN_ON_ERROR(status, status);
// allocate buffer for output;
model_get_output_size(&model_output_size);
model_output = malloc(model_output_size);

// sample inference loop
for (size_t batch_index = 0; batch_index < sizeof(data) /
    sizeof(data[0]); ++batch_index)
{
    status = model_load_input((uint8_t *)data[batch_index],
        sizeof(data[0]));
    RETURN_ON_ERROR(status, status);

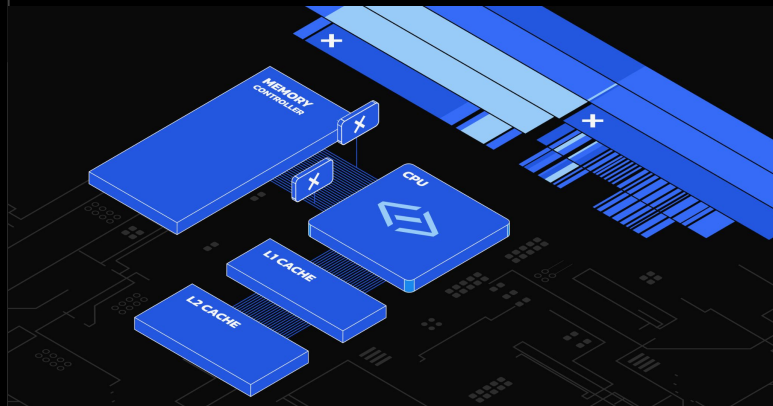
    status = model_run();
    RETURN_ON_ERROR(status, status);

    status = model_get_output(model_output_size, model_output, NULL);
    RETURN_ON_ERROR(status, status);

    format_output(output_str, sizeof(output_str), model_output);
    LOG_INF("model output: %s", output_str);
}
// ...
```

Testing end-to-end AI applications with Renode

- Renode: Antmicro's open source emulation framework allowing software developers to build, run and test software without hardware
- Deterministic, built with automation and testing in mind
- Run unmodified software in simulation for ARM, RISC-V, SPARC...
- Simulate heterogeneous multi-core CPUs, peripherals, AI coprocessors, sensors, effectors, other devices, multiple nodes, wired/wireless protocols
- Test embedded software end-to-end repeatedly in CI
- Co-simulates RTL designs (for e.g. development of AI accelerators)



RENODE™



Repository

github.com/renode/renode

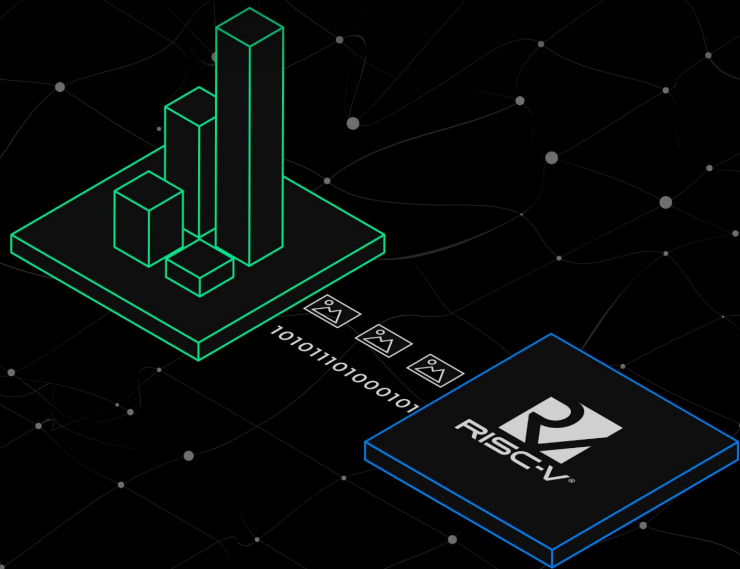


Website


renode.io/

Renode + Kenning integration

- Kenning comes with seamless integration with Renode, allowing to switch between simulated devices and actual devices
- Kenning uses Renode for additional metrics like used CPU instructions
- With Renode we can easily profile and evaluate AI models with Kenning during AutoML search, or verify application / inference library / model across all platforms supported by Zephyr - check out [Zephyr Dashboard](#)



Tested at scale with Renode: Zephyr Dashboard






ARCHITECTURE


- ARC
- ARM32
- ARM64
- MIPS
- RISCV32
- RISCV64
- RX
- SPARC
- X86
- X86-64
- XTENSA

BUILD DETAILS

☒ SHOW SIMULATION

 497209C74E

 ACD4851C0C 

 Search...

