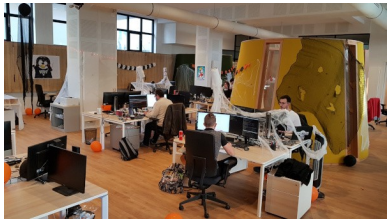
IOT BZH

Running Zephyr and Linux on the same SoC: making both worlds live together !

Julien Massot

IoT.bzh at a glance

Our location
Brittany



European **C**yber**S**ecurity **O**rganisation Cyber Valleys mapping

Our 30-year OS background
Wind River (1990) - Intel (2009) - IoT.bzh (2015)

VxWorks®
WIND RIVER
Real Time OS leader

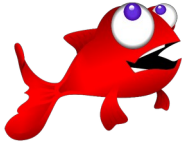
TIZEN
SAMSUNG
n°1 OS in TV market
Made by Intel in Brittany

AUTOMOTIVE GRADE LINUX
1st tech contributor
2016-2020
(+50 % of commits)

THE LINUX FOUNDATION
TOYOTA
mazda
SUZUKI
NISSAN
DENSO
Panasonic
RENESAS

Our team
~30 engineers

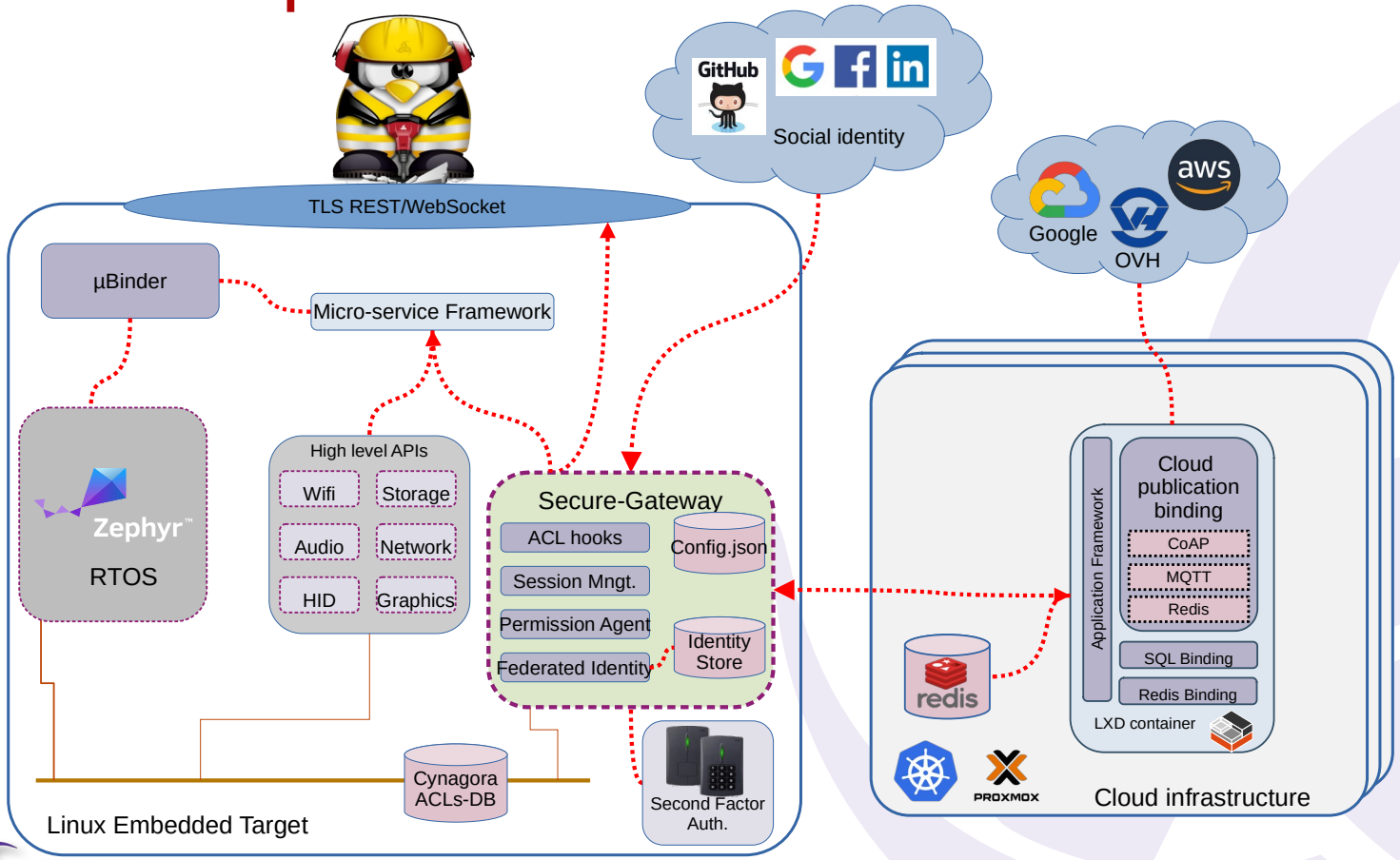
Our product



IOT BZH redpesk®

redpesk® is a pre-integrated « ready-to-use » CI/CD SW factory generating a custom & secured OS, with Long Term Support

redpesk®: from sensor to cloud



Agenda

- What is a co-processor and Heterogeneous Multiprocessing
- Reasons for running Zephyr and Linux on the same SoC
- Introducing Zephyr in Linux environment
- Communicating between Zephyr and Linux applications
- Manage your co-processor from Linux
- Demo

