Embedded Linux Conference
Europe

NuttX for Embedded Linux Developers

Masayuki.Ishikawa@sony.com
Senior Software Engineer
Sony Home Entertainment & Sound Products Inc.

THE LINUX FOUNDATION

Agenda

- Sony audio products based on NuttX
- Feasibility studies for NuttX enhancements
  - SMP
  - Networking
  - Porting the AVS device SDK
- Support for Sony Spresense board
- NuttX workshop

About Me

Senior Software Engineer
at Sony Home Entertainment & Sound Products Inc.

Technical background

- 3D graphics, home networking, Internet-to-Home, Embedded Systems

Product development

- Portable Media Player (Linux/Android)
- Digital voice recorder, music player, head phone (NuttX)

Public talks

- Arm Techcon 2016, ELC2017NA, OpenIoT2018NA, NuttX2019, ELC2019NA

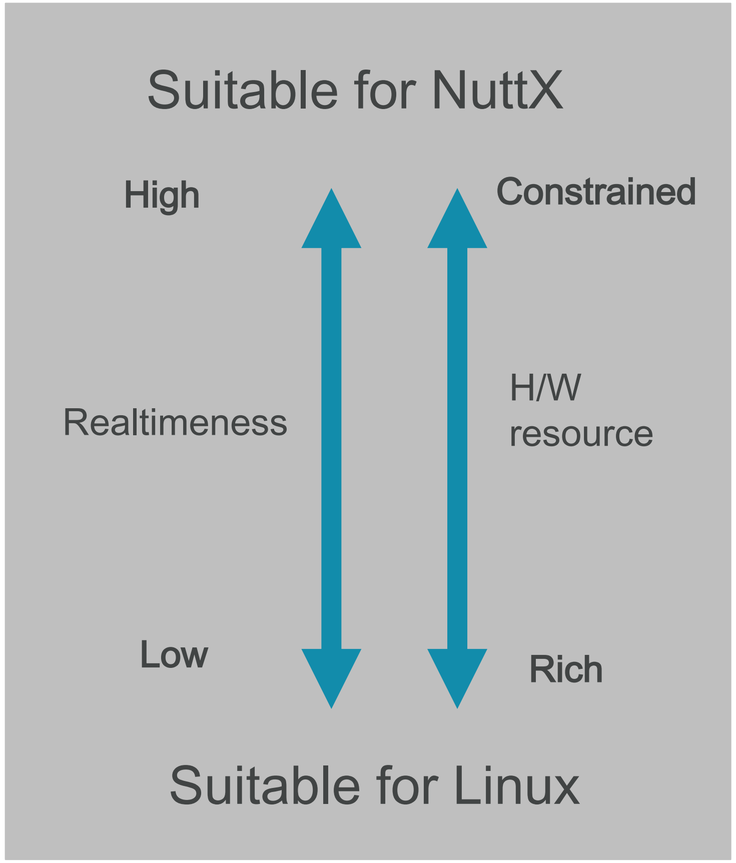
NW-A800

NW-ZX1

ICD-UX560

About NuttX (1/2)

- Gregory Nutt released in 2007 as an open source
- Mostly POSIX-compliant* real-time OS
  - Suitable for robotics ** including Drones.
- From 8bit to 32bit CPU are supported
  - Z80, x86, Arm7/9/11, Cortex-Mx/Ax/Rx, AVR, MIPS, RISC-V, Xtensa,...
  - Many evaluation boards (over 180) are supported
- Small footprint and runtime memory
  - Suitable for resource constraint devices



* Linux is also a mostly POSIX-compliant OS

** NuttX is the default RTOS for ROS2 embedded

About NuttX (2/2)

- Almost all code are written from scratch*
  - Straightforward and consistent without vendor HAL

- Many key features are implemented
  - Virtual file systems, loadable module, tickless, SMP
  - FAT12/16/32, SmartFS, romfs, procfs, NFS (client only)
  - Networking (IPv4,IPv6,UDP,TCP, ...)
  - USB (CDC/ACM, MSC, HID, RNDIS, ...),

- Many example applications are included
  - NuttX shell, webserver and client, telnet daemon,...

* Some drivers are ported from other open source

Activities

- Sony audio products based on NuttX
  - Development history
  - Audio products
  - Reasons for choosing NuttX

- Feasibility studies for NuttX enhancements

Development history*

SONY

- Oct 2013 -
  - Ported NuttX to LC823425 (ARM7)
- Apr 2014 -
  - Ported bluetooth stack to NuttX + QEMU
- Jul 2014 -
  - Ported NuttX to LC823450 (Cortex-M3) FPGA
- Jan 2015 -
  - Migrated to LC823450-ES board
- Sep 2015 -
  - Released the first NuttX-based audio products.
- Oct 2016 -
  - Talked at Arm TechCon 2016, ELC NA 2017*-2019, NuttX 2019

SONY

Developing Audio Products with Cortex-M3/NuttX/C++11

Masayuki.Ishikawa@sony.com
Senior Software Engineer
Sony Video & Sound Products Inc.

Embedded Linux Conference North America
OpenIoTSummit North America

SMP and Networking support on NuttX / LC823450

Masayuki.Ishikawa@sony.com
Koichi.Okamoto@sony.com
Sony Video & Sound Products Inc.

*https://www.youtube.com/watch?v=TjuzH6JthxQ ** https://www.youtube.com/watch?v=T8fLjWyI5nI

NuttX based audio products

FY2015

Voice Recorders

Music Player

FY2016,17

Voice Recorders

Wireless Headphones

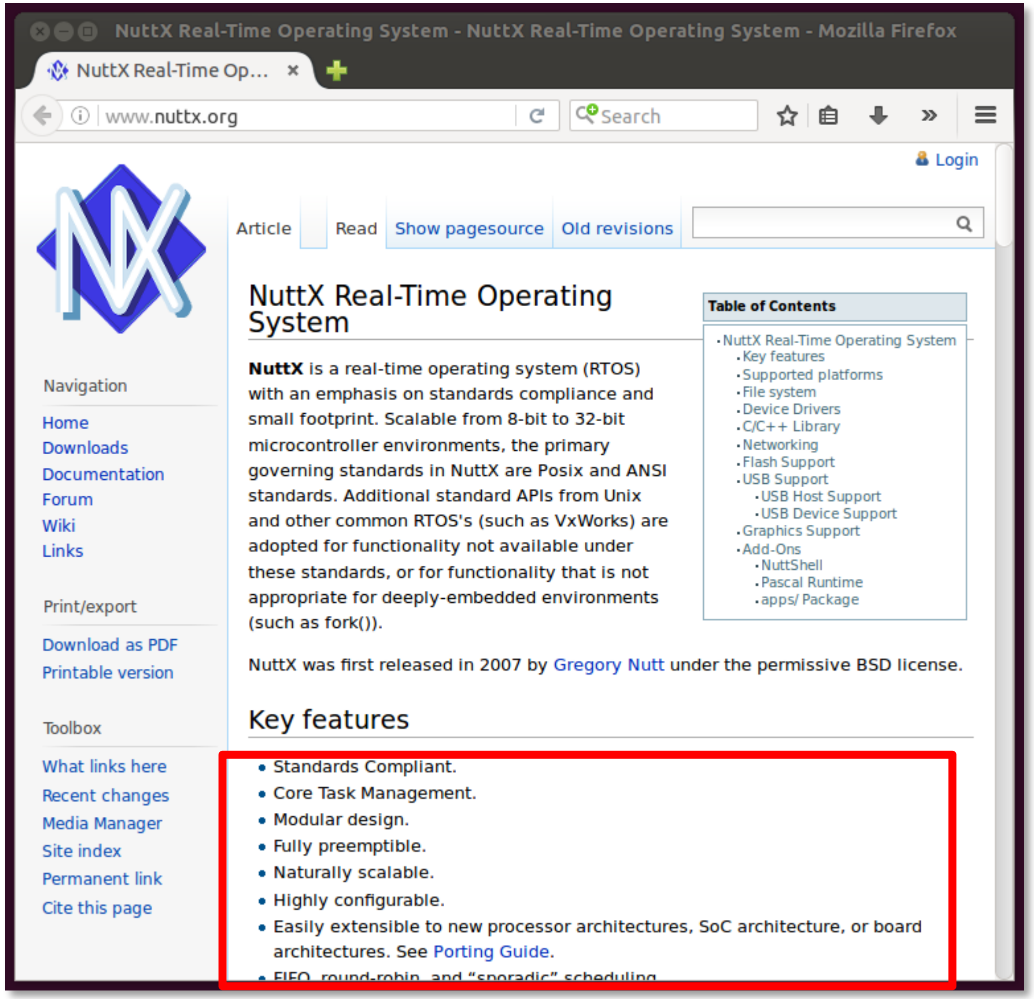
FY2018

Wireless Headphones

PCM Recorders

Why we chose NuttX for audio products

- POSIX and libc are supported
  - Can reuse existing software
  - Can reduce learning costs
- ELF* is supported
  - Can divide a big app into small apps
- Driver framework is supported
  - Helps us implement drivers
- Has Linux-like configuration system
  - Helps us develop multiple products
- Many MCUs and boards are supported
  - Helps us port NuttX to new MCU
- Provided with BSD license



NuttX Real-Time Operating System

NuttX is a real-time operating system (RTOS) with an emphasis on standards compliance and small footprint. Scalable from 8-bit to 32-bit microcontroller environments, the primary governing standards in NuttX are Posix and ANSI standards. Additional standard APIs from Unix and other common RTOS's (such as VxWorks) are adopted for functionality not available under these standards, or for functionality that is not appropriate for deeply-embedded environments (such as fork()).

NuttX was first released in 2007 by Gregory Nutt under the permissive BSD license.