BOARD NAME	HELLO WORLD	PHILOSOPHERS	SHELL MODULE	TENSORFLOW LITE MICRO	MICROPYTHON	BLI
RISCV64 (14)						
BeagleBoard BeagleV@-Fire [soc: polarfire] [variant: e51] polarfire	PASSED	PASSED	PASSED	PASSED	PASSED	NOT
BeagleBoard BeagleV@-Fire [soc: polarfire] [variant: u54] polarfire	PASSED	PASSED	GENERATED	PASSED	GENERATED	NOT
BeagleBoard BeagleV@-Fire [soc: polarfire] [variant: u54/smp] polarfire	PASSED	PASSED	PASSED	PASSED	PASSED	NOT
Microchip Technology mpfs_icicle [soc: polarfire] [variant: e51] polarfire	PASSED	PASSED	PASSED	PASSED	PASSED	PAS
Microchip Technology mpfs_icicle [soc: polarfire] [variant: u54] polarfire	PASSED	PASSED	PASSED	PASSED	PASSED	PAS
Microchip Technology mpfs_icicle [soc: polarfire] [variant: u54/smp] polarfire	PASSED	PASSED	PASSED	PASSED	PASSED	PAS
OpenHW Group OpenHW Group cv64a6 on Genesys 2 cv64a6_imafdc	PASSED	PASSED	PASSED	PASSED	PASSED	NOT
SiFive HiFive Unleashed [soc: fu540] [variant: e51]	PASSED	PASSED	PASSED	PASSED	PASSED	NOT

Open source OTA updates of your firmware and model

- RFDm - open source OTA updates, including partial updates (e.g. AI models) and fleet management
- RFDm OTA updates provide:
 - Signing and verification of delivered updates
 - Mechanisms to commit and rollback updates
 - Mechanisms for running health checks upon update
- RFDm supports Linux, Android and **Zephyr RTOS**
- RFDm on Zephyr uses MCUBoot for performing A/B updates and rollbacks, and current and previous application version slots
- RFDm Management Server talks to MCUmgr which talks to the Zephyr platform using BLE, UDP or Serial



Repository