Co-processor and Heterogenous multiprocessing

In the same SoC we may have multiples CPU and architectures

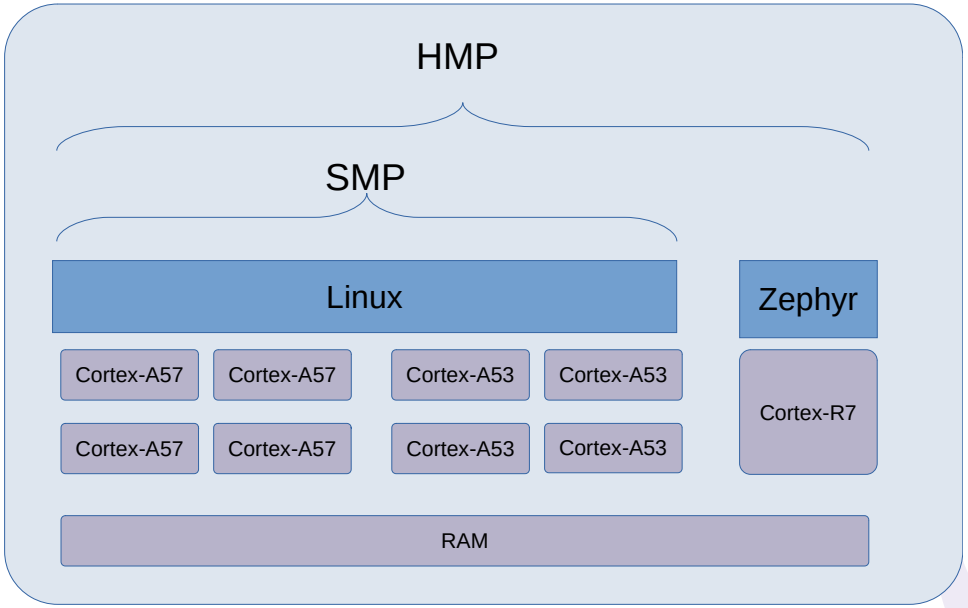
Application Core e.g: ARM Cortex A53, Cortex A7

Digital Signal Processing: Hexagon, Xtensa

Low Power and Realtime processor: ARM Cortex M4, Cortex r7

Neural Processing Unit

Symmetric multiprocessing and Heterogenous multiprocessing



Symmetric multiprocessing:
Run an OS on multiple processors, of a compatible architecture.
E.g: Linux on BigLittle ARMv8-A A57/A53

Heterogenous multiprocessing:
Run several OS on multiple processors which can be of different architectures.
E.g: Linux on ARMv8-A A57/A53
Zephyr on ARMv7-R Cortex-R7

Linux alone may not be enough

Linux is a rich operating system but implementation can be challenging when:

- System should met hard deadline: realtime cases

- Should be wake up on sensor

- Should be certified for safety purpose

Sometimes other processors can also be more efficients for specifics tasks:

video encoding, audio processing, neural network computing..

Embedded applications requirements

- Real Time
- Performance
- Low power consumption
- Application startup time
- Secure application
- Safety certified software