Key features

- Standards Compliant.
- Core Task Management.
- Modular design.
- Fully preemptible.
- Naturally scalable.
- Highly configurable.
- Easily extensible to new processor architectures, SoC architecture, or board architectures. See Porting Guide.
- FIFO, round-robin, and “sporadic” scheduling

From http://www.nuttx.org/

* ELF = Executable and Linking Format

LC823450 Features

SONY

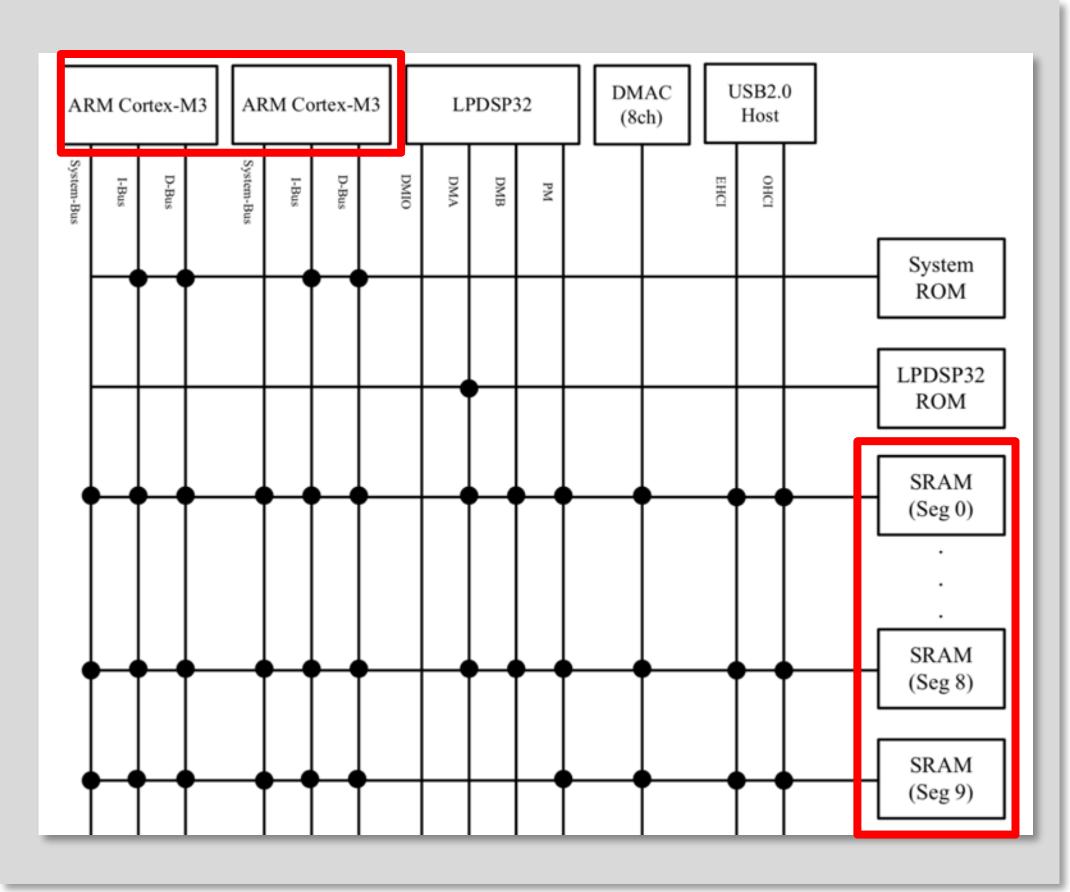
- ARM dual Cortex-M3
- 32bit fixed point, dual-MAC original DSP
- Internal SRAM (1656KB) for ARM and DSP
- I2S I/F with 16/24/32bit, MAX 192kHz (2chx2)
- Hard wired audio functions
  - MP3 encoder and decoder, EQ (6-band equalizer), etc.
- Integrated analog functions
  - Low-power Class D HP amplifier, system PLL
  - Dedicated audio PLL, ADC
- Various interfaces
  - USB2.0 HS device / host (not OTG), eMMC, SD card, SPI, I2C, etc.
- ARM and DSP clock max frequency
  - 160MHz at 1.2V
  - 100MHz at 1.0V

ON Semiconductor LC823450

From http://www.onsemi.com/PowerSolutions/product.do?id=LC823450

Why SMP with LC823450?

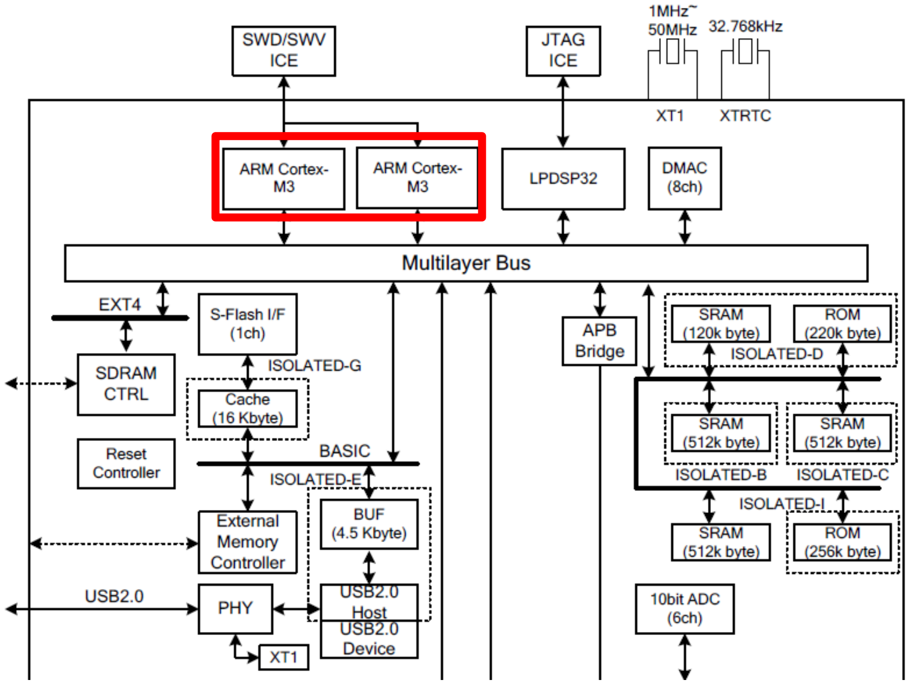
- Motivation
  - Run existing applications in SMP mode
  - Establish knowledge on debugging
  - Confirm effects on memory contention because the processor does not have CPU cache.
  - Confirm power consumption
  - Very challenging theme (because NuttX is not just a scheduler)
- Other reasons…
  - The architecture is much simpler than quad Cortex-A9.
  - Suitable system to understand SMP kernel.

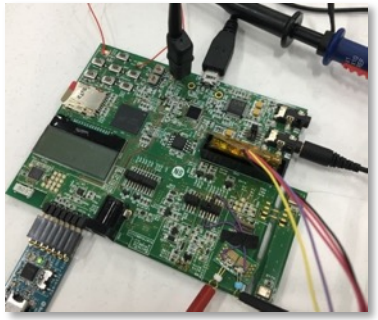


Supporting SMP on the processor

SONY

- Port existing drivers to the latest NuttX
  - UART, Timer, GPIO, DMA, I2C, SPI, LCD
  - eMMC (including boot), SD, USB, ADC, …
- Implement SMP related code
  - lc823450_cpuidlestack.c, lc823450_cpuindex.c
  - lc823450_cpupause.c, lc823450_cpustart.c, lc823450_testset.c (NOTE: H/W Mutex is used instead of ldex, strex)
- Performance improvement
  - Introduced spin_lock_irqsave(), spin_unlock_irqrstore()
  - Applied APIs inside the driver code.
  - Up to 20% performance improvement achieved



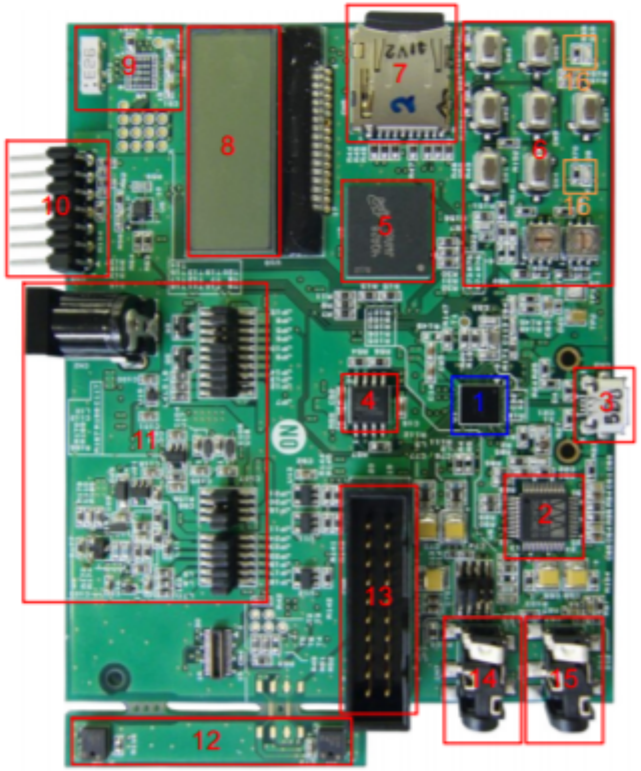


Feasibility studies for networking

- Adding Networking
- NuttX networking features
- USB RNDIS and Bluetooth PAN

Adding Networking

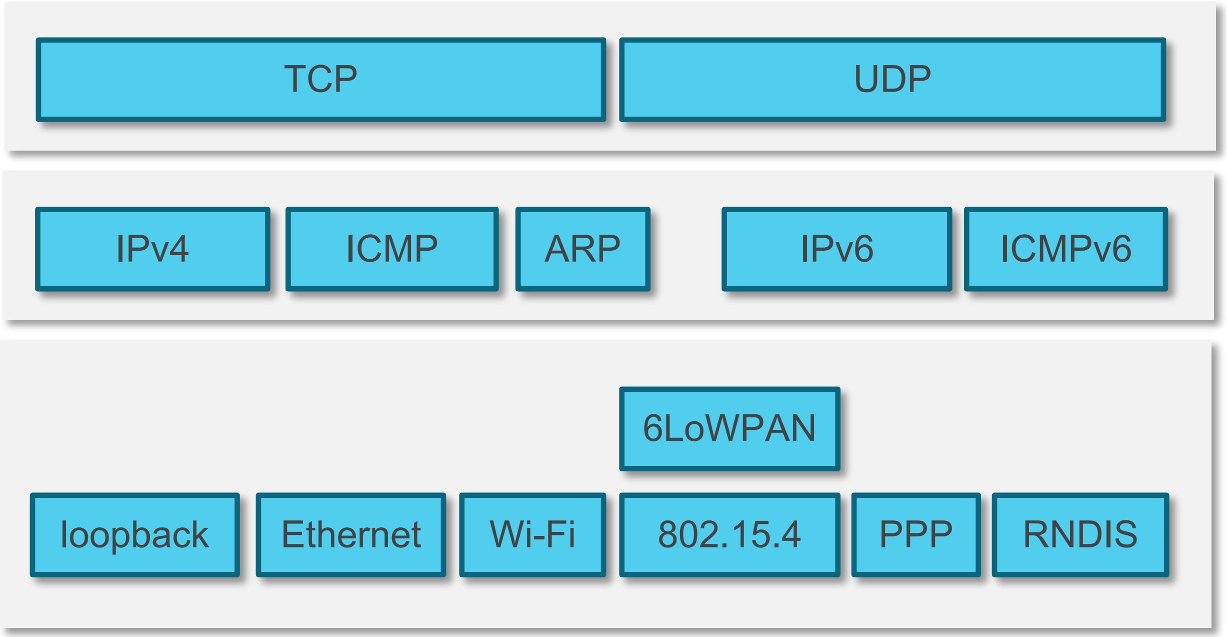
- Motivation
  - Confirm NuttX network stack feasibility
    - IPv4, IPv6, ICMP, UDP, TCP, …
  - Run the network stack with minimum efforts. (We already have a USB driver for LC823450)
  - Audio streaming (PCM and MP3)
  - Run the network stack in SMP mode
  - Do various tests via telnet



