

Industrial & Open-Source Crossover

LLEXT: a new Zephyr subsystem

Presented by: Cedric Lescop



Industrial need: Flexibility





New technology identified: LLEXT subsystem

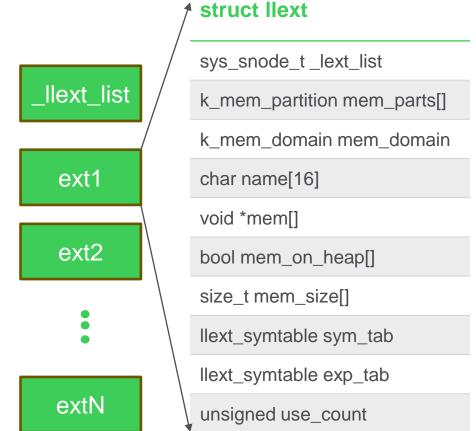
Linkable Loadable EXTensions state-of-the-art

- What does it do?
 - Load data into memory
 - Using ELF file
- How?
 - Serial back-end
 - Parse the whole instruction code
- Simple test example
 - 1 printk("Hello, World")



LLEXT: Extension manager

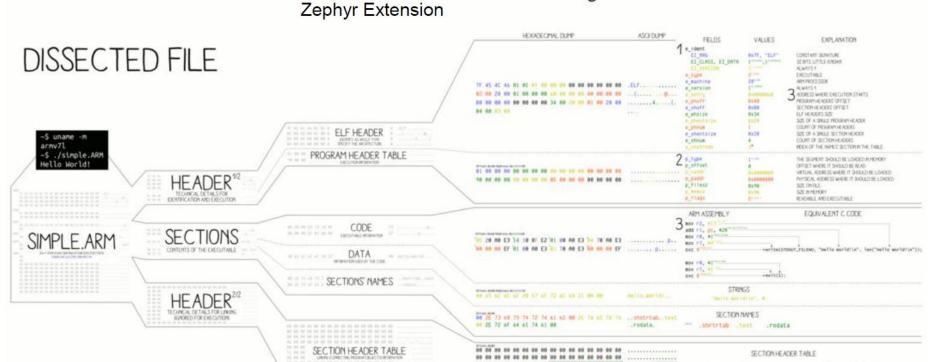
- Ilext_load(), Ilext_unload(), Ilext_find_sym()
- Maintains a list of loaded extensions
- Maintains a reference count for each extension
- Manages memory for extensions
- Manages symbol tables for the base firmware and each extension





ELF¹⁰¹ a Linux executable walk-through







A008-55

6 2 10 0x 60000000

WEST SE

0×8000000 ((×8)) ((×8))

RAIL.

PEEK SAVE

3" , shrtrtab

84 88 88 88 68 80 80 80 88 sil 88 ch 10 29 88 88 80 80

88 88 88 88 88 88 88 88 88 54 55 55 55 50 88 88

11 40 40 00 01 01 01 01 40 87 80 60 60 58 80 50 00

98 98 99 98 99 99 99 99 99 99 99 99 99

Relocations: Linking and Placement

Disassembly of section '.text' using gcc:

arm-zephyr-eabi-objdump -r -d -x hello_world.elf gives:

```
0: b580
         push {r7, lr}
2: af00 add r7, sp, #0
4: 4b08
             ldr r3, [pc, #32]; (28 < hello world + 0x28 > )
28: 00000004
                .word 0x00000004
                                                           Rewrite with address of .rodata + 0x00000004
           28: R ARM ABS32 .rodata
2c: 00000000
                .word 0x00000000
           2c: R ARM ABS32 printk
                .word 0x00000014
30: 00000014
                                                           Rewrite with address of printk function
           30: R ARM ABS32 .rodata
                  r3
34: 4718
              bx
36: 46c0
                           ; (mov r8, r8)
              nop
```



00000000 <hello world>:

Relocations: some key points

- Kinds of Relocations that show up depend on
 - Architecture: x86, armv7m, xtensa lx7, etc
 - ELF linkage: shared, static, relocatable
 - Compiler flags: e.g. -mlong-calls
- May require opcode decoding, updating, and reencoding
- May require generation of a jump table
 - Opcodes are sometimes location dependent
 - Limits range of address accessibility for that opcode
 - E.g. PC relative call instructions encoded as 16bit instruction opcode



Proof-of-Concept

Technical view

- <inf> app: Loading plugin count /NAND:/plugin counter strip.elf Using a 00:00:00.076,000] <inf> app: Successfully loaded plugin count @ 0x200202c0 in 9535037 ns / 762803 cycles <inf> app: Starting plugin count
- <inf> app: Loading plugin can /NAND:/plugin_canopen_strip.elf
 <inf> app: Successfully loaded plugin can @ 0x20021cc0 in 160411750 ns / 12832940 cycles [00:00:00.237,000] <inf> app: Starting plugin can
- Work with or without MPU
- Going further
 - XIP (eXecute In Place) to execute from flash
 - Zephyr User-space
 - Performance measurement



write

paramètres

callback

debug

GPIO

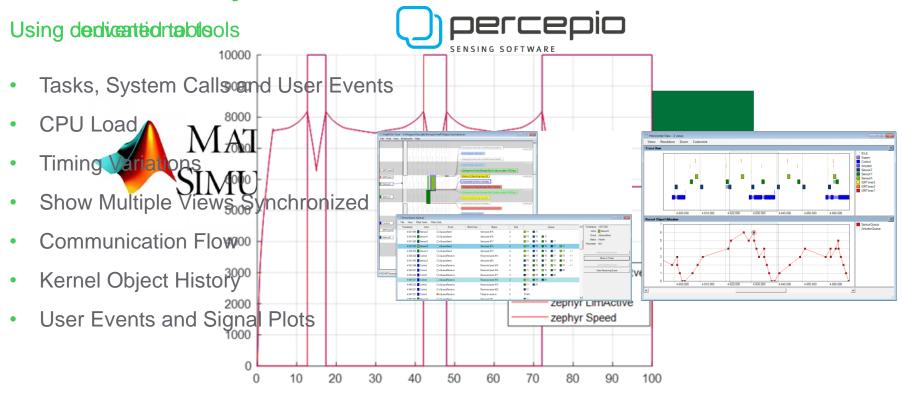
node plugin

CAN



CAN bus transceiver

Real-time Analysis





Impacts on products

Beneficits vs. drawbacks

- Updates
 - Partial / isolated / processes
 - Faster application build and load
- Cybersecurity
 - Expose to new threats: untrusted ELF could do damage
- Safety
 - Need new certification?



Credits

LLEXT was created by numerous people, and built on Zephyr infrastructure

- Chen Peng, Initial ELF Loader (Former (intel))
- Lucas Burelli, Armv7m and CMake tooling (Arduino)
- Guennadi Lyakh, Xtensa Support (intel Sound Open Firmware)
- Ederson de Souza, EDK ((intel) Zephyr Team)

A special thank you to **Tom Burdick** for gluing it together and letting me largely copy his presentation!





Life Is On Schneider

se.com