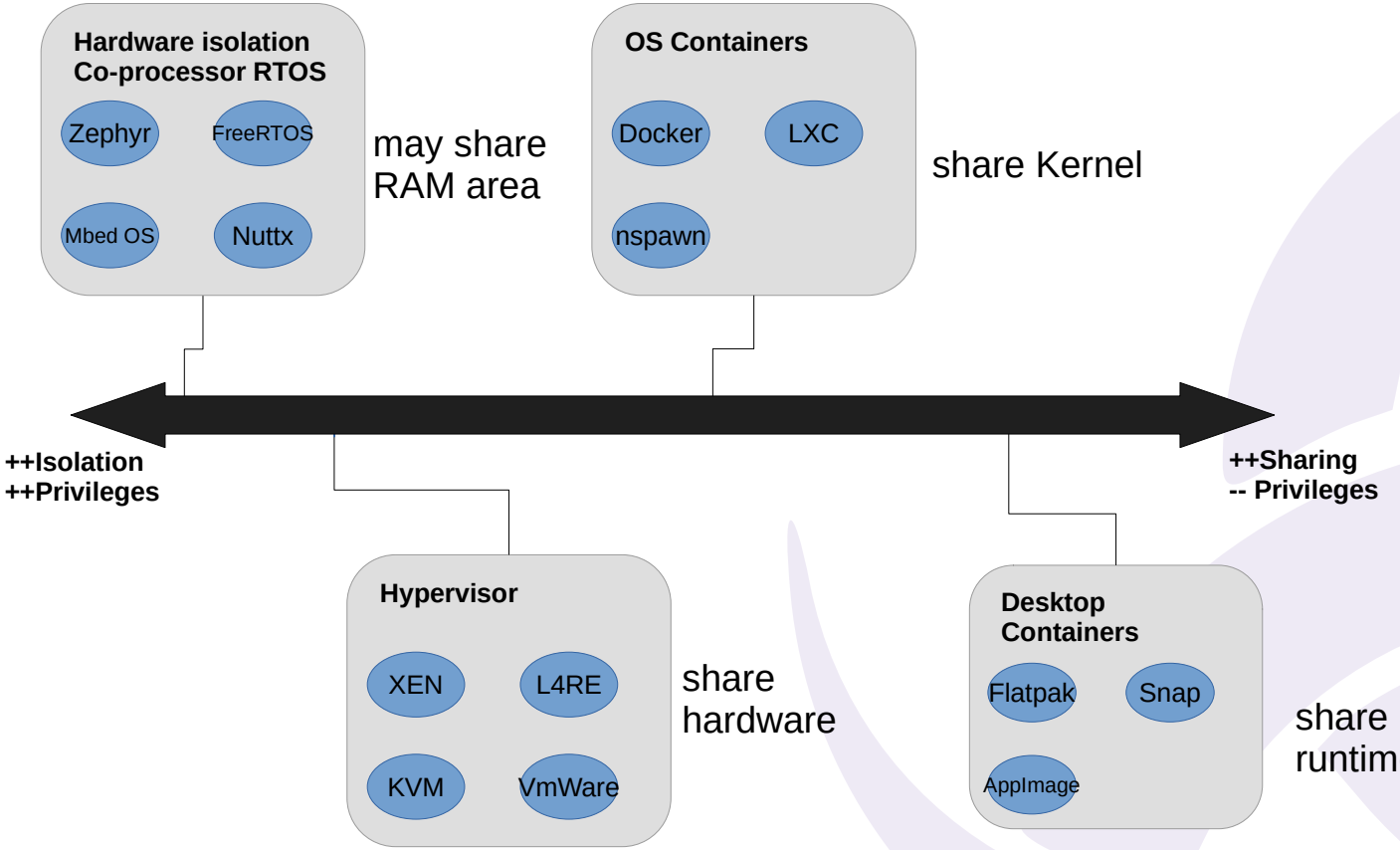
Embedded applications with heterogenous multiprocessing

- Can isolate software function, make it more predictable
- Can aggregate sensors, extract high level data
- Offload recuring tasks
- Reduce the need of external components

But can be challenging:

- Need to split or share resources (RAM, Devices, Storage..)
- Add another system to maintain
- Share messages between both systems

