Zephyr on a hearing aid

Mark Ruvald Pedersen / Oticon
Klaus Petersen / Oticon
"Zephyr is a real-time-operating-system optimized for for resource constrained devices."

- A hearing aid is resource constrained in a number of ways:
  - Physical dimensions
  - Computationally: MIPS and memory
  - Battery life

- Complex commercial product for the end-consumer - in volume
Why this talk?

"If you can run it on a hearing aid, you can run it anywhere"

• Not about:
  Low level porting code to a specific arch. Other talks have addressed that.

• Is about:
  The challenges and solutions of using Zephyr on a very custom system - Oticon hearing aids.
Not just a glorified amplifier

- Advanced **DSP algorithms**, always-on sound processing
- **Wireless** 2.4GHz Bluetooth Low Energy and proprietary protocols
- **Magnetic** link between left/right hearing aid
- Battery the size of an **Altoids mint**
Under the hood of a hearing aid

- **3 custom ASICs**: DSP, RF, Analog
- Many-core and real-time system
- Network on chip (NoC)
- **Bluetooth** baseband
- Custom processors, each optimized for specific tasks
Custom processor ➔ Custom tools

First challenge:

• Remove road-blocks for non-gnu/commercial compilers, by abstraction
Toolchain abstraction - before 😞

- Zephyr was limited by GNU assumptions in both C and **CMake**. E.g. **/CMakeLists.txt** was a mixed-bag of hardcoded flags&concerns:

```cpp
zephyr_compile_options(
    ${OPTIMIZATION_FLAG}  # Usually -Os
    -g  # TODO: build configuration enough?
    -Wall
    -Wformat
    -Wformat-security
    -Wno-format-zero-length
    -imacros ${AUTOCONF_H}
    -ffreestanding
    -Wno-main
    ${NOSTDINC_F}
    ${TOOLCHAIN_C_FLAGS}
)
```
Toolchain abstraction - after 😊

• Abstract {compiler, linker, ...} flags by **intent** and categorize. Now each toolchain can implement without loss of translation:

```c
toolchain_cc_warning_base()
toolchain_cc_freestanding()
toolchain_cc_nostdinc()
toolchain_cc_nocommon()
toolchain_cc_imacros(${AUTOCONF_H})
zephyr_compile_options(
    ${TOOLCHAIN_C_FLAGS}
)
```

• Still rough edges: userspace, `-L` in `ext/`, still-messy `/CMakeLists.txt`

• Magic `zephyr_cc_option` can be OK, but still taint root CMake file
Zephyr: Small part of a big system

Second challenge:

- Support out-of-tree code
Out of tree code

- Motivation
  - Specialized system (Processor, Network on Chip, IP)
- Proprietary code
  - Drivers
  - Subsystems
  - 3rd party code

Open sourcing would be preferred but not always possible.
Out of tree (Solution)

- Solution
  - Supporting out of tree code
    - Architectures
    - Boards
    - Toolchain
    - Drivers (TBD)
    - Subsystems (TBD)
Out of tree (Solution cont.)

- Why not .gitignore?
  - Proprietary is in separate version control systems
  - Others?
Endian portable Zephyr BT LL stack

Third challenge:

- Zephyr largely already endianess portable
- Hard to find endian issues
- Generic semi-automatic solution
Endianness bugs

- Subtle, rampant, time-consuming to locate by review, easy to fix once located, likely to reappear (World is LE, except us).

- Endian-portable: Zephyr proper. BT {Host, Controller}.

- Static analysis? Resort to {review, empirical discovery}:
  - **BabbleSim**: Reproducible, faster, better debug than in BE Target. Reproducible: Do not worry about wireless stochastic behavior.

BabbleSim ∈ Linux userland: **Capitalize on the best tools open source can buy**: QEMU, Valgrind, rr/gdb, Coccinelle, Docker, ...

- **Detect**: Run tests under both `qemu-mips{,el}`-static. Diff logs.
- **Locate**: Use Coccinelle to instrument code, inject traces. Diff.
debian_foreign: `for_each_arch $payload`

Docker of userlands (Debian \{mips, mipsel, arm64, ...\}) + QEMU user.

schroot: 32-bit debian MIPS (QEMU)

Semantics patch: Trace executed statements

```c
@tracecontrolflow@
type TR;
identifier fun;
expression e1;
@@
TR fun(...) { <+...
- e1;
+ MYTRACE(e1);
+ e1;
...+> }

Make up a suitable `MYTRACE` macro.
E.g. arch/posix/include/posix_cheats.h:

```c
#define MYTRACE(e)                     
    printf(                              
        "MYTRACE %20s:%-4i : %-40s %s\n"   
        , __FILE__, __LINE__, __FUNCTION__ 
        , #e                               
    )
```
@trace_assignment_scalar_u8@
type TR; identifier fun;
{uint8_t, u8_t, unsigned char} x;
assignment operator ASSOP = { =, +=, -=, *=, /=, %=
, <<=, >>=, &=, ^=, |= };
expression e;
@@
TR fun(...) { <...
x ASSOP e;
+ MYTRACE_SCALAR_U8(x); /* trace_assignment_scalar_u8 */
...> }
Semantic patch: Final touches

Achieve idempotence:

```c
@remove_traces@ @@
- MYTRACE(...);
- MYTRACE_CASE(...);
- MYTRACE_THEN_LEAVE(...);
- MYTRACE_ELSE_LEAVE(...);
- MYTRACE_RETURN_VOID(...);
- MYTRACE_SCALAR_U8(...);
```

{Review, merge} cocci result:

```c
function instrument {
    local in="$1"
    local out="${in}.cocci_tmp"
    spatch                      
        --disable-multi-pass 
        --no-show-diff      
        --sp-file inject.cocci
        "$in" -o "$out"
    && {
        sha256sum "$out"
        meld "$in" "$out"
    }
}
```
Oticon Zephyr-contributions overview

- **Portability** pipecleaning ~ Fixing hard GNU assumptions
  - Standard C99: Fix `void*` arith, semicolons. 6 commits
  - Toolchain abstraction in CMake: cc, ld, binutils. >28 commits
- Out of tree {Toolchains, Arch, BT {HCI commands, LLL}}.
- Bluetooth subsystem: LL split, Endianess, Settings storage, Vendor extensions, Audio-aware scheduler, BLE 6.0 collab (WIP, private).
- native_posix: Run Zephyr kernel+app as normal Linux executable.
- BabbleSim: PHY simulator+model libraries, e.g. BLE via NRF52.
- Tracing: CTF backend (Babeltrace/TraceCompass/KernelShark2.0)
Future work and direction

- More {Bluetooth LE, Optimizations}.
- Toolchain abstraction CMake work is near completion. Still too low-level for our taste but sufficient for Oticon.
- Improve graphical presentation of Zephyr CTF in TraceCompass. Investigate KernelShark 2.0 (still WIP).
- Zephyr as a component: More modularity, stable and efficient APIs/HALs
Resources

- https://babblesim.github.io
- https://docs.zephyrproject.org/latest/boards/posix/native_posix/doc/index.html
- https://github.com/mped-oticon/debian_foreign
- https://docs.zephyrproject.org/latest/guides/tracing/ctf.html
Questions?