ELC 2017

War Story: Using Zephyr to Develop a Wearable Device

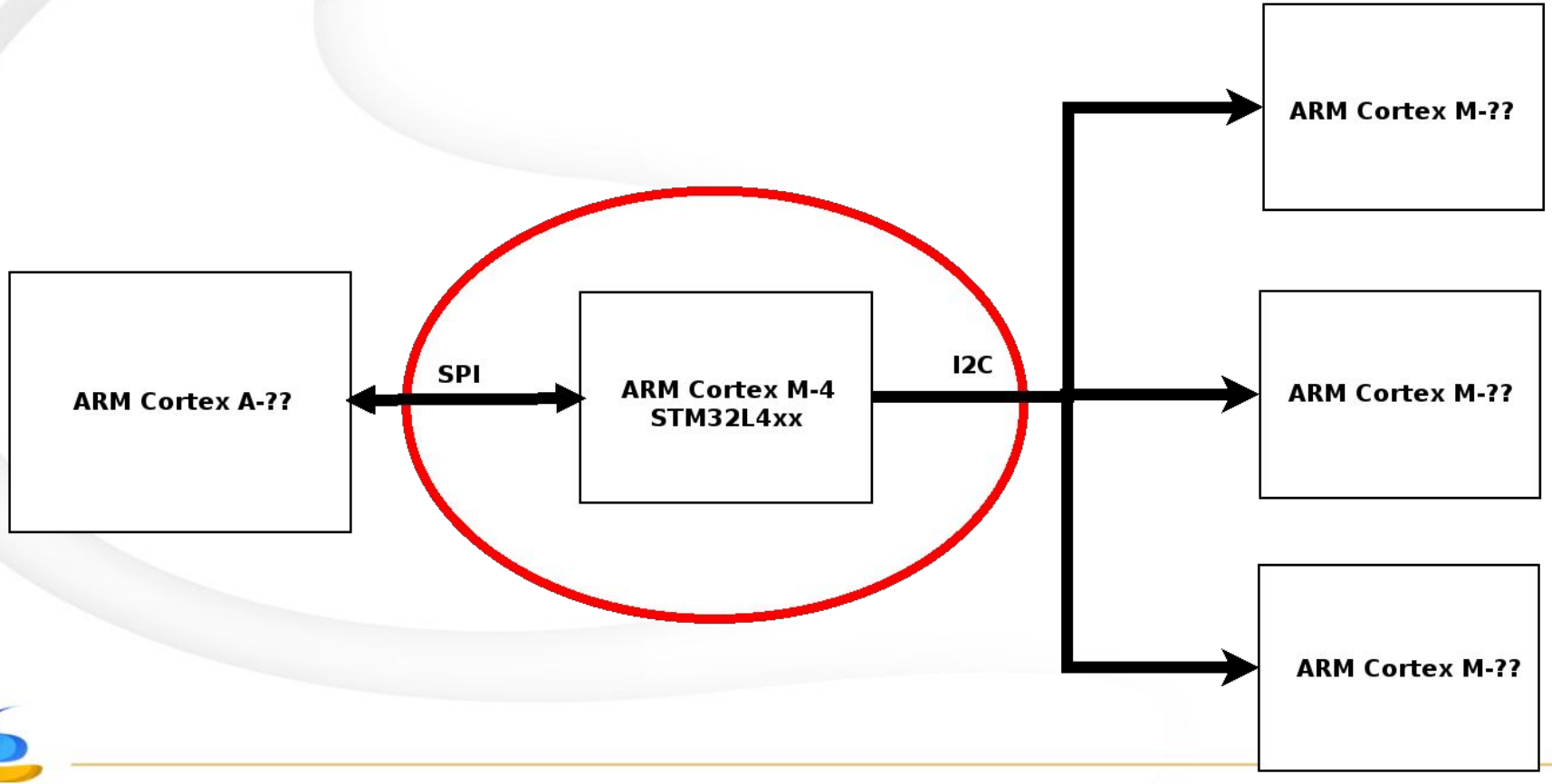
Neil Armstrong & Fabien Parent

Agenda

- Based on a true story
- Many choices, why Zephyr?
- Zephyr in a nutshell
- Porting & upstreaming a new platform
- Community differences compared to Linux
- Conclusion


True Story

Platform


ARM Cortex A-??
SPI
ARM Cortex M-4
STM32L4xx
I2C
ARM Cortex M-??
ARM Cortex M-??
ARM Cortex M-??