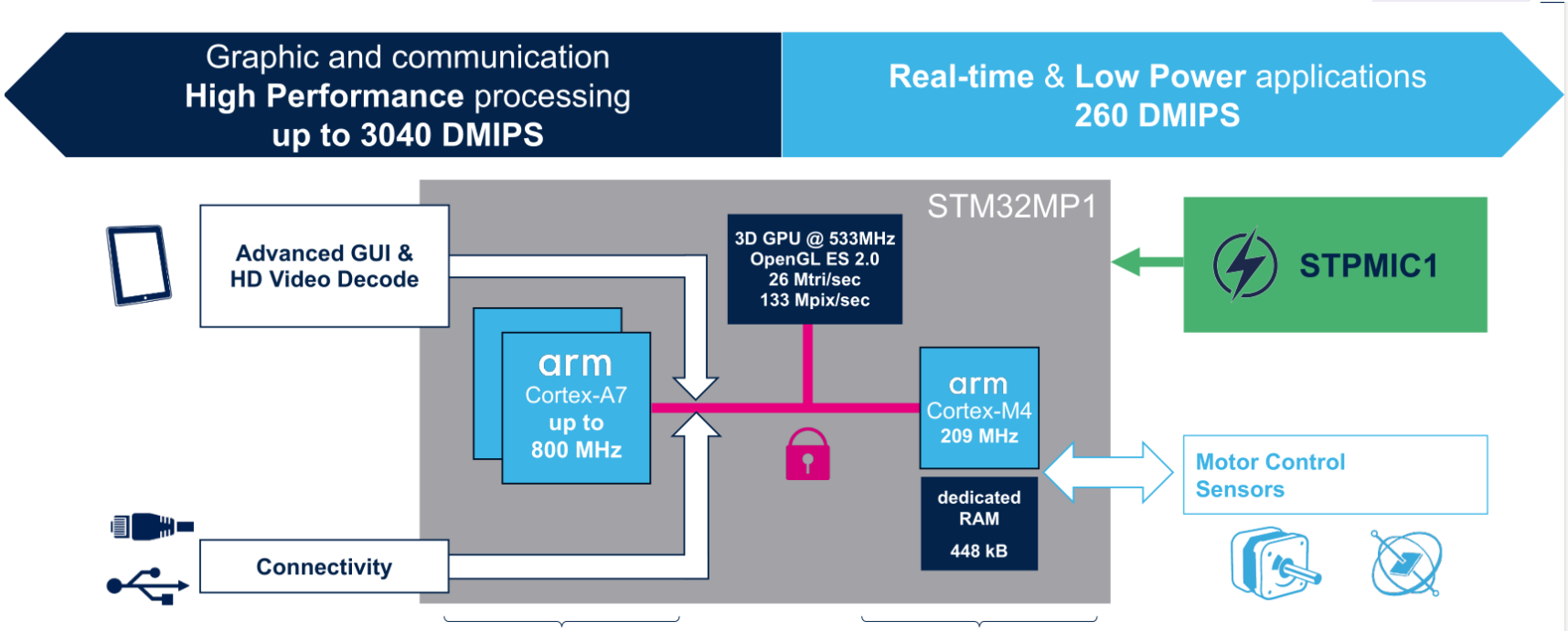
Software isolation



Some embedded platforms with HMP

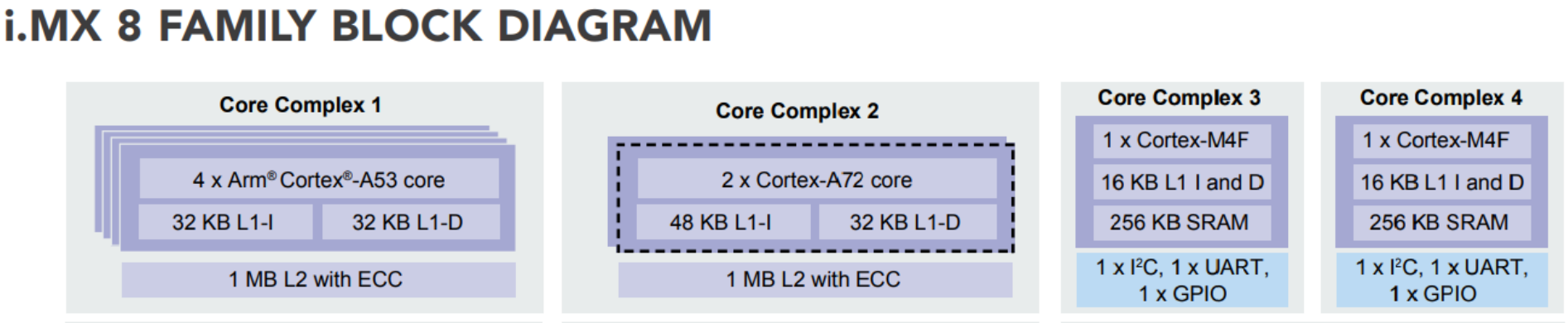
STM32MP1 SoC series

Cortex A7 + Cortex M4



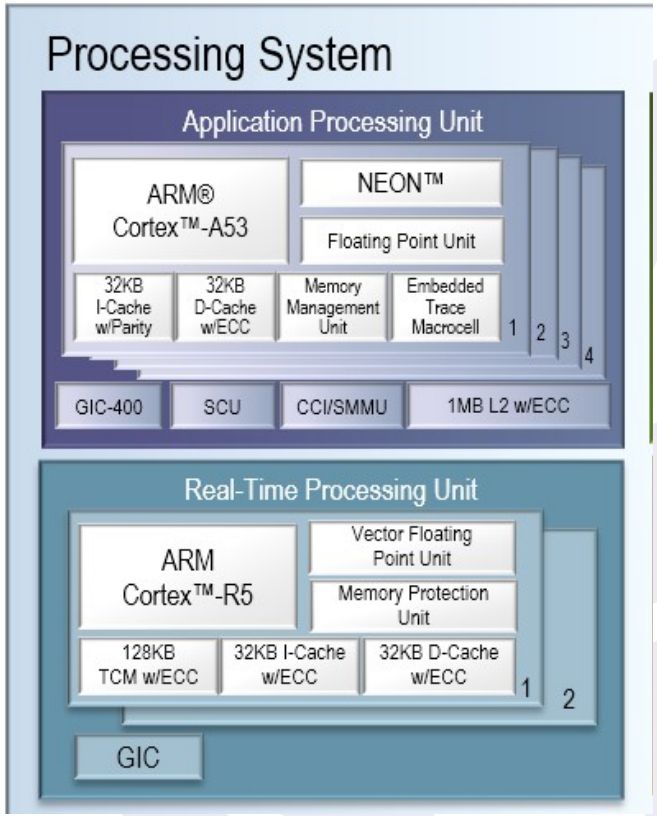
IMX8 SoC series

Cortex A72/A53 + Cortex M4F



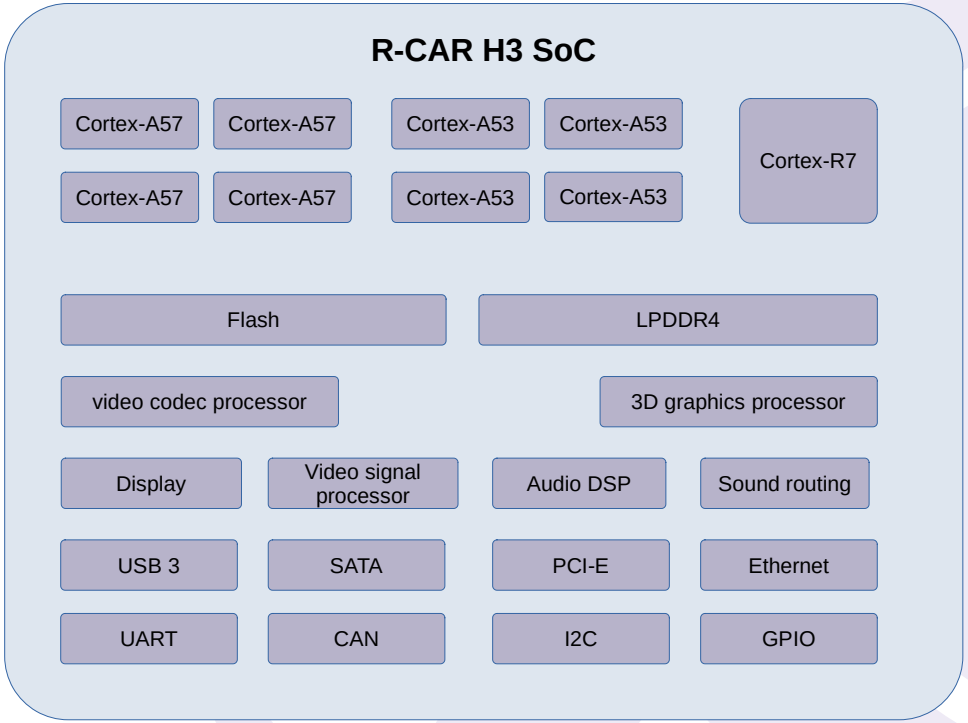
Xilinx Zync ultrascale

Cortex A53 + Cortex r5



Renesas R-CAR H3 SoC

Cortex A57/A53 + Cortex R7



Renesas Cortex-R7

- General purpose microcontroller
- Not already affected to audio processing or video compression...
- Can access any memory mapped devices (CAN, I2C, ..)
- Armv7 800MHz
- Dual core lockstep, suitable for safety
- GIC interrupt controller
- Enough to run complex tasks, sounds cool !

The Zephyr project

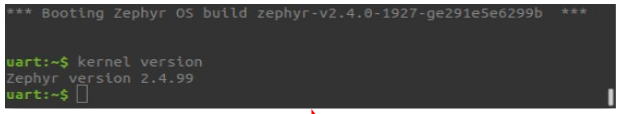
- Linux Foundation project
- Small kernel
- Many protocol stacks (IPv4, IPv6, BLE, CAN)
- Auditable code base developed with a goal of safety certifications (IEC 61508)
- Long term support (LTS) with security updates
- Apache 2.0 open source license
- Many sponsors and contributors
  Facebook, Google, Intel, Nordic Semiconductor, NXP, Linaro

Run your application on Zephyr

- Dedicate hardware devices and resources to Zephyr
- Port your platform, if not already supported
- Write your board dts
- Write or reuse your required driver serial, CAN, I2C
- Re-use stack and OS services IPv6 BLE, CANOpen, MQTT
- Port your application

Here we are

but too much isolated !



