arm

CMSIS

Update Meeting

Replay March 2023

Arm MCU Tools Team
22 March 2023

Reference Application Framework: Production Quality Examples

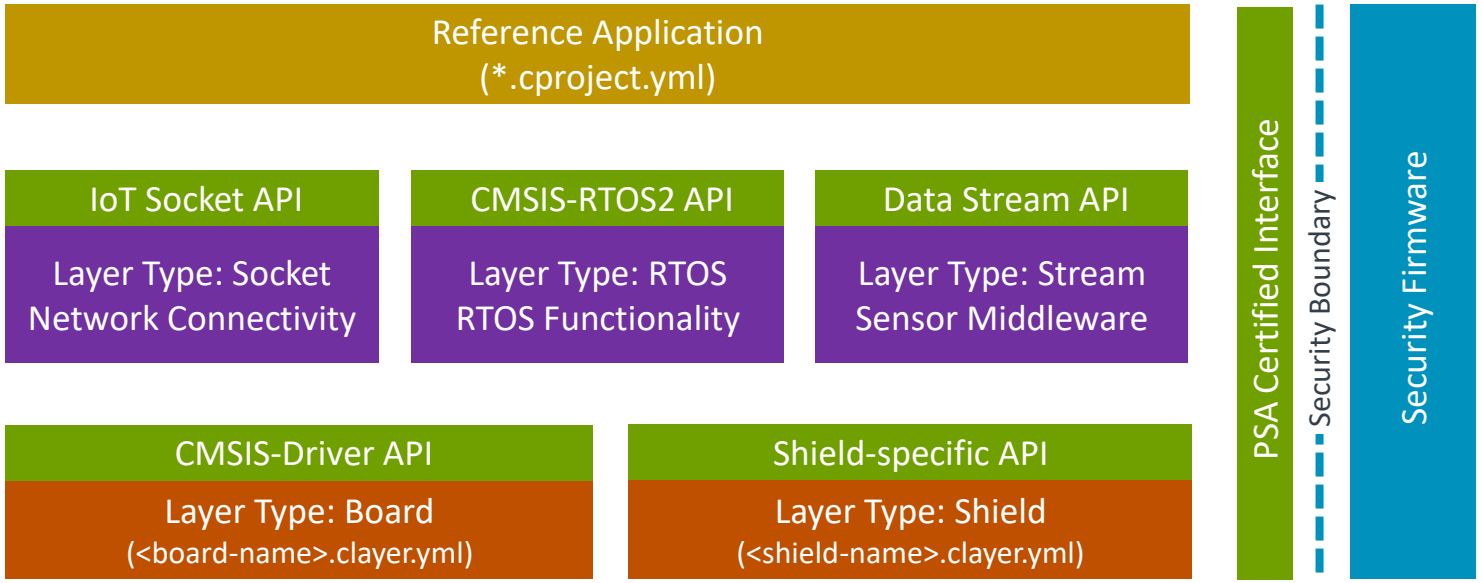
Reusable software applications; developed for standardized hardware abstraction layers

Supports a range of application examples:

- Cloud connectivity using SDKs from Cloud Service Providers.
- Sensor reference examples.
- Machine Learning applications that use sensors and audio inputs.
- Middleware examples such as TCP/IP stack and file system.

Target various evaluation boards, production hardware and even Arm Virtual Hardware:

- Software layers with defined and standardized interfaces contain re-usable parts of applications.
- Description of standardized connections between these software layers.
- Consistent bootstrap and startup sequence.
- Board and Shield layer combined provide target hardware abstraction for many applications and this could be extended further.