LC823450XGEVK board

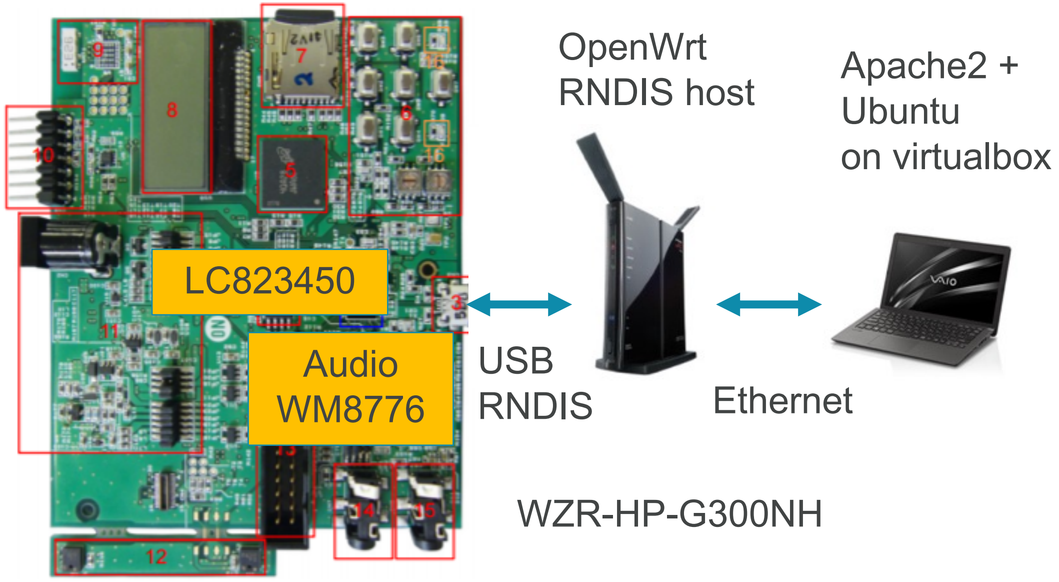
NuttX networking features

- Ethernet and IEEE 802.11 Full MAC
- 6LoWPAN for radio network drivers (IEEE 802.15.4 MAC)
- USB RNDIS (since 7.23), CDC-ECM (since 7.26)
- SLIP, TUN/PPP, local loopback devices
- IPv4, IPv6, TCP, UDP, ARP, ICMP, ICMPv6, IGMPv2
- IP forwarding
- BSD compatible socket layer
- DNS name resolution / NetDB
- User socket (listen/accept are supported in 7.26)
- Bluetooth socket



PCM audio streaming via RNDIS

- Fix RNDIS driver for NuttX
  - Fix data corruption
  - Add USB high speed mode support
- Receive window control has been added
  - Needs more improvement
- Modify nxplayer to support HTTP streaming
  - Currently only WAV format is supported.
- Still testing with SMP kernel
  - In various conditions (clock speed, network traffic, etc)

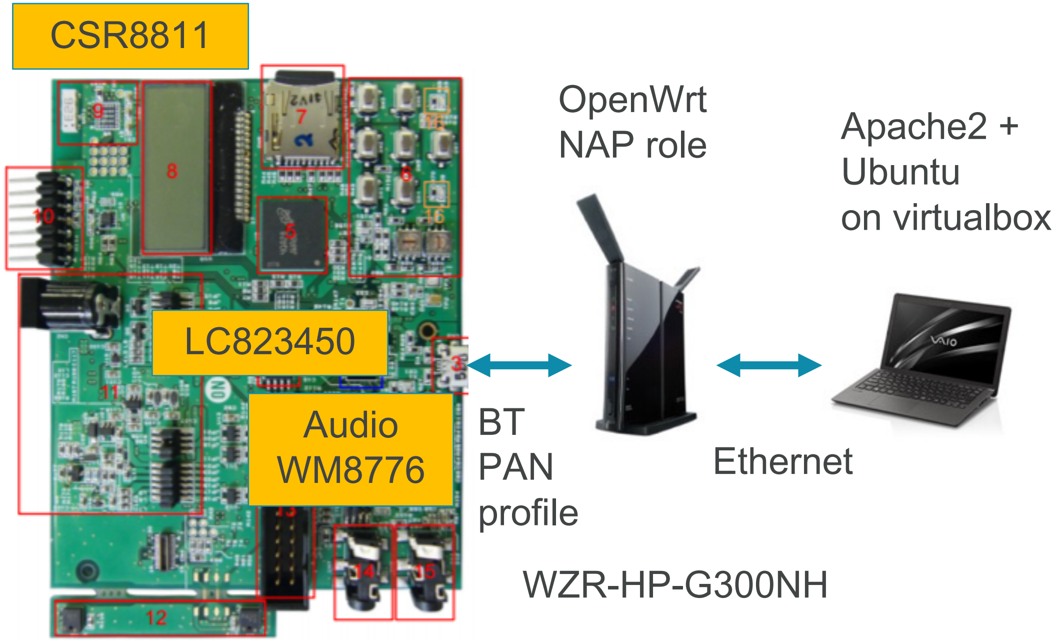


PCM audio streaming demo environment

MP3 audio streaming via Bluetooth

SONY

- Port the BTstack* by Bluekitchen to NuttX
  - Based on posix-h4** with H/W flow control
  - UART speed : 921600 baud
  - Tested with iOS/Android/macOS/OpenWrt
  - Free for non-commercial use
- Add TAP mode to the NuttX tun driver
  - TAP mode is used for network bridge
  - NOTE: TUN mode is used for network routing
- Add H/W MP3 decoder to lc823450_i2s.c
- HCI_RESET issue in SMP mode
  - CSR's mode change with HCI_RESET is tricky
  - Still unstable in SMP mode