console

Critical application

SPI

I2C

PWM

UART

GPIO

CAN

Zephyr™

Cortex R7

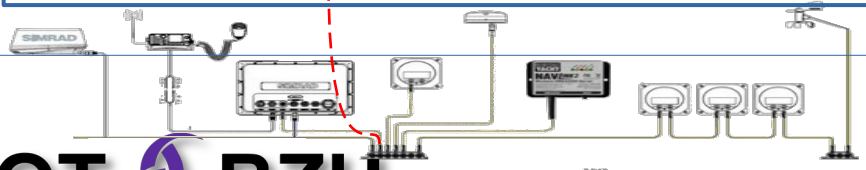
App Framework

Signal Composer Binding

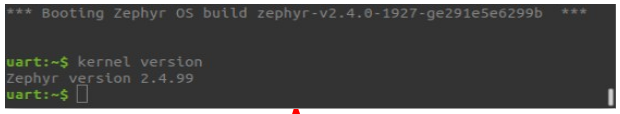
Application Custom Logic

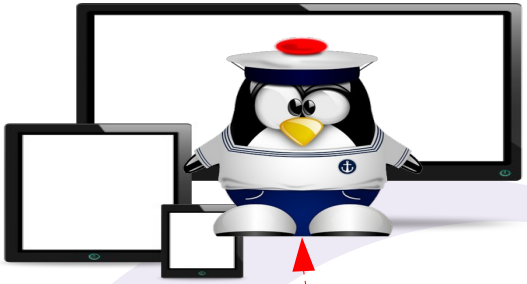
Linux Cortex A57/A53

Renesas Gen3 SoC



Let's add communication





console

Realtime & Secure Area

Critical application

OpenAMP

libmetal

Mailbox

SPI

I2C

PWM

UART

GPIO

CAN

Zephyr™

Cortex R7

rpmsg

interrupts

App Framework

Remoteproc

Shared Memory

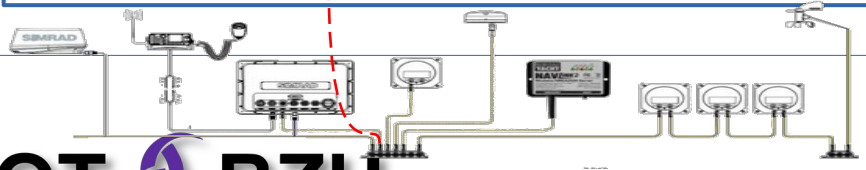
Mailbox
Interprocessor Interrupts

Signal Composer
Binding

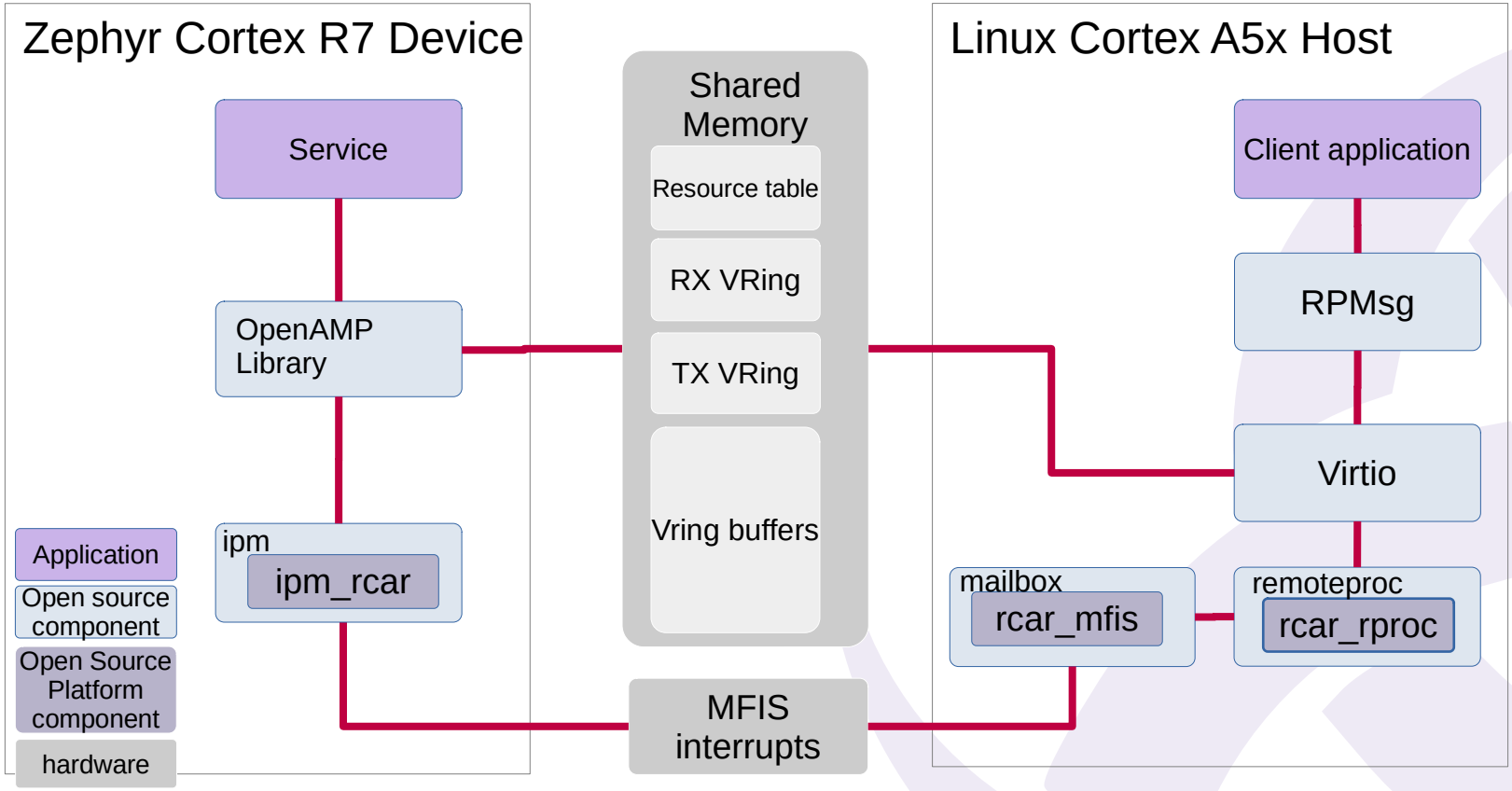
Application
Custom Logic

Linux Cortex A57/A53

Renesas Gen3 SoC



OpenAMP details



Linux Kernel Configuration

CONFIG_REMOTEPROC=y --> Remote proc subsystem for life cycle management

CONFIG_RCAR_RPROC=y --> Your hardware remote proc driver here for Renesas R-Car

CONFIG_MAILBOX=y --> Mailbox subsystem to send and receive interrupts

CONFIG_RCAR_IPCC=y --> Your hardware mailbox driver

CONFIG_RPMSG=y --> Remote processor messaging subsystem

CONFIG_RPMSG_VIRTIO=y --> Your RPMsg transport, OpenAMP rely on virtio

Linux Kernel Device Tree for H3ULCB board

```
reserved-memory {
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;

        cr7_ram: cr7_ram@0x40040000 {
                no-map;
                reg = <0x0 0x40040000 0x0 0x1fc0000>;
        };

        vdev0vring0: vdev0vring0@42000000 {