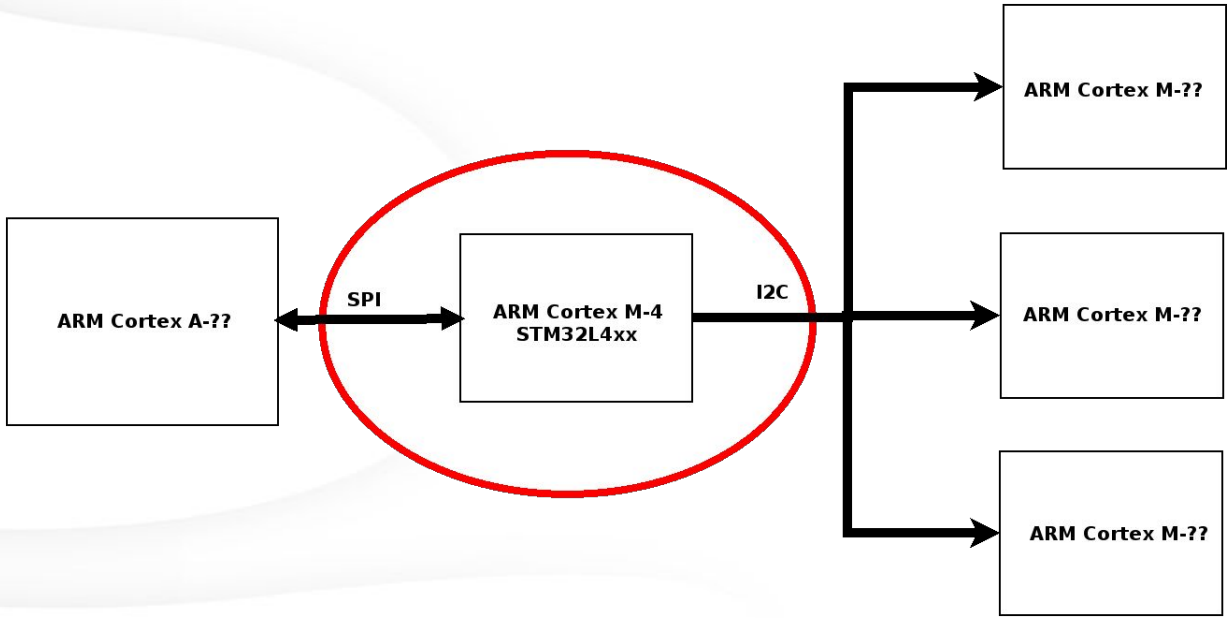
OS requirements

Drivers

- UART
- I2C master driver
- SPI slave driver

Basic OS features

- Scheduler
- Timers
- Task/Threads
- Locks
- ...



Choosing the RTOS

Our constraints:

- Permissive License
- Free (as in free beer)

Zephyr™

List of candidates

- NuttX
  - 3-clause BSD license
- Zephyr
  - Apache License 2.0
- Bespoke OS

Option 1: Bespoke OS

Pros

- Fun to implement
- Can be tailored to our needs
- Easy to understand the whole code base

Cons

- Takes time
- No community, fewer eyes on code
- Little time to mature and fix early bugs

Option 2: NuttX

Pros

- Familiar, used it on previous project (Google’s Project ARA)
- Already supports our SoC (STM32L4xx)

Cons

- Build system is completely unreliable
- Optional GPL components are scary
- No meaningful community
- BFDL contributions and maintenance are questionable

Option 2: NuttX

```
commit 3810231337f94c28e41d36815eb823d2811610b7
Author: Gregory Nutt <gnutt@nuttx.org>
Date:   Fri Sep 26 07:15:11 2014 -0600

    Typo fix. Hmmm... how did this compile before?

diff --git a/nuttx/fs/fs_poll.c b/nuttx/fs/fs_poll.c
index cbe30481b5..78d68602cc 100644
--- a/nuttx/fs/fs_poll.c
```

No peer review?

```
commit 100d49eeb4e49816af1b3279f7cddeecf9cdd07d
Author: Gregory Nutt <gnutt@nuttx.org>
Date:   Wed Sep 17 09:52:07 2014 -0600

    Ooops... a file that I forgot to add yesterday

diff --git a/nuttx/arch/arm/src/armv7-a/arm_virtpgaddr.c
new file mode 100644
```

Option 3: Zephyr

Pros

- Similar in many ways to Linux (coding style, Kbuild, Kconfig, …)
- Strong community is a goal
- Embraces maintainers concept
- Great documentation

Cons

- No support for STM32L4xx
- Project still very young, unsure of its maturity
- Not aware of any real product using it

Verdict

Obvious winner:

Option X: Apache myNewt

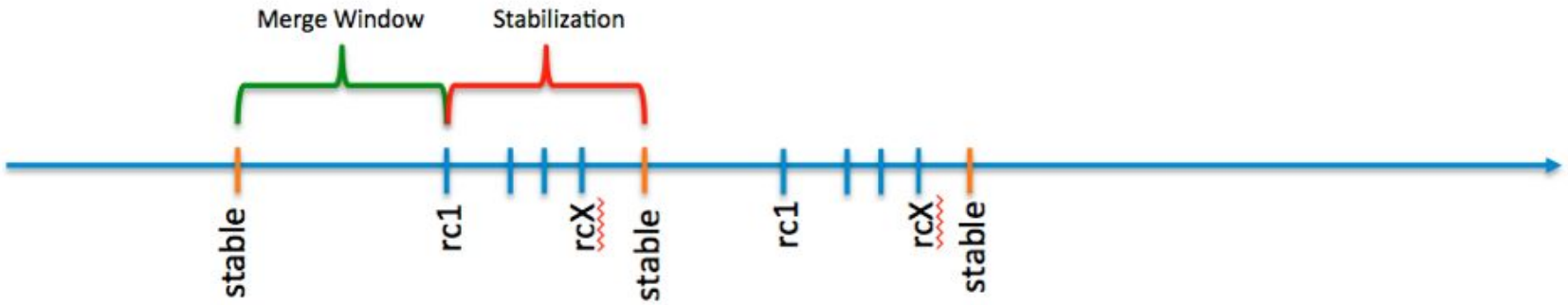
Came to our attention after OS selection was completed