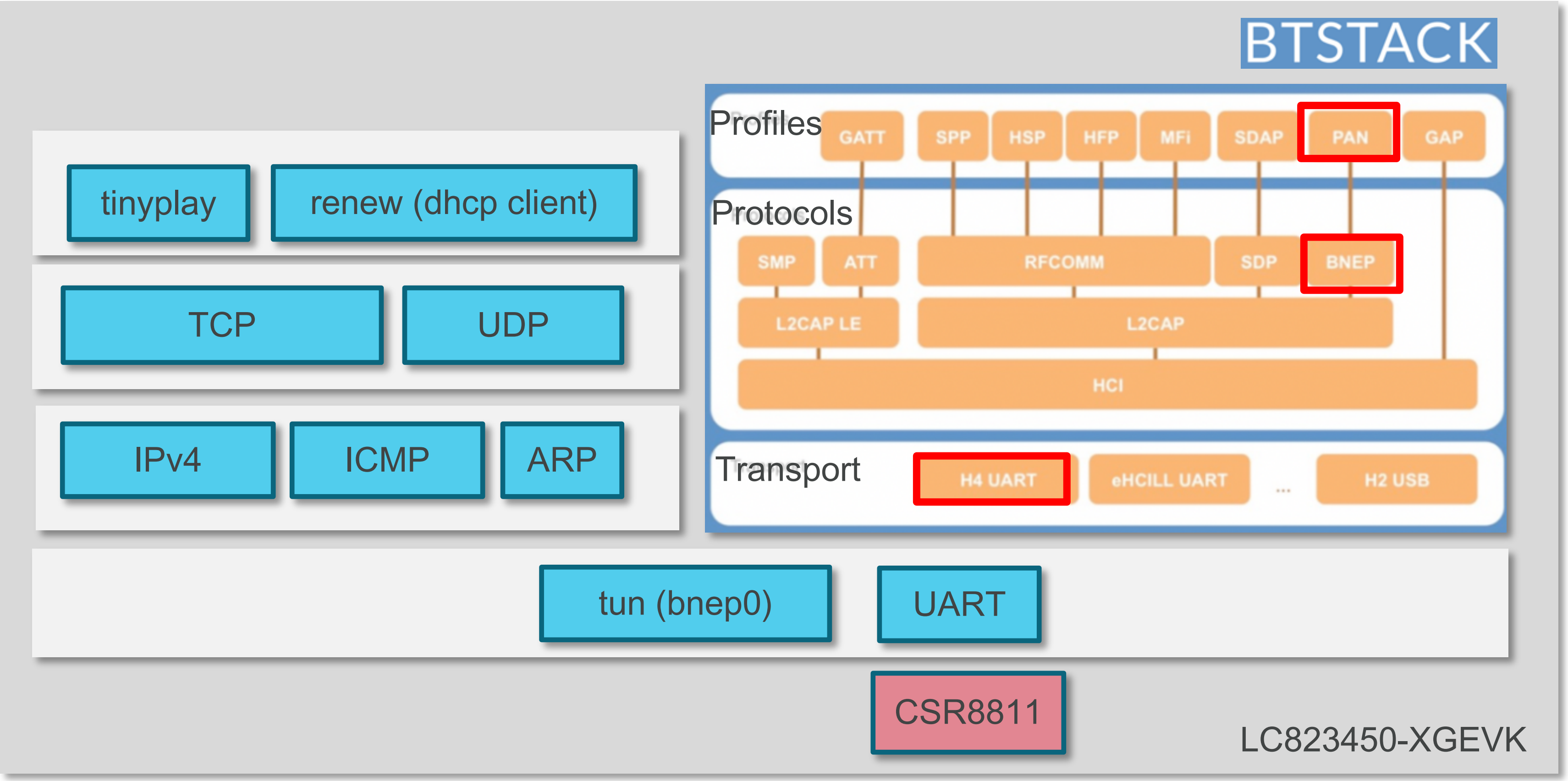


MP3 audio streaming demo environment

* https://bluekitchen-gmbh.com/

** We can use posix-h5 (3-wire protocol) as well. However, it has performance drawbacks.

Running the BTstack on NuttX



BT
PAN
profile

OpenWrt
PAN-NAP* role
DHCP server

WZR-HP-G300NH

*PAN: Personal Area Network
*BNEP: Bluetooth Network Encapsulation Protocol

*NAP: Network Access Point

Porting the AVS* device SDK to NuttX

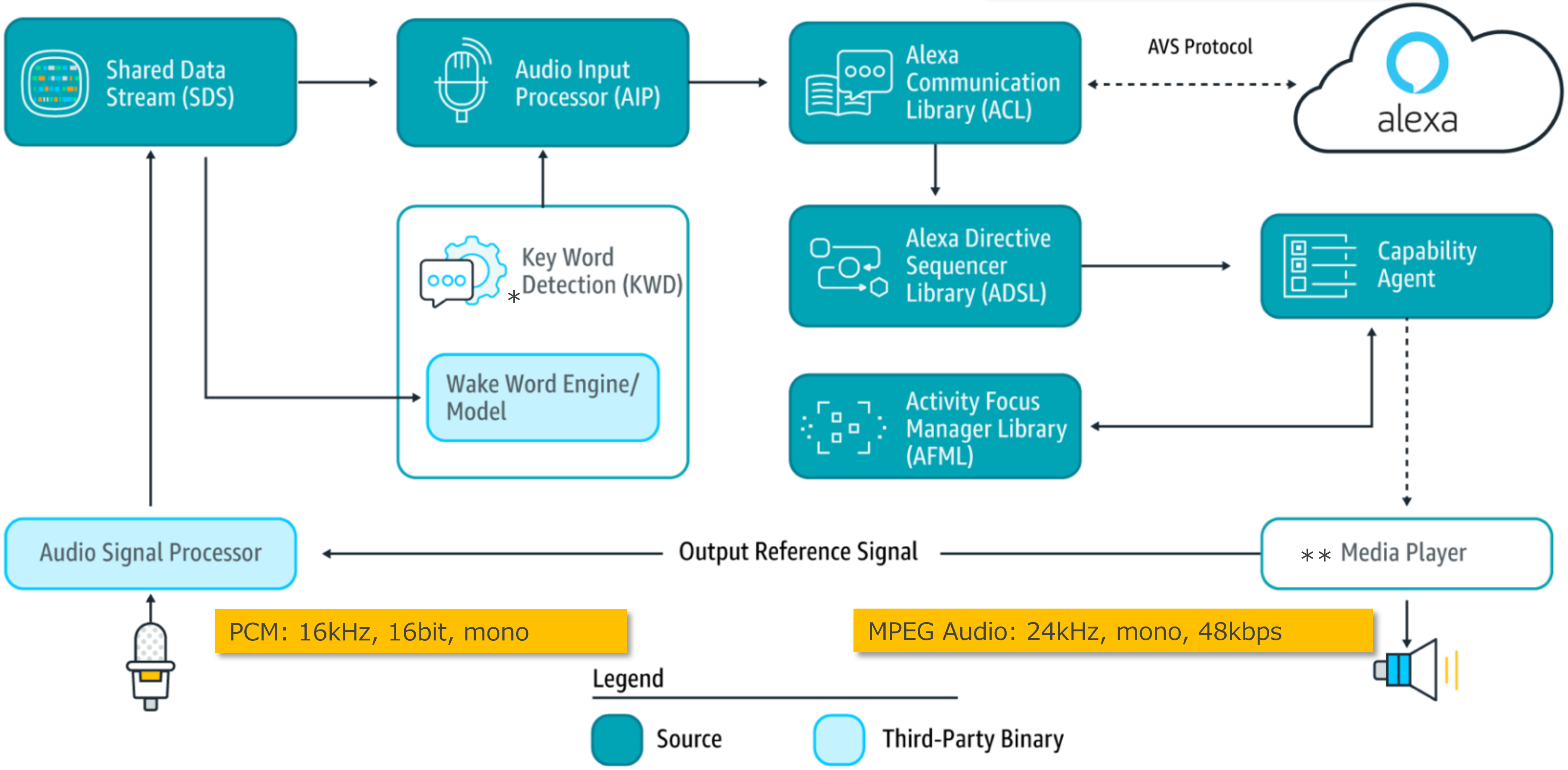
- Motivation
  - Confirm OSS portability with large software systems

- Approaches
  - Build the AVS device SDK on Linux
  - Run the AVS device SDK on Linux
  - Port OSS components such as mbedTLS to NuttX
  - Implement missing components such as MediaPlayer with minimal efforts
  - Reduce runtime memory on NuttX

* AVS stands for Alexa Voice Service

Inside the AVS device SDK

HTTP2 over TLS
ECDHE-RSA-AES256-GCM-SHA384



PCM: 16kHz, 16bit, mono

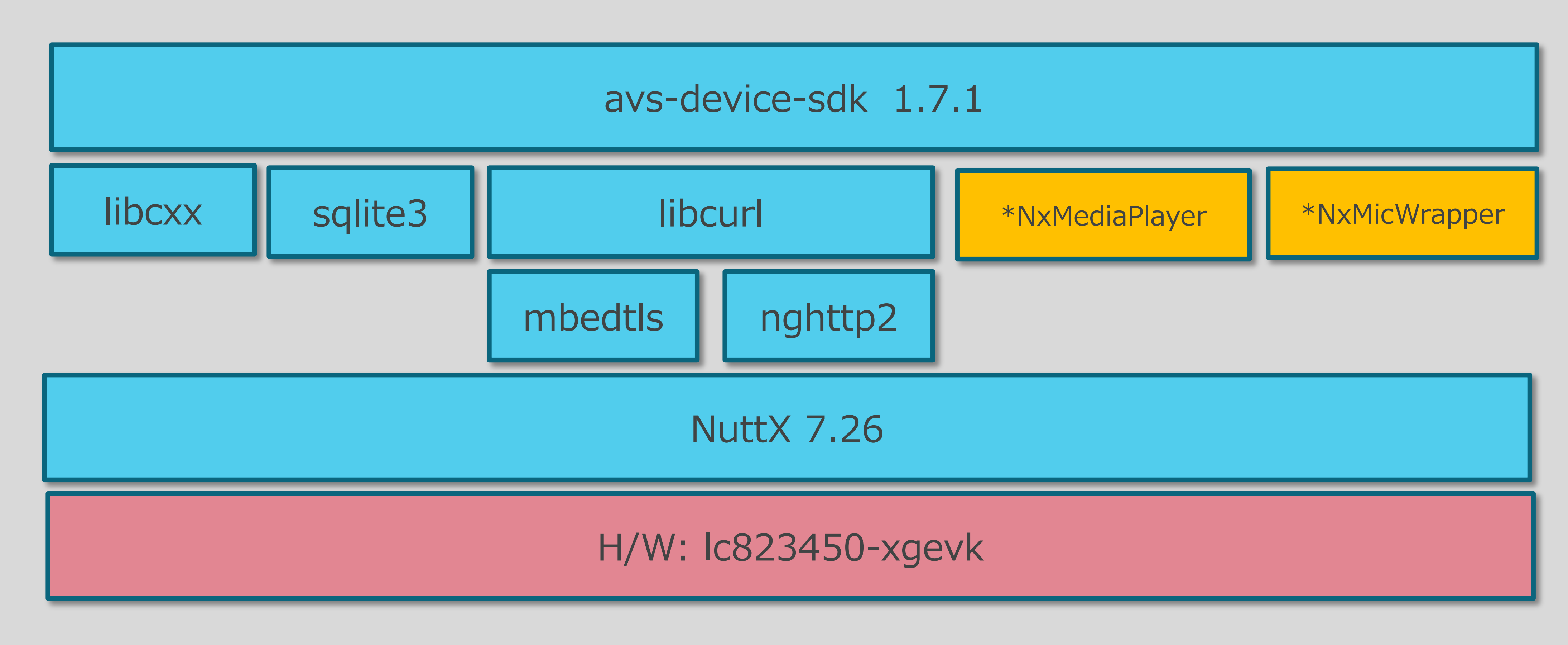
MPEG Audio: 24kHz, mono, 48kbps

* Push-to-talk was used to test avs device sdk inst4ead of Key Word Detection

** Source code for GStreamer and AndroidSLES are available

Software stack

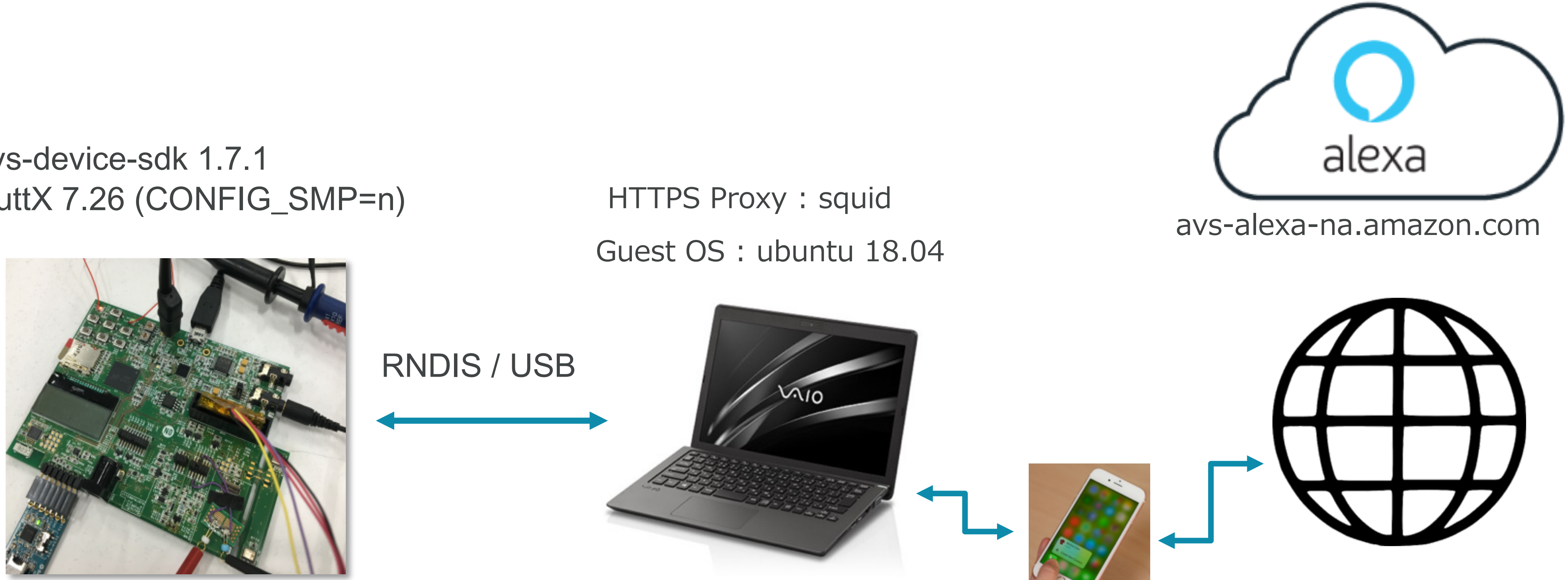
SONY



Total code size is about 3.6MB

*NxMediaPlayer and NxMicWrapper were newly implemented for LC823450
On Linux, GStreamer and PortAudio are used to playback and capture audio

