EOSS

EMBEDDED
OPEN SOURCE
SUMMIT

EMBEDDED
OPEN SOURCE
SUMMIT

Introducing a New Zephyr Sensing Subsystem

Hebo Hu, Ke Han
Intel Corporation

#EMBEDDEDOSSUMMIT

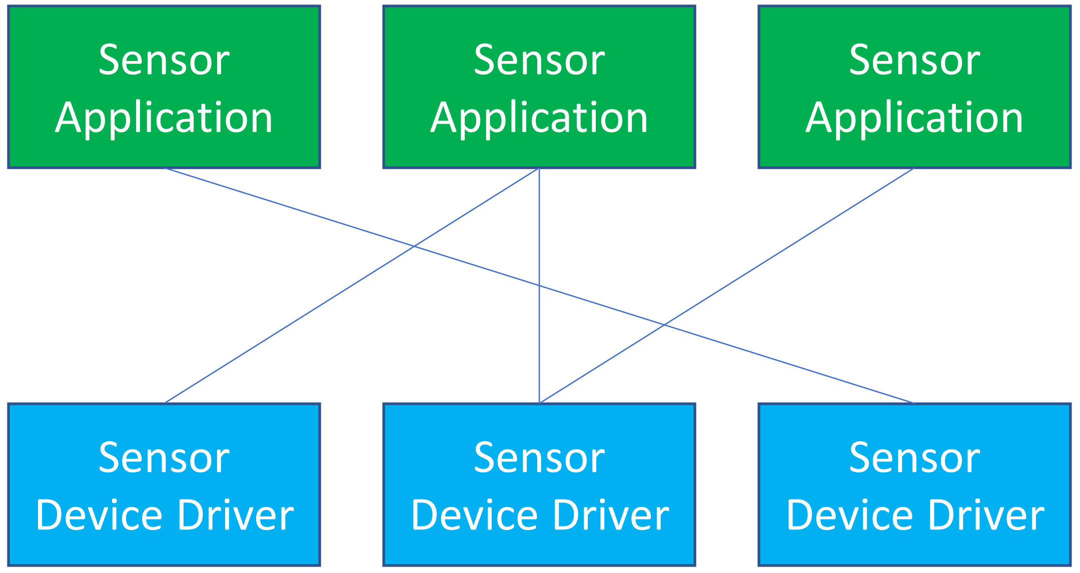
• Zephyr Sensing Subsystem	Agenda Items
• Why do we need it?	Background
• What is our goal?	Motivation
• What is it?	Introduction
• What can it do?	Application
• How far did we go?	Status
• How does it work?	Demo
• What is next?	Next Plan

Background

Zephyr Sensors Today

- device tree config
- sensor fetching
- sensor triggering
- sensor attribute
- sensor calibration
- sensor processing
- ~140 device drivers
- ~900 commits

Typical Zephyr Sensor Usages



Improvement area: sensor applications are directly interacting with sensor device drivers for all sensor operations, there is a lack of high-level abstraction, general function management, multi-client arbitration, etc. in Zephyr Sensor solution