github.com/antmicro/rdfm

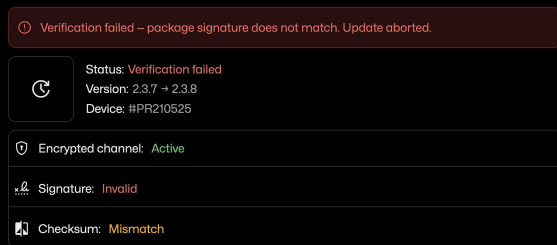
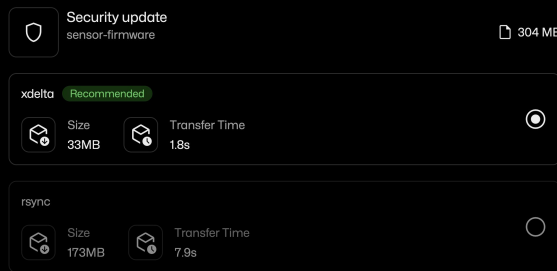
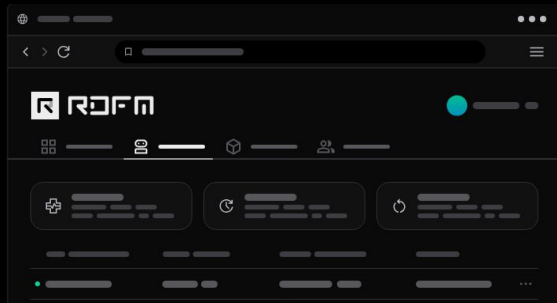


Documentation

antmicro.github.io/rdfm

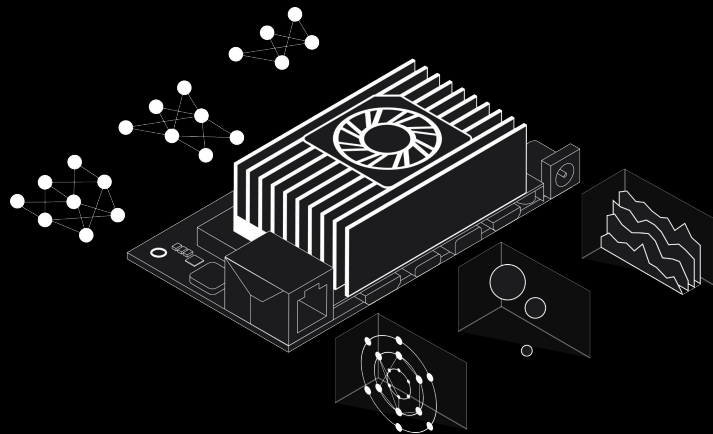
Remote Device Fleet Manager features

- Management Server for overseeing, managing devices, deploying updates
- Management Website / UI to control devices from the browser
- Tools for creating update files, firmware updates, delta/xdelta updates for smaller images, and partial updates for updating e.g. AI models
- Useful scripts, Yocto recipes, Zephyr library
- Easy integration with:
 - Authentication services (e.g. Keycloak, Amazon Cognito)
 - Storage services (for updates, e.g. Amazon S3 buckets)
 - Databases (for devices and packages data, e.g. PostgreSQL, Amazon Aurora)



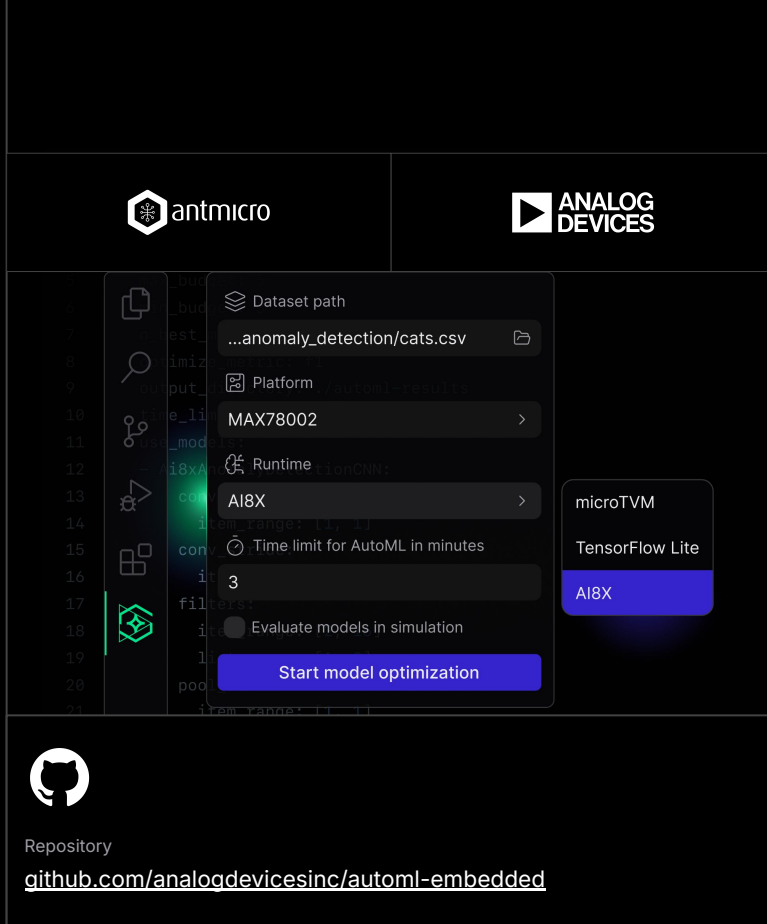
Creating tailored models with Kenning AutoML

- Kenning's platform-aware AutoML module in its most basic form takes:
 - Target platform - allows determining available memory, compute capabilities, architecture and recommended inference libraries
 - Dataset to train the model
 - Problem the model is supposed to tackle (e.g. anomaly detection)
 - End condition (time limit or e.g. expected accuracy)
- The AutoML module then:
 - Searches for the right parameters for the training and the right model architecture (NAS) given the configuration space
 - Trains another ML model to find increasingly better model candidates
 - Verifies models before training to make sure they can be deployed on the target device with the selected runtime and post-training optimizations (quantization, pruning, ...)