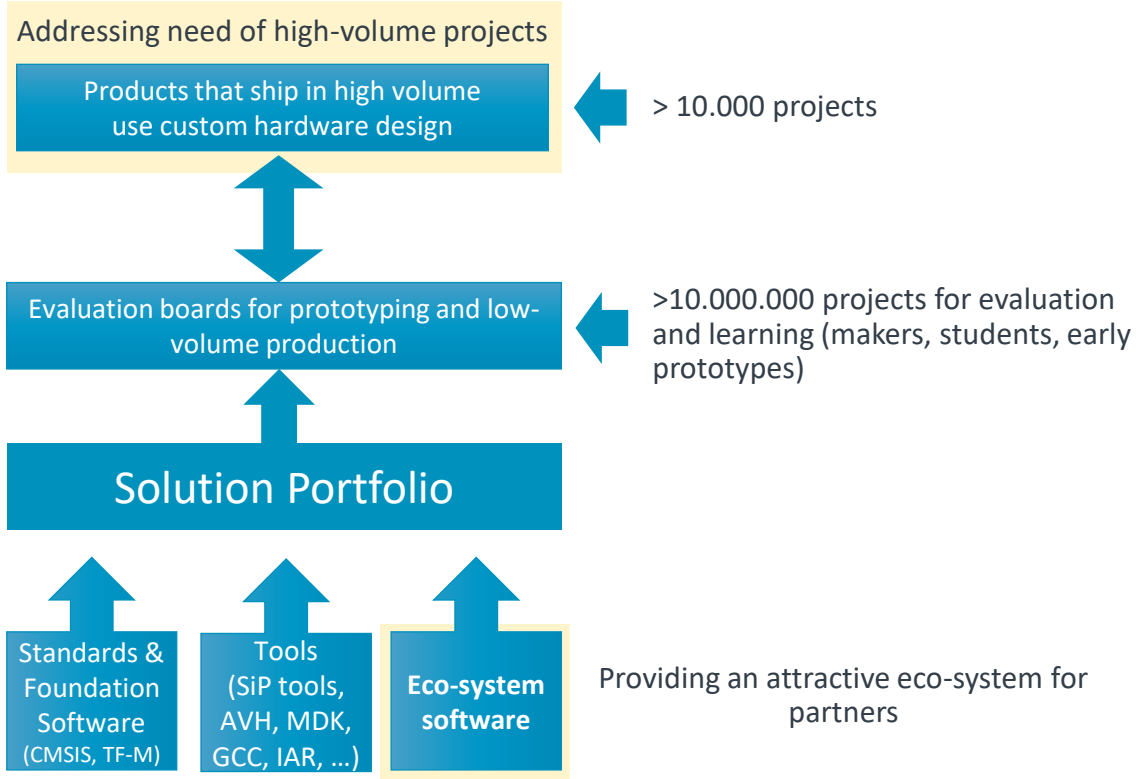


- CMSIS-Toolbox helps selecting compatible layers for target hardware boards
- Arm Virtual Hardware enables CI testing of reference examples on standardized hardware abstraction layers
- SDS Framework enables test data streaming during CI validation
- Initial implementation uses CMSIS-Driver, but is open to other driver standards

What are the care abouts of our target audience?

MCU designs care about cost; software reuse is key for productivity and quality

It's the software that takes the time.



Re-useable software components with standardized interfaces:

- Allow integration into many different software projects.
- Use established verification and validation development processes that are independent of final target hardware.

Frequently machine learning models are developed and trained in isolation of the final hardware target.