Network topology



avs-device-sdk 1.7.1
NuttX 7.26 (CONFIG_SMP=n)

LC823450XGEVK

RNDIS / USB

HTTPS Proxy : squid

Guest OS : ubuntu 18.04

＊ VAIO S11 + VirtualBox
Host OS : Windows 8.1

iPhone 6S

avs-alexa-na.amazon.com

Internet

* MacBook Pro + VMWare Fusion can be used as well

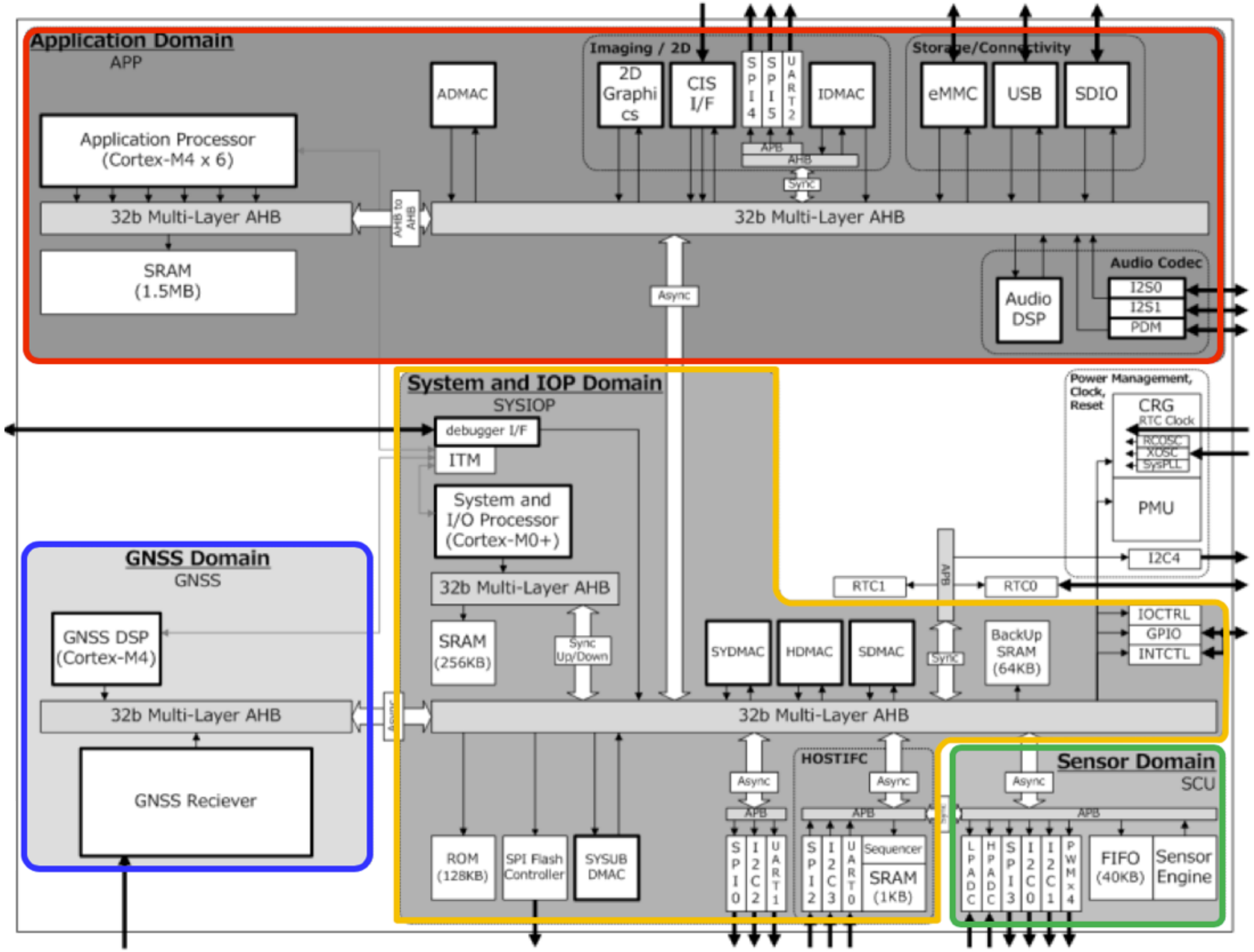
Activities

- Spresense overview
- Spresense SDK
- Upstreaming status
- Working with Wi-Fi

Spresense overview

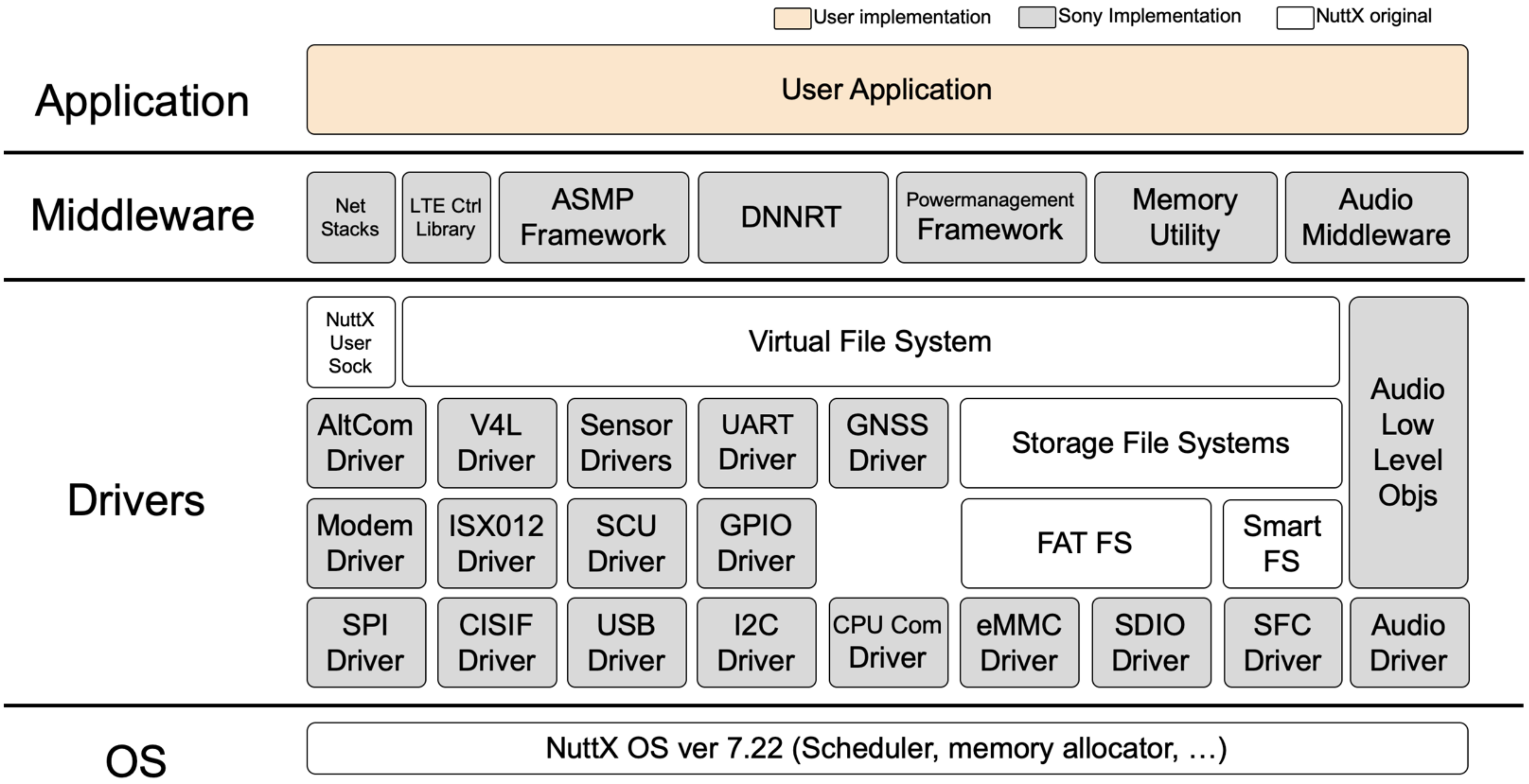
SONY

- Application Domain
  - Cortex-M4F x6 + SRAM 1.5MB
  - eMMC/USB/SDIO
- System and IOP Domain
  - System management
  - Power domain control
- Sensor Domain
  - Sensor Engine + FIFO (40KB)
- GNSS Domain
  - Independent GNSS positioning



Spresense (cxd5602) is here (size 6.5x6.5mm)