- Selected best candidates based on size and quality are evaluated on-device to generate [the final report on performance and quality](#)



AutoML plugin for embedded platforms

- VSCode plugin built for Analog Devices, encapsulating Kenning's AutoML features for generating anomaly detection models for Zephyr apps
- Given the dataset and target platform it allows to find the model fitting in the platform within requested amount of time
- It allows picking the best model based on user's needs - user can pick between smaller, faster or better quality models
- Generates necessary header files with model data and implementation that are later picked by Kenning Zephyr Runtime - seamless flow for experimenting with models, runtimes and platforms

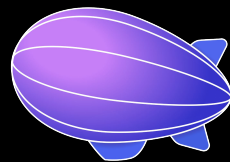
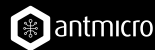


Repository

github.com/analogdevicesinc/auttml-embedded

In-depth AI model and runtime profiling in Zephyr

- AI models and other compute/resource-heavy apps need optimizations across many dimensions on embedded platforms
- Also when developing AI runtimes we may want to dig into performance issues and bottlenecks to implement better kernels
- As part of Antmicro's ongoing collaboration with ADI, we developed Zephyr Profiling Library called Zephelin for those use cases
- It's used to profile Zephyr apps and focuses on AI model execution
- It collects traces from execution with either:
 - **Tracing subsystem** and predefined or user-provided events (allowing to track functions, sections of code, loops, etc.)