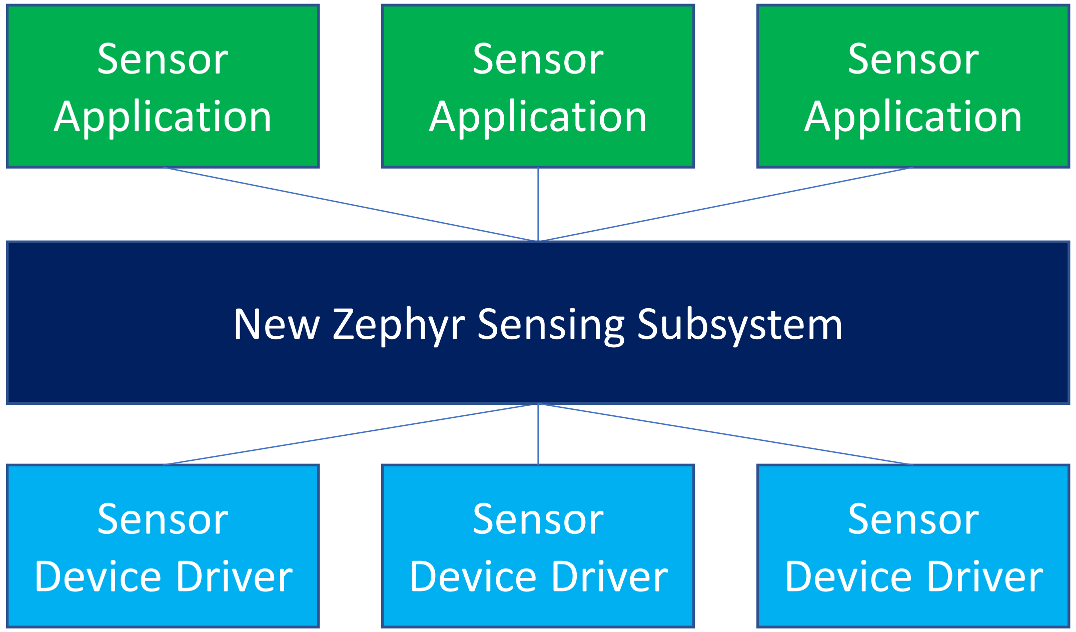
EMBEDDED OPEN SOURCE SUMMIT

Motivation

Take Zephyr Sensor Solution to Next Level

Plan: introduce a new sensing subsystem / framework to support

- Sensor high level abstraction
- General function management
- Multi-client arbitration
- Sensor topology and scheduling
- Sensor fusion and processing
- Sensor timestamping
- Sensor batching
- And more …



Furthermore:

- With additional components, new **Sensing Subsystem** shall be able to serve as sensor framework in sensor hub firmware, to support OS sensing stack (Windows*, Chrome*, Android*, Linux*) and CHRE*.