Zephyr in a nutshell

Zephyr Features

- Lib C (newlib)
- Designed for low memory usage (everything is statically allocated)
- Highly configurable and modular
- Cooperative and preemptive threading
- Pre-certification for security (someday)

Development Lifecycle



Zephyr 1.6
Release cycle: ~3/4 months
Merge Window: ~11 weeks
Stabilization: ~3 weeks

Linux 4.8
Release cycle: ~2/3 months
Merge Window: ~2 weeks
Stabilization: ~8 weeks

Leadership

- Linux Foundation Technical Steering Committee (TSC)
  - https://lists.zephyrproject.org/pipermail/zephyr-tsc
- Maintainers do not have to be Zephyr Project/TSC members
- Sub-maintainer concept same as Linux
- Planning and technical decisions spread across JIRA, Gerrit & mailing lists
- Blog posts controlled through separate committee

Top-down development

- Features planned in advance
  - Unified Kernel (ZEP-334)
  - New IP Stack (ZEP-322)
  - Thread Protocol (ZEP-337)
- Delayed features may delay closing the merge window
- Do planned features have priority vs community contributions?

Source:
https://lists.zephyrproject.org/pipermail/zephyr-tsc/attachments/20160817/90b00994/attachment-0001.pdf

Adding STM32L4xx support to Zephyr

s/STM32F1/STM32L4/g

Solution: copy/paste the closest SoC/board and use them as example.

```
cp -r arch/arm/soc/st_stm32/stm32{f1,l4}

cp drivers/clock_control/{stm32f107xx_clock.c,stm32l4xx_clock.c}

cp drivers/pinmux/stm32/pinmux_board_nucleo_{f103rb,l476rg}.c

cp drivers/serial/uart_{stm32,stm32lx}.c

cp -r boards/nucleo_{f103rb,l476rg}
```

Porting is fast & easy

- Tested platform support added in less than a week
  - STM32L4xx CPU
  - UART
  - I2C
  - SPI

- Most of that time was spent on I2C/SPI testing

Upstreaming the STM32L4 port

1. RTFM

   https://wiki.zephyrproject.org/view/Collaboration_Guidelines

2. Clean-up patches to follow coding standard and run “checkpatch”

3. Upload patches to gerrit

4. Wait for reviews

Upstreaming the STM32L4 port

1. RTFM

   https://wiki.zephyrproject.org/view/Collaboration_Guidelines

2. Clean-up patches to follow coding standard and run “checkpatch”

3. Upload patches to gerrit

4. Wait for reviews

5. Ping maintainers on IRC

Upstream ASAP!

- Zephyr is young and its APIs are changing fast
- Rebases are painful
- Power management implementation needed to be rewritten 3 times
- Conclusion: merge code quickly before the base shifts beneath you

Community differences compared to Linux

Zephyr and HALs

“

Our goal is [...] porting based on STM32 Cube SDK [...] Would it be ok for you to hold on your upstream, waiting for us to come up with our implementation proposal [...]

”

- Gerrit review comment

Source:
https://gerrit.zephyrproject.org/r/#/c/5194/

Linux and HALs

“

No HALs. We don't do HALs in the kernel.

”

- Dave Airlie, Linux DRM Maintainer

Source:
https://lists.freedesktop.org/archives/dri-devel/2016-December/126516.html

To HAL or not to HAL?

Input requested from maintainers

None is given

Result: vendor HALs are slowly replacing native drivers

Source:

https://lists.linuxfoundation.org/pipermail/zephyr-devel/2016-November/006633.html

Maintainers

“

The reason the toplevel maintainer (me) doesn't work for Intel or AMD or any vendors, is that I can say NO when your maintainers can't or won't say it.

”

- Dave Airlie, again.

Review tools

Gerrit

JIRA

Mailing list

Great talk from Greg K.H. about the tools used for Linux: https://youtu.be/L8OOzaqS37s

JIRA

Pros

- Manager-friendly

Cons

- Developer-unfriendly
- Yet another communication medium in addition to mailing-lists and gerrit
- Do community contributors need to use it?

Gerrit

Pros

- Easy to not forget patches

Cons

- Slow
- Unnecessarily complicated
- Patch submitter selects reviewers; no broadcast
- No patch series
- Archive search is bad

Mailing lists

Pros

- They work

Cons

- None

Conclusion

Zephyr Pros

- Similar enough to Linux
- APIs are simple and well documented
- Has a real and active community
- Good design for low memory usage and/or performance on small CPUs
- Flaws are quickly getting fixed

Conclusion

Zephyr Cons

- No HAL please!
- Tools for review make us sad
- Maintainer review has been discouraging
- Moving target: code is still young, APIs are changing fast

Questions?