 - **Instrumentation subsystem** (introduced to Zephyr in this project) that uses the compiler's instrumentation feature to mark entering and leaving functions, allowing fine-grained and low-level analysis of applications



Zephelin



Repository

github.com/antmicro/zephelin

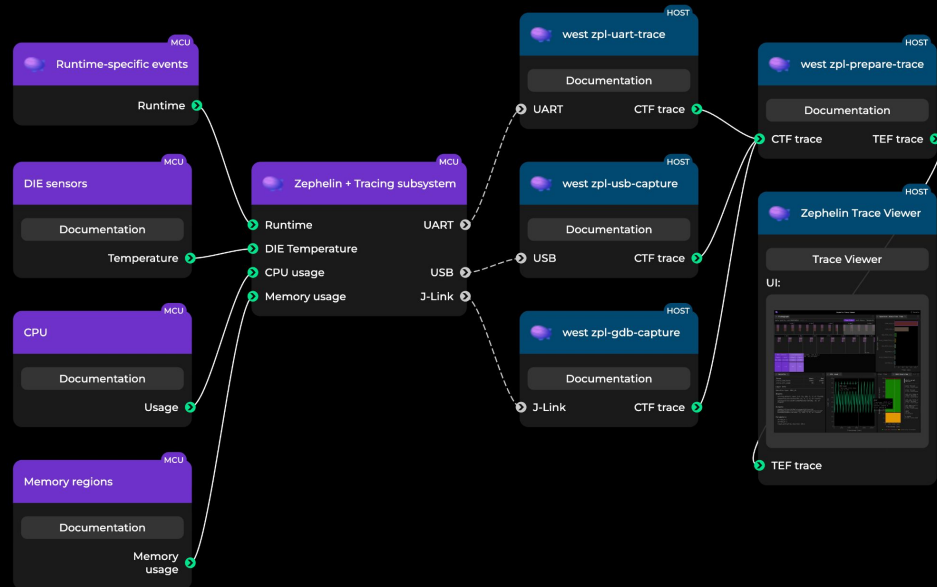


Documentation

antmicro.github.io/zephelin

Zephelin features

- Allows to analyze:
 - Regular Zephyr apps
 - Internals of the Zephyr RTOS (with instrumentation subsystem)
 - AI runtimes
 - Multithreaded applications
- Collects:
 - Traces
 - User-defined code scopes
 - Resource usage data provided by Zephyr subsystems - CPU load, memory usage (from runtime and west RAM/ROM reports)
 - Die temperature
 - Model-related data:
 - Global - inference
 - Per-layer - memory consumption, average/total time, parameters (e.g dimensions of input, output and weight tensors)



Zephelin - trace collection

- Zephelin outputs traces in text or binary Common Trace Format (CTF)
- Zephelin extends Zephyr's west command with:
 - Commands for capturing traces through
 - UART - west zpl-uart-capture
 - USB - west zpl-usb-capture
 - GDB - west zpl-gdb-capture
 - Command for enhancing collected traces with additional data and converting traces to widely supported TEF format for further processing
 - west zpl-prepare-trace
 - Trace preparation enhances traces with:
 - Build data - RAM/ROM report for memory analysis
 - AI model data - adding details on layers' types and parameters based on model data (TFLite) or compilation-level metadata (TVM)

```