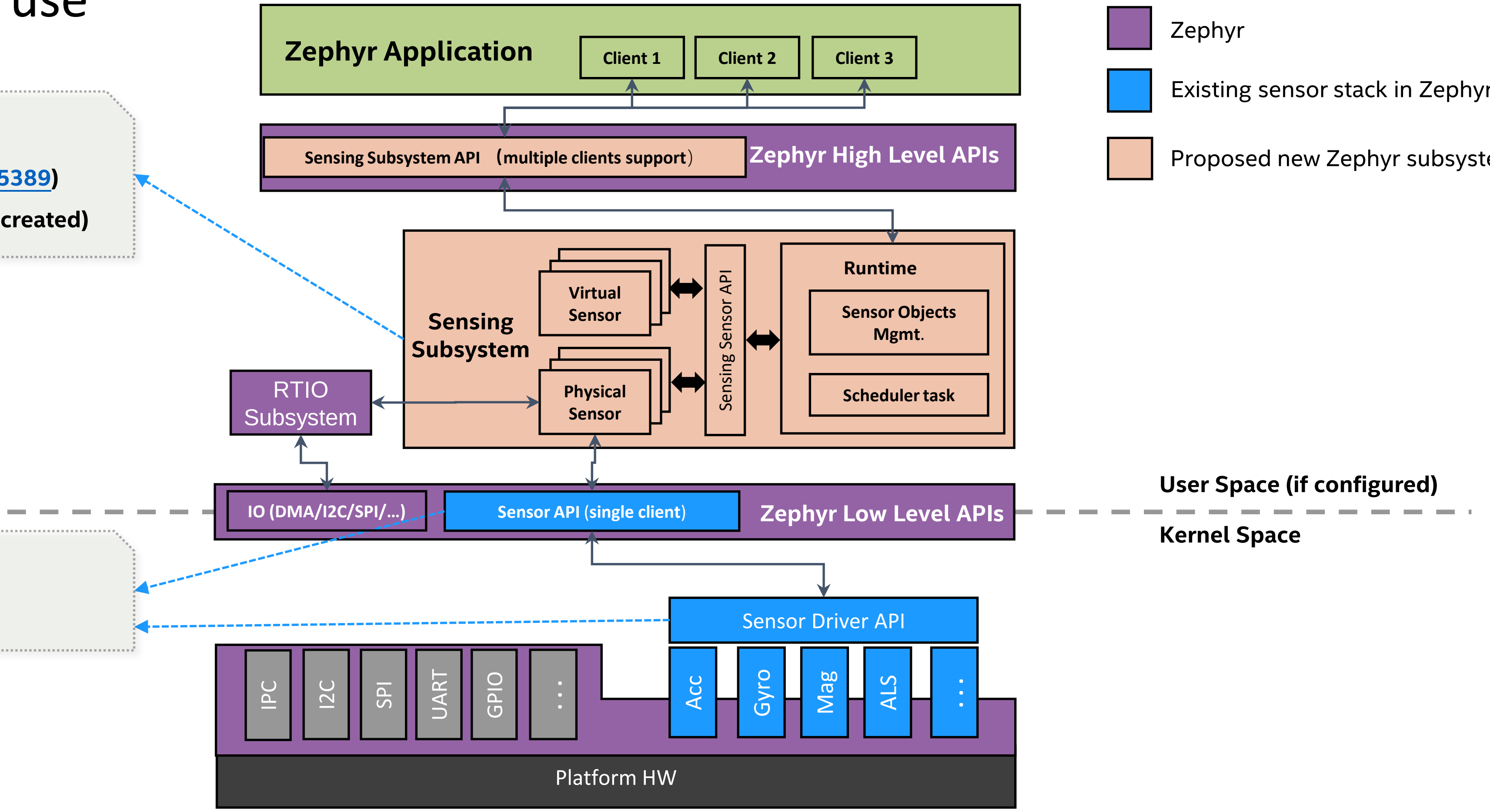
*Other names and brands may be claimed as the property of others

Introduction

New **Sensing Subsystem** runs on top of existing Zephyr sensor device drivers, serves as sensor management framework, and provides higher level sensor APIs for multiple Zephyr applications to use

Sensor Subsystem Proposal from Intel (PR#45370)
Sensing Subsystem Upstream PR1 from Intel (PR#55389)
Sensing Subsystem Upstream PR2 from Intel (to be created)

Zephyr existing sensor device driver API



🟪	Zephyr
🟦	Existing sensor stack in Zephyr
🟧	Proposed new Zephyr subsystem

EMBEDDED OPEN SOURCE SUMMIT

Introduction

Sensor Abstraction

- Virtual Sensor: sensor algorithms
- Physical Sensor: wrapper of exiting sensor device driver
- Sensor commonality handled by sensor framework

Tree Model

- Config flow from top to bottom
- Data streaming flow from bottom to top
- Single thread scheduling driven by data and config