                no-map;
                reg = <0x0 0x42000000 0x0 0x1000>;
        };

        vdev0vring1: vdev0vring1@42010000 {
                no-map;
                reg = <0x0 0x42010000 0x0 0x1000>;
        };

        vdev0buffer: vdev0buffer@42020000 {
                no-map;
                reg = <0x0 0x42020000 0x0 0x4000>;
        };
};
```

Reserve some memory so that Linux will not use these RAM area.

The remote processor RAM area to run the firmware from.

Area for Tx virtqueue.

Area for Rx virtqueue.

Area for the differents messages.

Zephyr Configuration

CONFIG_IPM=y --> Interrupt-based inter-processor mailboxes

CONFIG_IPM_RCAR=y --> Your IPM driver here for Renesas R-Car

CONFIG_OPENAMP=y --> Enable OpenAMP IPC library

CONFIG_OPENAMP_SLAVE=y --> Virtqueue are initialized by the host (Linux)

CONFIG_OPENAMP_RSC_TABLE=y --> Read virtqueue configuration from the resource table that the host have to fill before starting communication.

Zephyr Device Tree for H3ULCB board

```
model = "Renesas h3ulcb board";
compatible = "renesas,h3ulcb-cr7";

chosen {
        zephyr,sram = &sram0;
        zephyr,ipc_shm = &sram_shm;
        zephyr,ipc = &mfis;
        zephyr,can-primary = &can0;
        zephyr,console = &scif1;
        zephyr,shell-uart = &scif1;
};

sram_shm: memory@42000000 {
        compatible = "mmio-sram";
        reg = <0x42000000 0x400000>;
};
```

zephyr,ipc_shm: the memory node used for openamp.

zephyr,ipc: the node of the interprocessor interrupt hardware module.