$ west build -p -b max78002evkit/max78002/m4 samples/events/named_event --
-DCONFIG_ZPL_TRACE_FORMAT_CTF=y
$ west flash

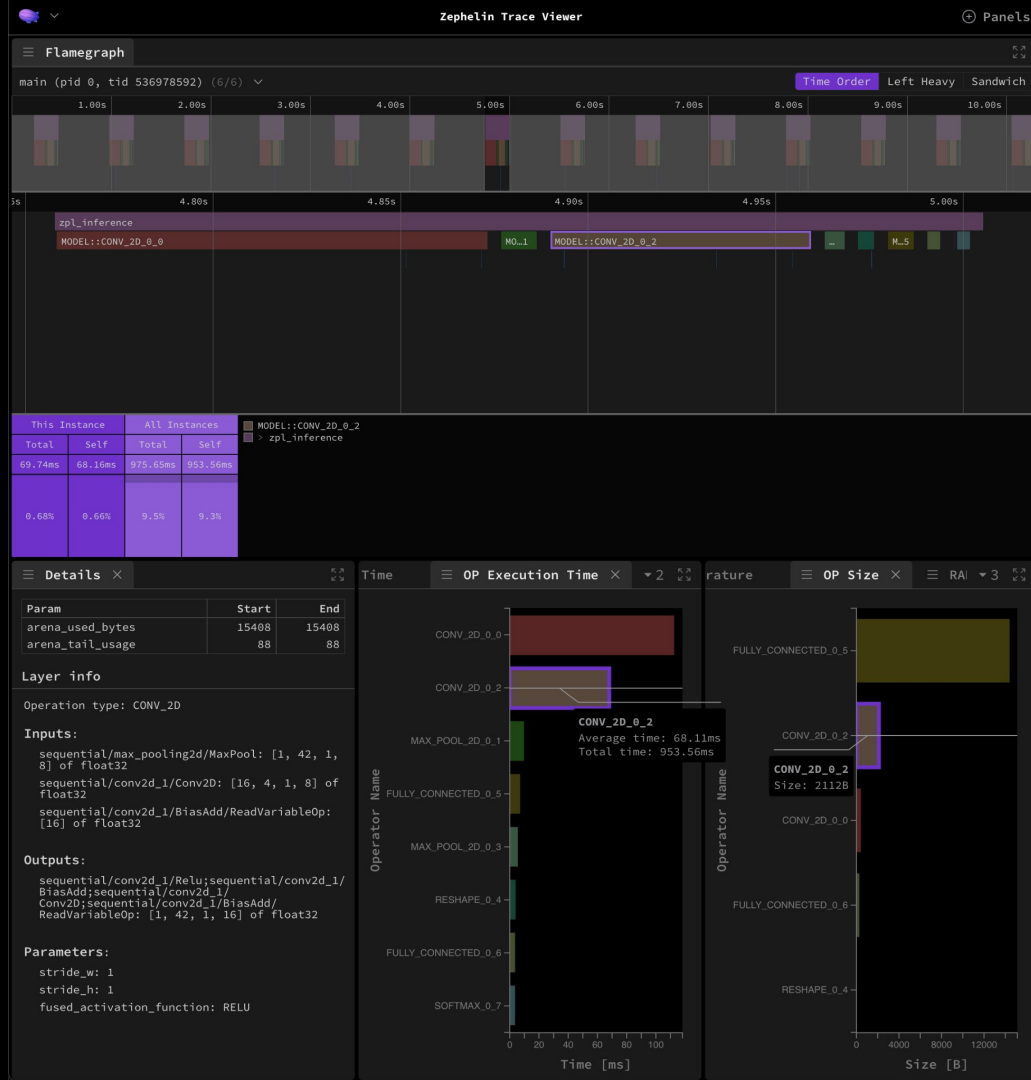
$ west zpl-uart-capture /dev/ttyACM0 115200 capture-ctf.bin
Capturing traces on /dev/ttyACM0@115200...
Press C-c to stop.

$ hexdump -C capture.bin | head -n 5
00000000 63 6f 75 6e 74 65 72 5f 76 61 6c 75 65 00 00 00 |counter_value...|
00000010 00 00 00 00 11 00 00 00 00 00 00 00 22 33 4d 31 |....."3M1|
00000020 62 63 6f 75 6e 74 65 72 5f 76 61 6c 75 65 00 00 |bcounter_value...|
00000030 00 00 00 00 00 12 00 00 00 00 00 00 00 c2 83 e9 |.....|
00000040 6c 62 63 6f 75 6e 74 65 72 5f 76 61 6c 75 65 00 |lbcounter_value..|
```

```
$ west zpl-prepare-trace -o capture-tef.json capture-ctf.bin
$ jq < capture-tef.json | head -n 14
[
  {
    "name": "counter_value",
    "cat": "zephyr",
    "ph": "B",
    "ts": 0.0,
    "pid": 0,
    "tid": 0,
    "args": {
      "name": "counter_value",
      "arg0": 0,
      "arg1": 0
    }
  },
  {
```

Trace Viewer

- <https://antmicro.github.io/zephelin-trace-viewer/>
- [Examples of traces](#)
- Visualizes TEF traces from Zephelin providing a detailed analysis of collected tracing, resource utilization and AI runtimes data:
 - Per-layer processing time, memory usage and parameters
 - CPU usage
 - Die temperature
 - Memory usage
 - RAM overview
- Panels are synchronized to highlight the timestamp you are focusing on
- [Interactive example](#)





Zephyr Meetup, EW NA 2025, Anaheim, 2025-11-05

Want to build AI-enabled Zephyr products?

Discover more at:

offering.antmicro.com

antmicro.com