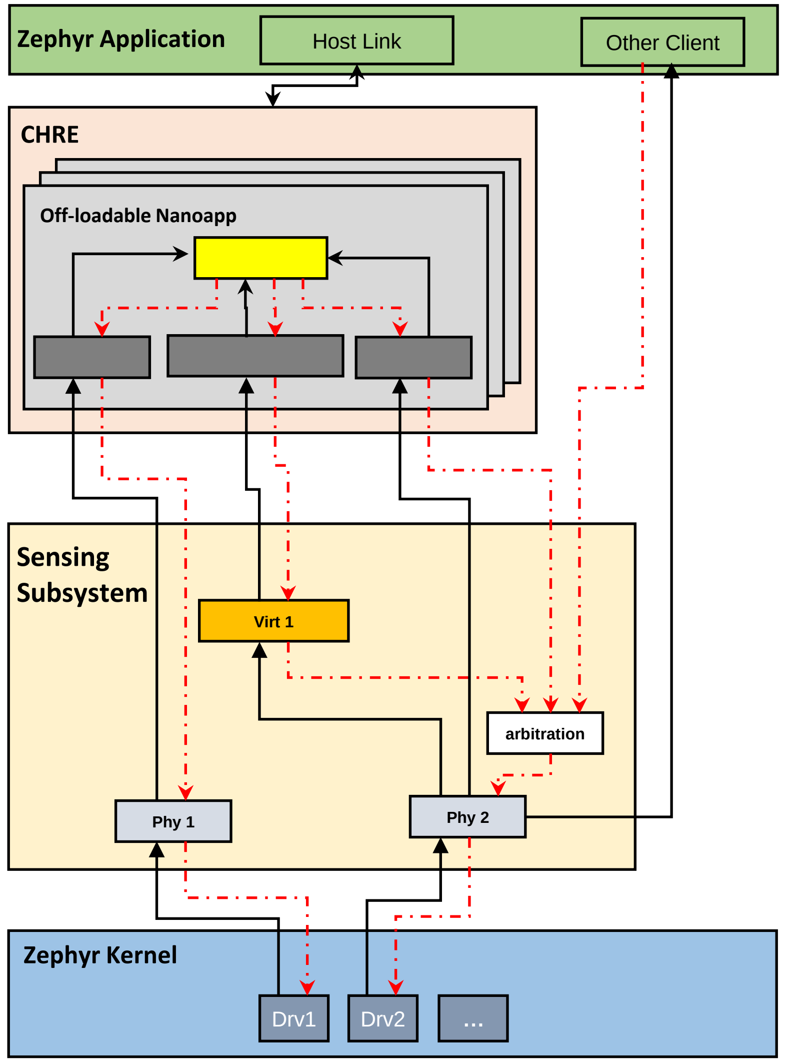
Multiple Clients w/ Arbitration

- Each Sensor can be opened & configured by multiple clients including App clients and Sensor clients
- Arbitration: report interval, change sensitivity, etc.

Support CHRE

- API can meet CHRE PAL's requirements

Support None-CHRE Zephyr Application



🟨	Sensor Implemented in nanoapp
⬛	Sensor Exposed by Sensor Framework
⇢	Config Path
→	Data Path
🟧	Virtual Sensor in Sensor Framework
⬜	Arbitration
▨	Physical Sensor in Sensor Framework
▨	Sensor device driver in Zephyr Kernel

EOSS EMBEDDED OPEN SOURCE SUMMIT

Introduction

Zephyr Sensing Subsystem Features

Scope

- focus on sensor **fusion**, **arbitration**, **sampling**, **timing** and **scheduling**

Sensor Abstraction

- sensor object for representing sensor instance, **physical** & **virtual**
- reporting tree model for sensor fusion, including data & config paths

Data Driven Model

- **polling mode**: periodical sampling rate
- **interrupt mode**: data ready, threshold interrupt etc.

Scheduling

- single thread main loop for all sensor objects sampling & processing

Buffer Mode for Batching

Sensor Power Management

Rely on Zephyr APIs

- access physical sensor devices via **Sensor API**
- others, such as **Task**, **Memory**, **RTIO**

Configurable Via Device Tree

- enabled or not, supported sensors if enabled
- defines sensor reporting relationships (including reporter sensors for virtual sensors)

Design Principles

Reference

- reference from **Intel ISH** proprietary sensor framework design – which has been in mass production for 10+ years w/ active evolution

Upstream

- fully open source as Zephyr native subsystem
- start upstreaming in 2023
- work together with Google*