Spresense SDK*



▨ User implementation ▨ Sony Implementation □ NuttX original

Application

User Application

Middleware

Net Stacks

LTE Ctrl Library

ASMP Framework

DNNRT

Powermanagement Framework

Memory Utility

Audio Middleware

Drivers

NuttX User Sock

Virtual File System

Audio Low Level Objs

AltCom Driver

V4L Driver

Sensor Drivers

UART Driver

GNSS Driver

Storage File Systems

Modem Driver

ISX012 Driver

SCU Driver

GPIO Driver

FAT FS

Smart FS

SPI Driver

CISIF Driver

USB Driver

I2C Driver

CPU Com Driver

eMMC Driver

SDIO Driver

SFC Driver

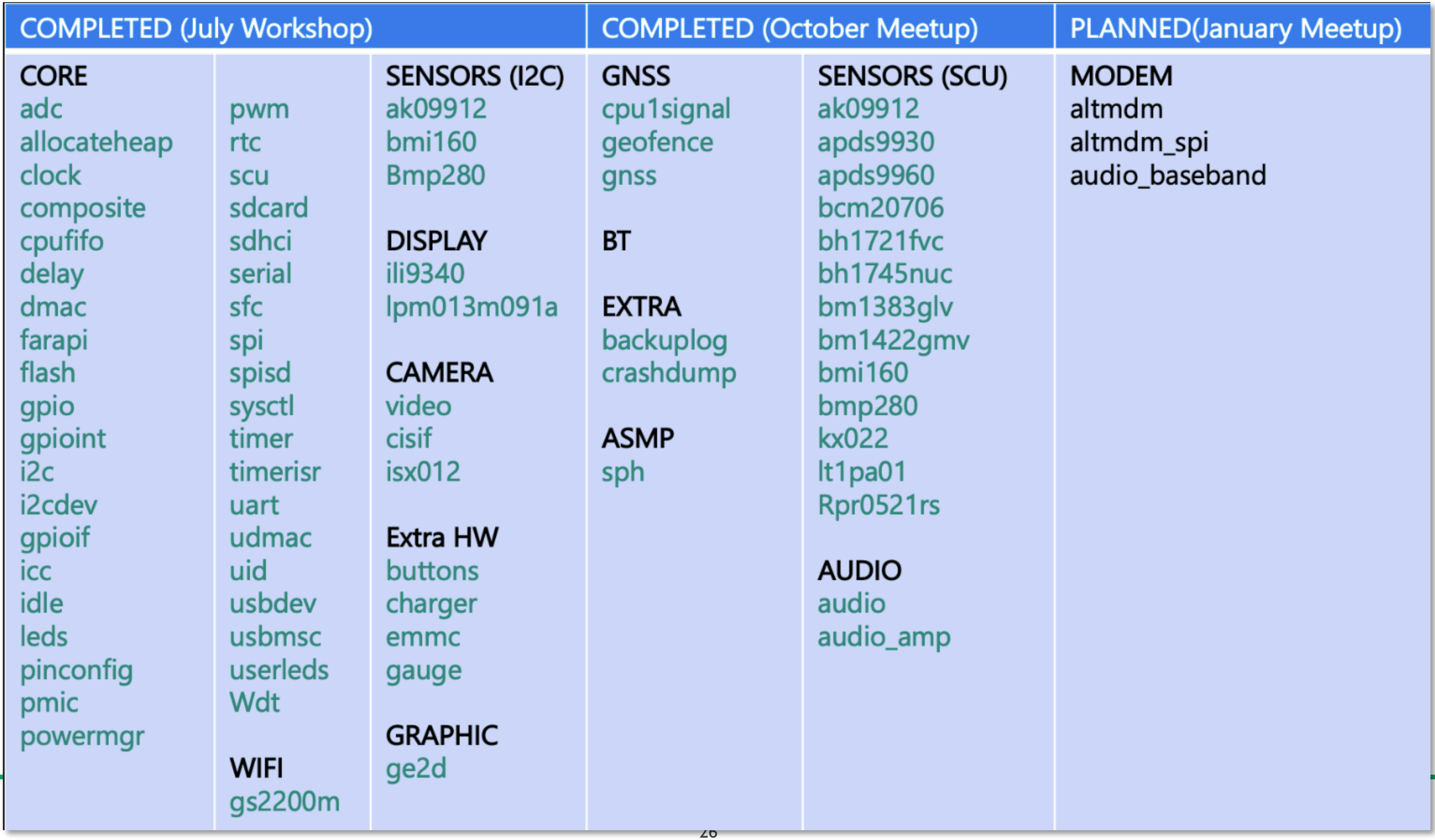
Audio Driver

OS

NuttX OS ver 7.22 (Scheduler, memory allocator, …)

* code is available at https://github.com/sonydevworld/spresense

Upstreaming status (as of Oct/12/2019)



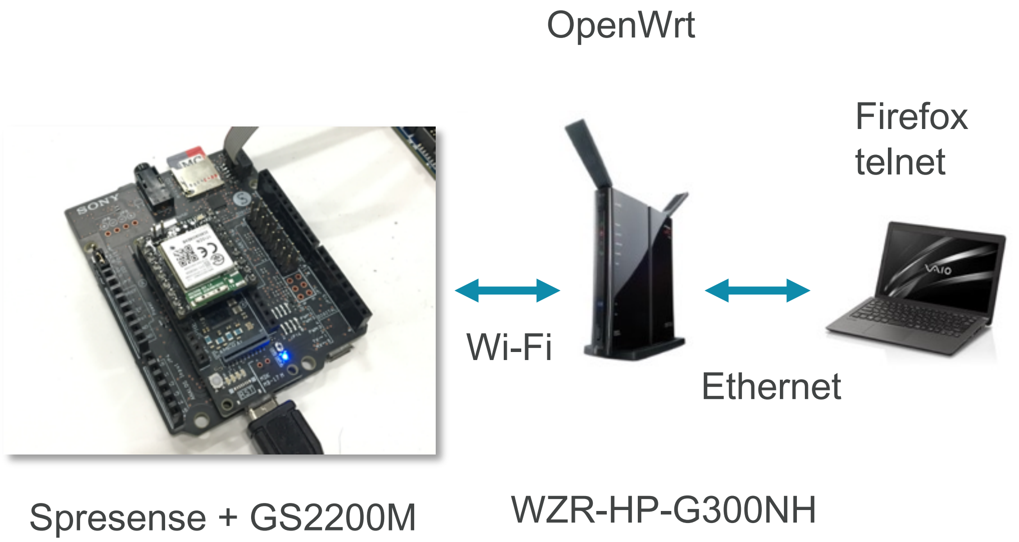
COMPLETED (July Workshop)			COMPLETED (October Meetup)		PLANNED(January Meetup)
CORE adc allocateheap clock composite cpufifo delay dmac farapi flash gpio gpioint i2c i2cdev gpioif icc idle leds pinconfig pmic powermgr	pwm rtc scu sdcard sdhci serial sfc spi spisd sysctl timer timerisr uart udmac uid usbdev usbmsc userleds Wdt **WIFI** gs2200m	**SENSORS (I2C)** ak09912 bmi160 Bmp280 **DISPLAY** ili9340 lpm013m091a **CAMERA** video cisif isx012 **Extra HW** buttons charger emmc gauge **GRAPHIC** ge2d	**GNSS** cpu1signal geofence gnss **BT** **EXTRA** backuplog crashdump **ASMP** sph	**SENSORS (SCU)** ak09912 apds9930 apds9960 bcm20706 bh1721fvc bh1745nuc bm1383glv bm1422gmv bmi160 bmp280 kx022 lt1pa01 Rpr0521rs **AUDIO** audio audio_amp	**MODEM** altmdm altmdm_spi audio_baseband

Working with Wi-Fi

- Wi-Fi module : Telit GS2200M
  - Radio protocols: 802.11b/g/n (2.4GHz)
  - Interface : SPI 10MHz with DMA

- Implement GS2200M driver from scratch*
  - Based on the NuttX usrsock
  - Both STA and AP modes are supported
  - Fix cxd56_gpioint.c for interrupt handling
  - TCP and UDP are supported