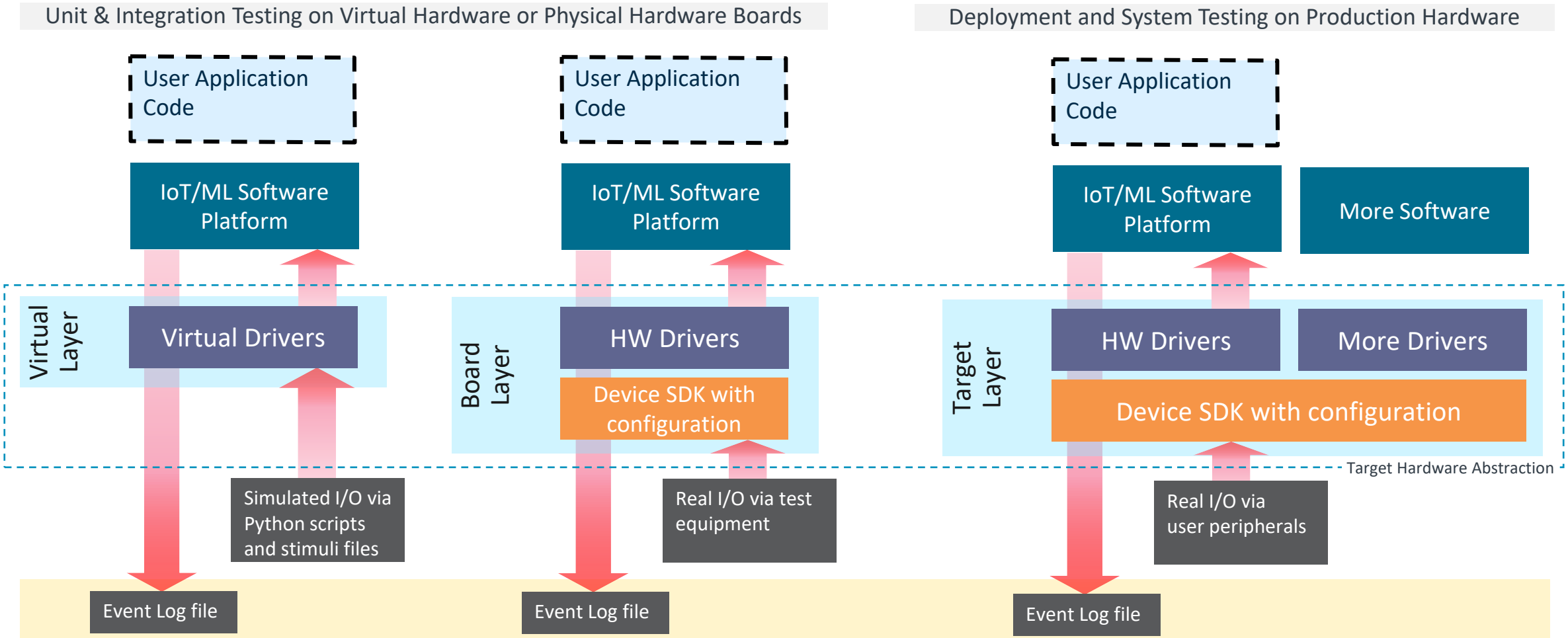
- Use MLOps workflows in the cloud with test and training data.

Big corporations re-use software across multiple projects with diverse development teams or external suppliers.

- Tools that enable code reuse are key, but we need to explain the usage.
- Therefore, tools should be complemented by methods and recommendations on how to structure software.

Application Software – from Virtual to Physical Hardware

Provide evidence of correctness on Arm Virtual Hardware Target and Physical Hardware



Essentially the same event logs are generated across the different deployments. This ensures correctness.