Configurability

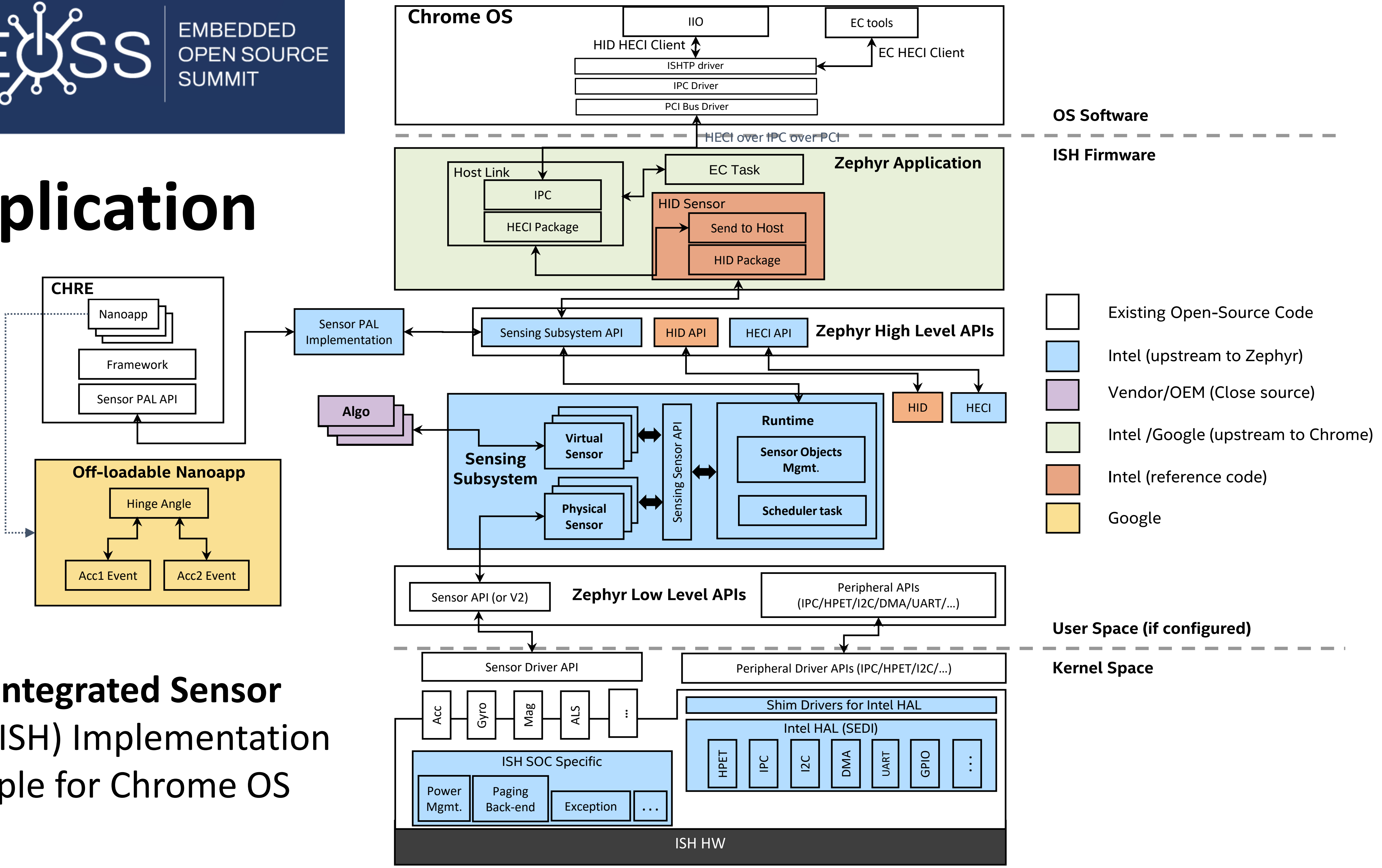
- **reusable standalone subsystem**
- build on Zephyr existing **low-level Sensor API**
- provide Zephyr **high level Sensing Framework API**
- separate optional **CHRE Sensor PAL Implementation** module to support **CHRE**
- decouple from any external communication protocols, it's **Zephyr Application**'s role to handle different protocols (**MQTT**, **HID**, or **Proprietary**, all configurable)