- Modify the uIP webserver app for NuttX
  - Add a directory listing feature

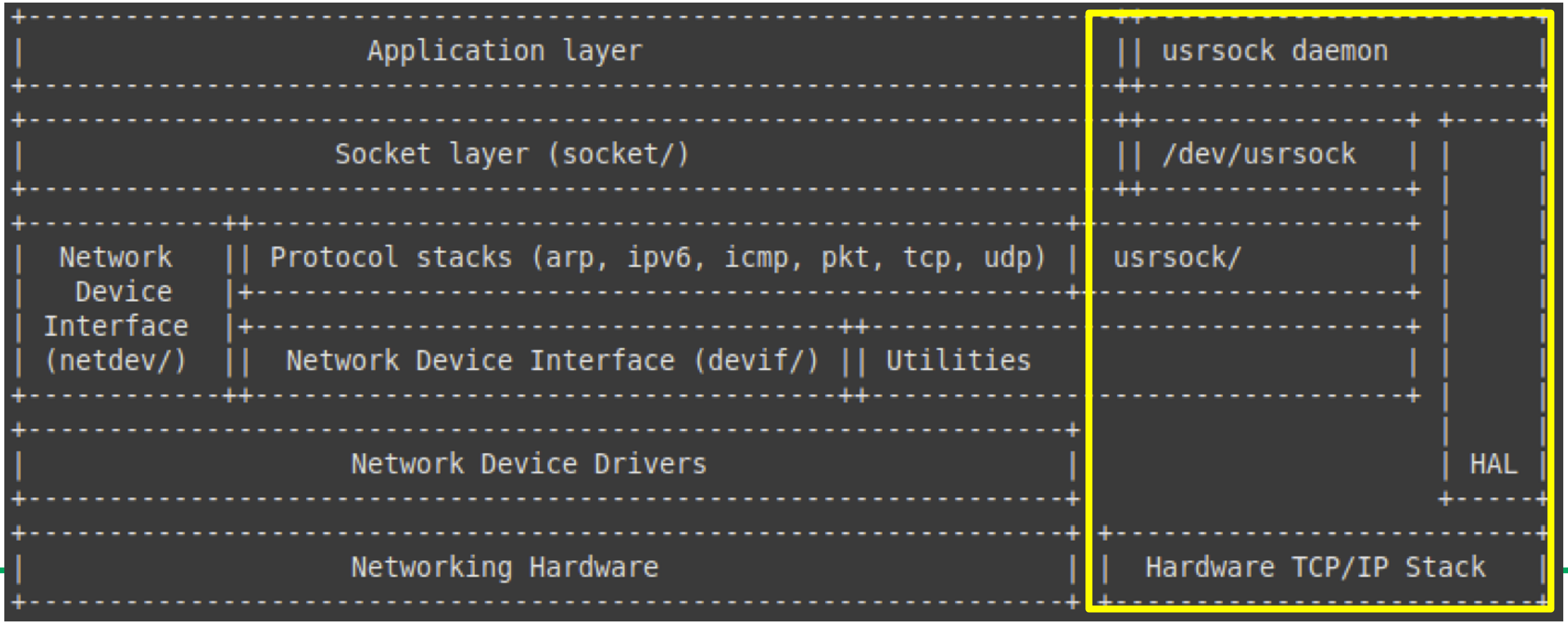


Wi-Fi demo environment

* The code is available at https://bitbucket.org/nuttx/nuttx

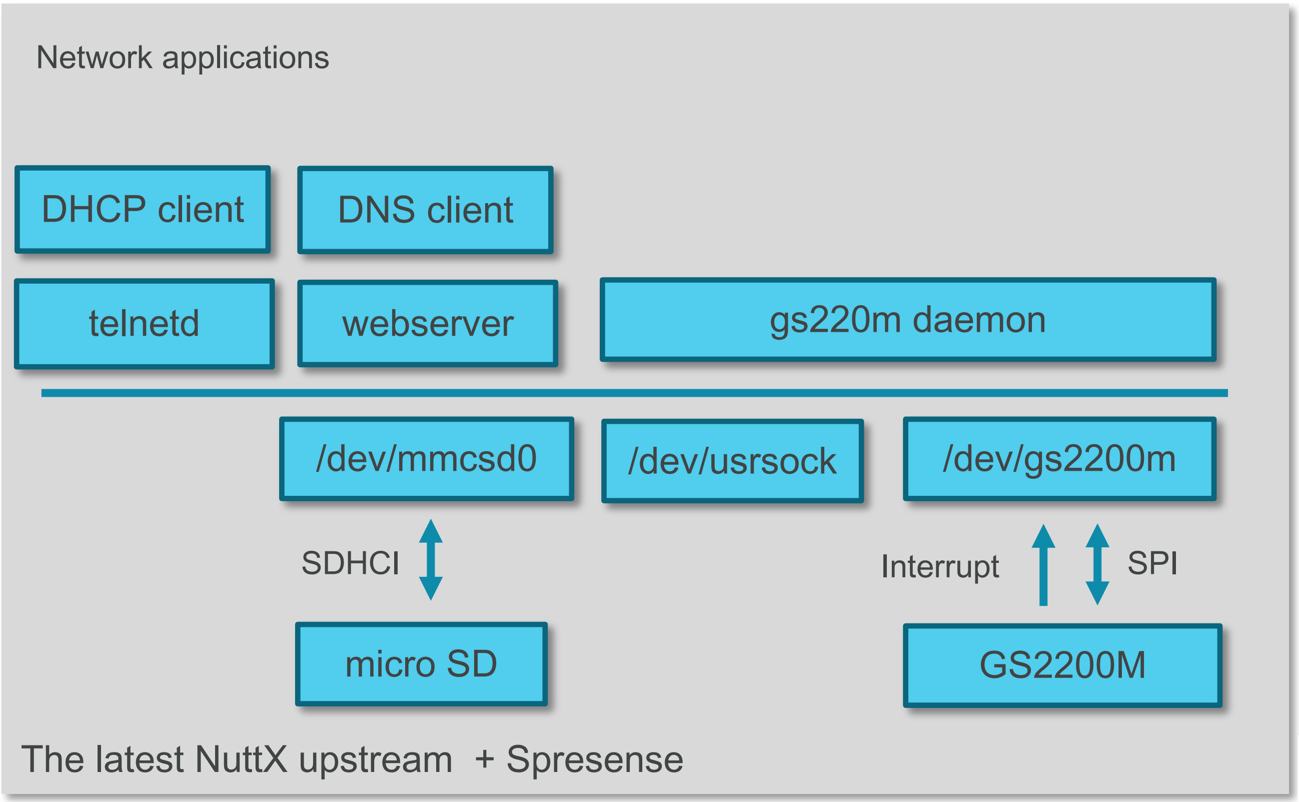
What is the usrsock ?

- User-space networking stack API defined in NuttX
- User-space daemon and HAL provide NuttX networking
- This allows seamless integration of HW-provided TCP/IP stacks to NuttX



```
+-------------------------------------------------------------------++------------------------+
|                    Application layer                              || usrsock daemon         |
+-------------------------------------------------------------------++------------------------+
+-------------------------------------------------------------------++----------------+ +-----+
|                  Socket layer (socket/)                           || /dev/usrsock   | |     |
+-------------------------------------------------------------------++----------------+ |     |
+------------++--------------------------------------------------+--------------------+ |     |
|  Network   || Protocol stacks (arp, ipv6, icmp, pkt, tcp, udp) | usrsock/           | |     |
|   Device   |+--------------------------------------------------+--------------------+ |     |
| Interface  |+------------------------------------++---------------------------------+ |     |
| (netdev/)  ||  Network Device Interface (devif/) || Utilities                       | |     |
+------------++------------------------------------++---------------------------------+ |     |
+----------------------------------------------------------------+                      |     |
|                   Network Device Drivers                       |                      | HAL |
+----------------------------------------------------------------+                      +-----+
+----------------------------------------------------------------+ +--------------------------+
|                   Networking Hardware                          | |  Hardware TCP/IP Stack   |
+----------------------------------------------------------------+ +--------------------------+
```

Use-case for Webserver via Wi-Fi



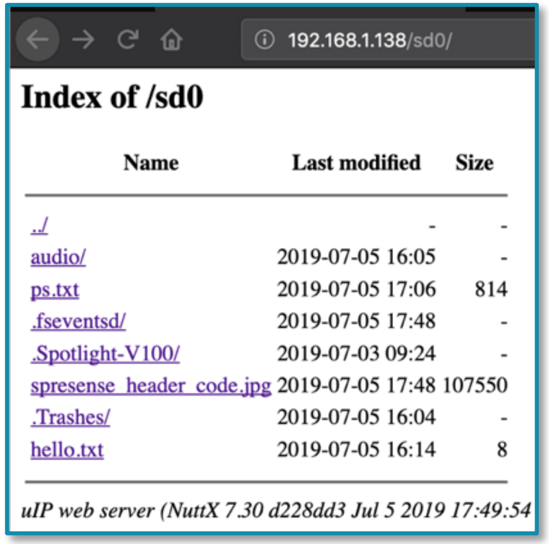
Wi-Fi

OpenWrt

WZR-HP-G300NH

Wi-Fi

Web browsing with Firefox



192.168.1.138/sd0/

Index of /sd0

Name	Last modified	Size
../	-	-
audio/	2019-07-05 16:05	-
ps.txt	2019-07-05 17:06	814
.fseventsd/	2019-07-05 17:48	-
.Spotlight-V100/	2019-07-03 09:24	-
spresense_header_code.jpg	2019-07-05 17:48	107550
.Trashes/	2019-07-05 16:04	-
hello.txt	2019-07-05 16:14	8

uIP web server (NuttX 7.30 d228dd3 Jul 5 2019 17:49:54

VirtualBox + Ubuntu

Code size:
wifi: .text=157,234 .bss=7,956, rndis*: .txt=166,924, .bss=15,384

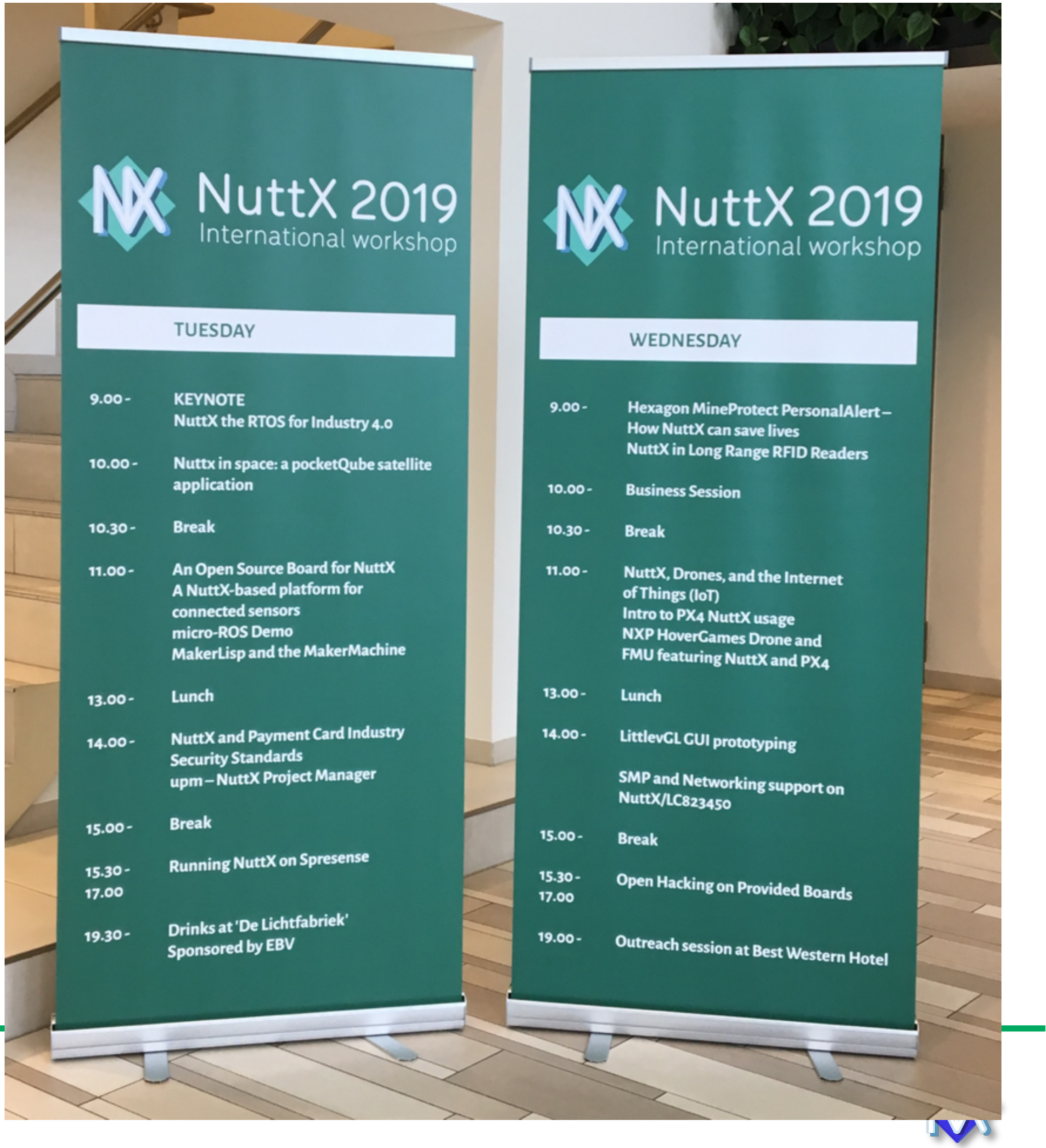
* USB MSC was removed to compare, because it is not used in Wi-Fi configuration

NuttX workshop

- About NuttX2019
- Applications Introduced at NuttX2019
- Why they chose NuttX

About NuttX2019

- The 1st international workshop for NuttX
- Held in Gouda, the Netherlands (Jul/16,17)
- Over 40 people joined