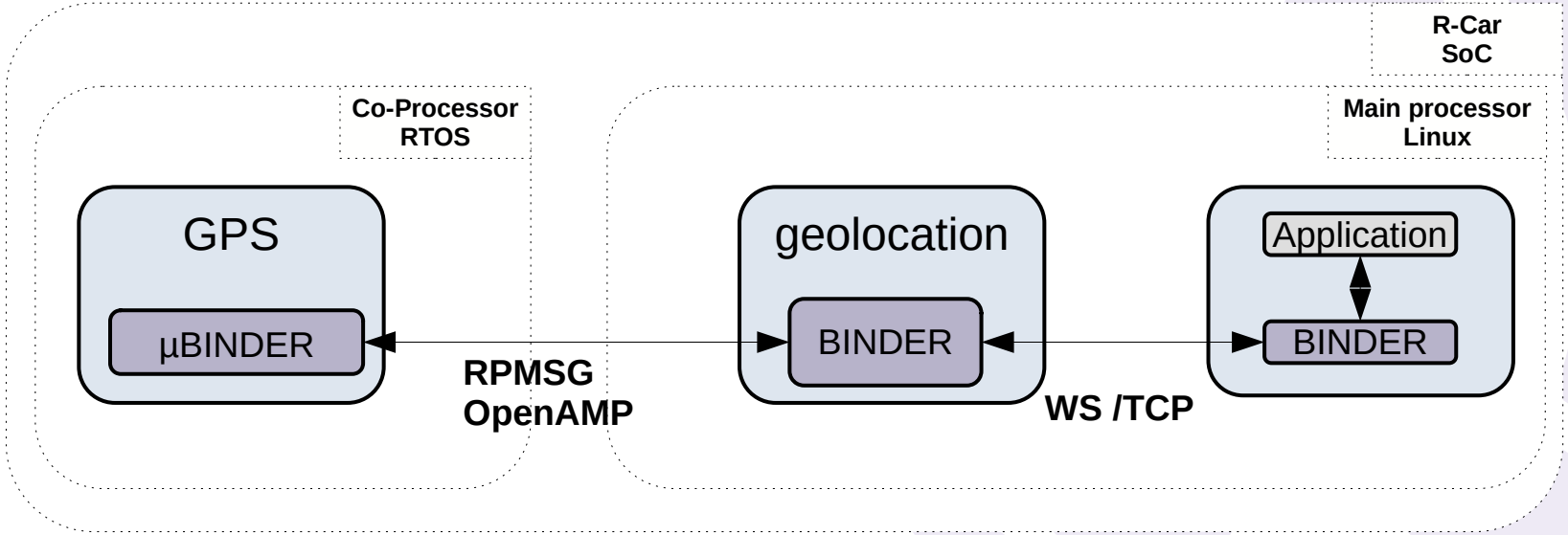
sram_shm: RAM area for buffers and virtqueues, details will be given by Linux in the resource table.

make API call over OpenAMP

Could be more convenient to access API and not buffer.

OpenAMP is just a transport, that can send or receive datas.

Use OpenAMP as a transport for AFB Binder as of TCP or websocket (In progress)



Co-processor life cycle management

Linux Remoteproc framework allows to:

- Let Linux handle the co-processor LCM
- Load a specific firmware for the filesystem
- Update the firmware over the air
- Start stop a remote processor / Attach to a running processor
- Initiate shared memory for communication
- Get debug trace from the debugfs

Playing with remoteproc

- **Specify a firmware to boot**

  $ echo -n "renesas/zephyr.elf" > /sys/class/remoteproc/remoteproc0/firmware
- **Start stop a remote processor**

  $ echo start > /sys/class/remoteproc/remoteproc0/state

  remoteproc remoteproc0: powering up cr7

  remoteproc remoteproc0: Booting fw image renesas/zephyr.elf, size 413184

  remoteproc remoteproc0: remote processor cr7 is now up

  $ echo stop > /sys/class/remoteproc/remoteproc0/state

  remoteproc remoteproc0: stopped remote processor cr7
- **Get the debug output**

  $ cat /sys/kernel/debug/remoteproc/remoteproc0/trace0

  *** Booting Zephyr OS build zephyr-v2.5.0 ***

What you can currently do with Zephyr for Renesas R-Car Gen3

- Get console and shell through UART
- Use GPIO
- Read and write to an I2C device
- Send and receive CAN frames
- Start firmware from Linux
- Communicate with Linux through OpenAMP/RPMsg
- Fetch and compile:

```
$ west init -m git@github.com:iotbzh/zephyr.git --mr renesas-v2.6
$ west sync
$ west build -b rcar_h3ulcb_cr7 zephyr/samples/basic/blinky  --build-dir ulcb-blinky
```

Current work

- Bringing more drivers for Renesas H3ULCB in Zephyr mainline

  https://github.com/zephyrproject-rtos/zephyr/issues/33274

  and add other Renesas R-Car based boards.
- Upstreaming Renesas R-Car Linux remoteproc driver ?
- OpenAMP Integration with AFB binder application framework

Q&A

Lorient Harbour, South Brittany, France

Links

- IoT.bzh:
  - Website: https://iot.bzh/
  - Publications: https://iot.bzh/en/publications
  - Github: https://github.com/iotbzh
  - Renesas Zephyr: https://github.com/iotbzh/zephyr/tree/renesas
- Zephyr:

  - Getting Started https://docs.zephyrproject.org/latest/getting_started/index.html
- OpenAMP/Remoteproc:
  - Linux and Zephyr “Talking” to each other in the same SoC Diego Sueiro
  - Wiki OpenAMP https://github.com/OpenAMP/open-amp/wiki
  - STM32MPU wiki: https://wiki.st.com/stm32mpu/wiki/Category:How_to
  - Remote proc subsystem https://www.kernel.org/doc/html/latest/staging/remoteproc.html