Application

- **Chrome**, **Linux**, **Windows**, **Android** OS sensing support
- **CHRE** (Context Hub Runtime Environment) **PAL** support

*Other names and brands may be claimed as the property of others

Application



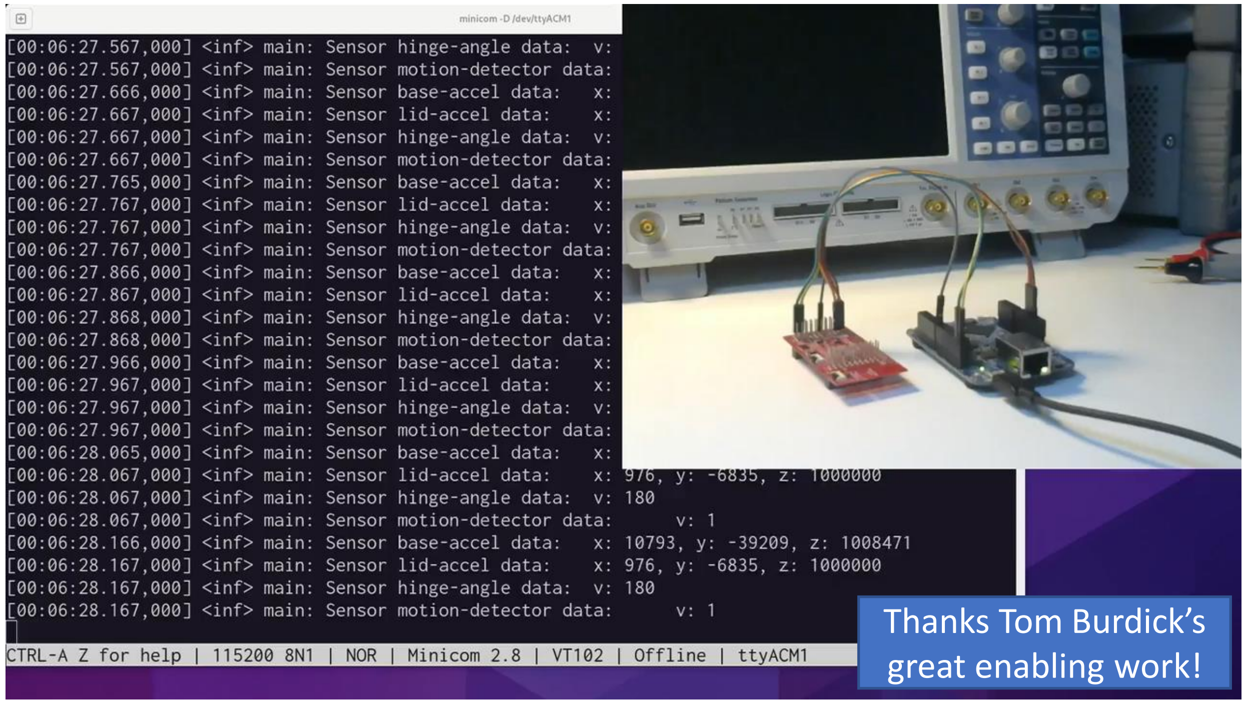
Intel **Integrated Sensor Hub** (ISH) Implementation Example for Chrome OS

Status

- Support Intel Chrome Client Reference Platform sensor BOM
- Features implemented
  - All sensing subsystem APIs except batching
  - Fusion framework with physical and virtual sensor abstraction
  - Multiple clients with CHRE and HID sensor apps running at the same time
  - Multiple clients’ arbitration on sensor report interval and sensitivity
  - Utilization of Zephyr sensor device API for physical sensors
  - Polling and interrupt data driven
  - Single thread scheduling
  - Configurable with device tree
  - E2E sensor streaming data path with HID/IIO for demo (physical sensor + virtual sensor)
- Upstream work
  - Sensing Subsystem Proposal has been reviewed with Zephyr community
  - Sensing Subsystem Upstream PR1 has been submitted and is under review