NuttX 2019
International workshop

TUESDAY

9.00 -	KEYNOTE NuttX the RTOS for Industry 4.0
10.00 -	Nuttx in space: a pocketQube satellite application
10.30 -	Break
11.00 -	An Open Source Board for NuttX A NuttX-based platform for connected sensors micro-ROS Demo MakerLisp and the MakerMachine
13.00 -	Lunch
14.00 -	NuttX and Payment Card Industry Security Standards upm – NuttX Project Manager
15.00 -	Break
15.30 - 17.00	Running NuttX on Spresense
19.30 -	Drinks at 'De Lichtfabriek' Sponsored by EBV

NuttX 2019
International workshop

WEDNESDAY

9.00 -	Hexagon MineProtect PersonalAlert – How NuttX can save lives NuttX in Long Range RFID Readers
10.00 -	Business Session
10.30 -	Break
11.00 -	NuttX, Drones, and the Internet of Things (IoT) Intro to PX4 NuttX usage NXP HoverGames Drone and FMU featuring NuttX and PX4
13.00 -	Lunch
14.00 -	LittlevGL GUI prototyping SMP and Networking support on NuttX/LC823450
15.00 -	Break
15.30 - 17.00	Open Hacking on Provided Boards
19.00 -	Outreach session at Best Western Hotel

* Slides are available at https://nuttx2019.org/slides-and-videos/

Applications* introduced at NuttX2019

Drone by Verge Aero

Pixhawk (flight controller)

NXP drone reference design

micro-ROS Kobuki by Bosch

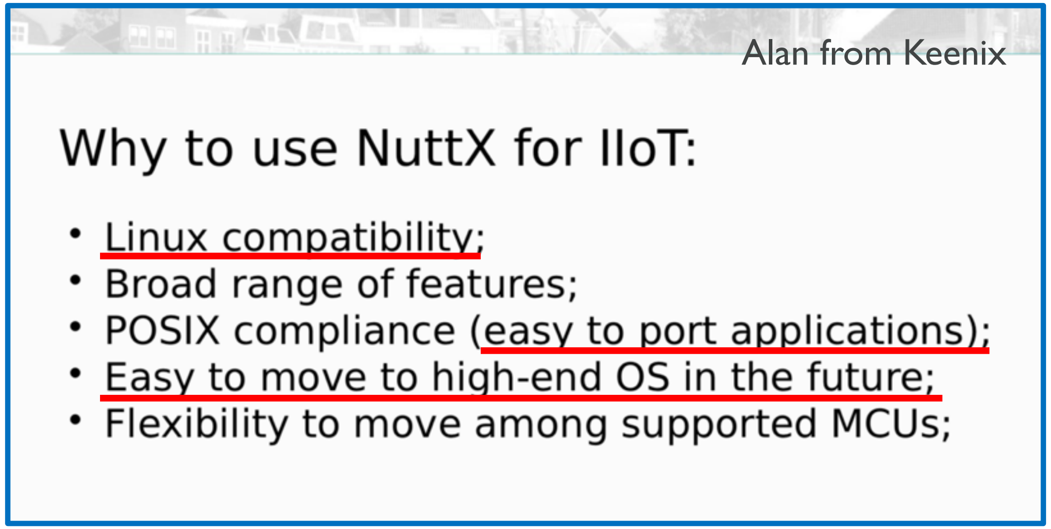
Personal Alert system by Hexagon

TagMaster by TagMaster AB

CubeSat by STARA

* We also introduced audio products and Spresense.

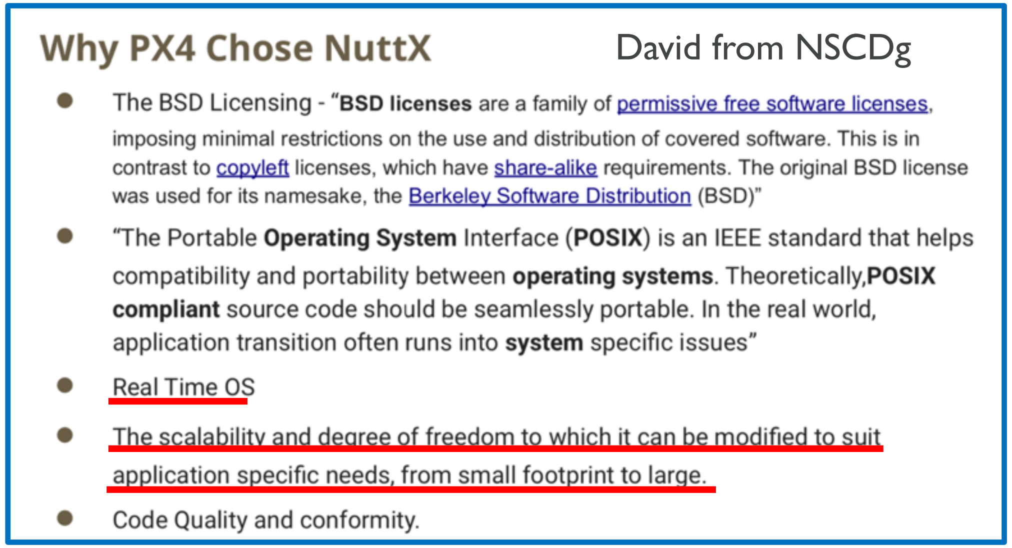
Why they chose NuttX



Alan from Keenix

Why to use NuttX for IIoT:

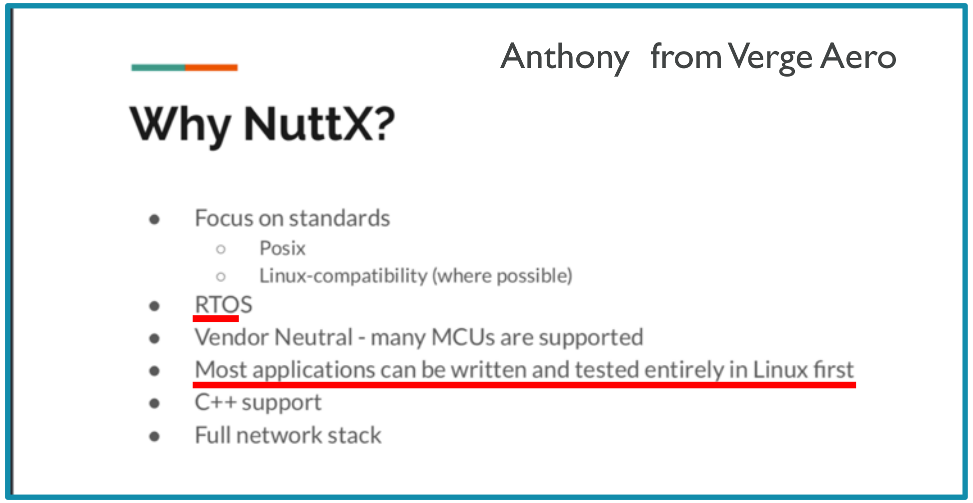
- Linux compatibility;
- Broad range of features;
- POSIX compliance (easy to port applications);
- Easy to move to high-end OS in the future;
- Flexibility to move among supported MCUs;



Why PX4 Chose NuttX

David from NSCDg

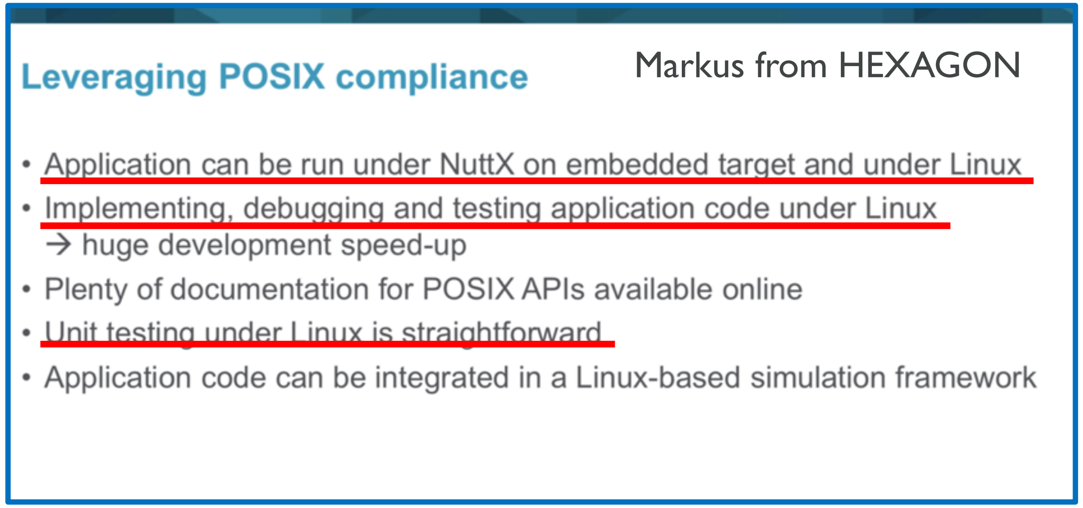
- The BSD Licensing - "**BSD licenses** are a family of permissive free software licenses, imposing minimal restrictions on the use and distribution of covered software. This is in contrast to copyleft licenses, which have share-alike requirements. The original BSD license was used for its namesake, the Berkeley Software Distribution (BSD)"
- "The Portable **Operating System** Interface (**POSIX**) is an IEEE standard that helps compatibility and portability between **operating systems**. Theoretically,**POSIX compliant** source code should be seamlessly portable. In the real world, application transition often runs into **system** specific issues"
- Real Time OS
- The scalability and degree of freedom to which it can be modified to suit application specific needs, from small footprint to large.
- Code Quality and conformity.



Anthony from Verge Aero

Why NuttX?

- Focus on standards
  - Posix
  - Linux-compatibility (where possible)
- RTOS
- Vendor Neutral - many MCUs are supported
- Most applications can be written and tested entirely in Linux first
- C++ support
- Full network stack



Leveraging POSIX compliance

Markus from HEXAGON

- Application can be run under NuttX on embedded target and under Linux
- Implementing, debugging and testing application code under Linux
  → huge development speed-up
- Plenty of documentation for POSIX APIs available online
- Unit testing under Linux is straightforward
- Application code can be integrated in a Linux-based simulation framework

Demo videos

- LC823450 + AVS Device SDK
- Spresense + Wi-Fi
- LC823450 + HTTP Audio streaming in SMP

Any Questions?

Embedded Linux Conference

Europe