Demo

Sensor Data Streaming Through CHRE Path



```
[00:06:27.567,000] <inf> main: Sensor hinge-angle data:  v:
[00:06:27.567,000] <inf> main: Sensor motion-detector data:
[00:06:27.666,000] <inf> main: Sensor base-accel data:   x:
[00:06:27.667,000] <inf> main: Sensor lid-accel data:    x:
[00:06:27.667,000] <inf> main: Sensor hinge-angle data:  v:
[00:06:27.667,000] <inf> main: Sensor motion-detector data:
[00:06:27.765,000] <inf> main: Sensor base-accel data:   x:
[00:06:27.767,000] <inf> main: Sensor lid-accel data:    x:
[00:06:27.767,000] <inf> main: Sensor hinge-angle data:  v:
[00:06:27.767,000] <inf> main: Sensor motion-detector data:
[00:06:27.866,000] <inf> main: Sensor base-accel data:   x:
[00:06:27.867,000] <inf> main: Sensor lid-accel data:    x:
[00:06:27.868,000] <inf> main: Sensor hinge-angle data:  v:
[00:06:27.868,000] <inf> main: Sensor motion-detector data:
[00:06:27.966,000] <inf> main: Sensor base-accel data:   x:
[00:06:27.967,000] <inf> main: Sensor lid-accel data:    x:
[00:06:27.967,000] <inf> main: Sensor hinge-angle data:  v:
[00:06:27.967,000] <inf> main: Sensor motion-detector data:
[00:06:28.065,000] <inf> main: Sensor base-accel data:   x:
[00:06:28.067,000] <inf> main: Sensor lid-accel data:    x: 976, y: -6835, z: 1000000
[00:06:28.067,000] <inf> main: Sensor hinge-angle data:  v: 180
[00:06:28.067,000] <inf> main: Sensor motion-detector data:     v: 1
[00:06:28.166,000] <inf> main: Sensor base-accel data:   x: 10793, y: -39209, z: 1008471
[00:06:28.167,000] <inf> main: Sensor lid-accel data:    x: 976, y: -6835, z: 1000000
[00:06:28.167,000] <inf> main: Sensor hinge-angle data:  v: 180
[00:06:28.167,000] <inf> main: Sensor motion-detector data:     v: 1

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.8 | VT102 | Offline | ttyACM1
```

Thanks Tom Burdick's great enabling work!

Intel Client Reference Board with ISH (Minute-IA)

❑ **Physical sensors:**
- Base Acc: BM160 (I2C)
- Lid Acc: BM160 (SPI)

❑ **Virtual sensors:**
- Hinge angle (Base Acc + Lid Acc as inputs)
- Motion detector (Lid Acc as input)

FRDM-K64F Board (ARM Cortex-M4F)

❑ **Physical sensors:**
- Base Acc: BM160
- Lid Acc: fxos8700

❑ **Virtual sensors:**
- Hinge angle (Base Acc + Lid Acc as inputs)
- Motion detector (Lid Acc as input)

EMBEDDED OPEN SOURCE SUMMIT

Next Plan

- **Upstreaming**

Zephyr 3.4
- Introduce sensing subsystem core (PR)

Zephyr 3.5
- Extend sensing subsystem features and board support

Zephyr 3.6 (LTS 3)
- Stabilize sensing subsystem

- **End to End Chrome Sensors Support**
  - Meet Google/Intel Chrome production requirements
  - Enable Intel ISH with Latest UPX Open Dev Board

- **Advanced Sensing Features**
  - CHRE end to end support
  - Cross-OS support with HID sensors
  - Human presence sensing
  - Sensor batching
  - Multi-thread processing
  - More

EOSS

EMBEDDED
OPEN SOURCE
SUMMIT

EMBEDDED OPEN SOURCE SUMMIT

Notices & Disclaimers

Sensor Data Streaming Through CHRE Path

back