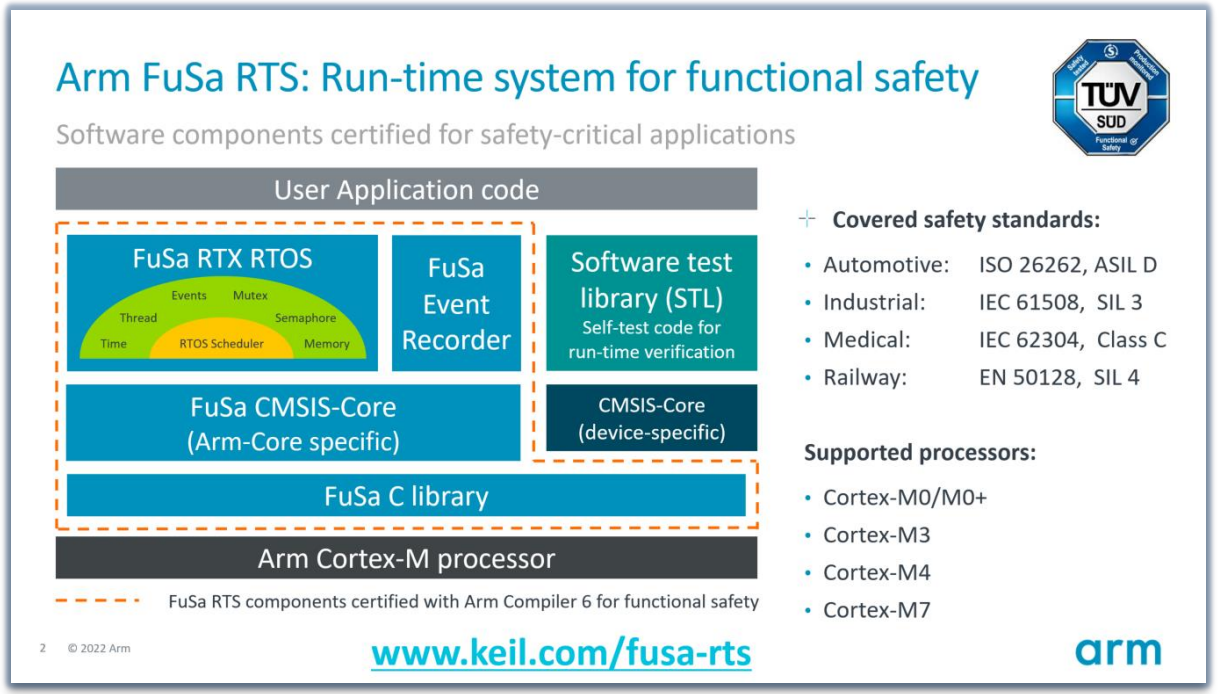
Arm’s commitment to functional safety

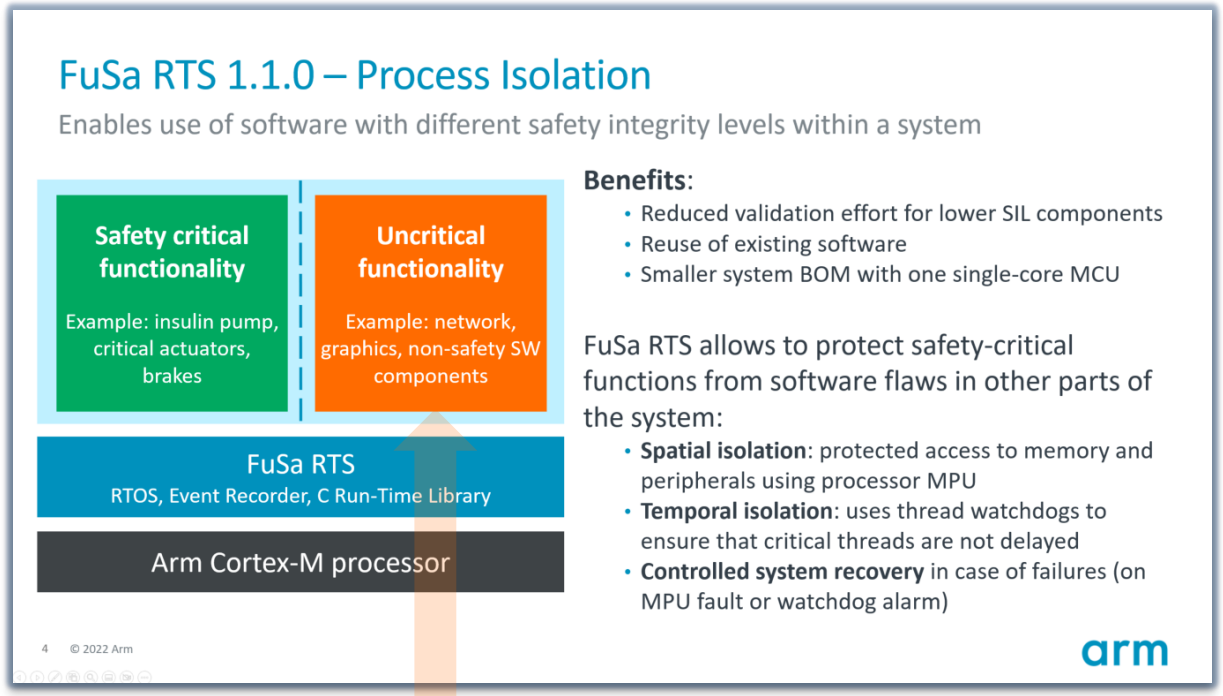
arm.com/safety

Products, tools, platforms, and software to enable functional safety

Development Tools include

- Software Test Libraries (STL) for processors
- Functional Safety Run-Time System (FuSa RTS)
- Arm Compiler for Embedded FuSa
- Verification Tools



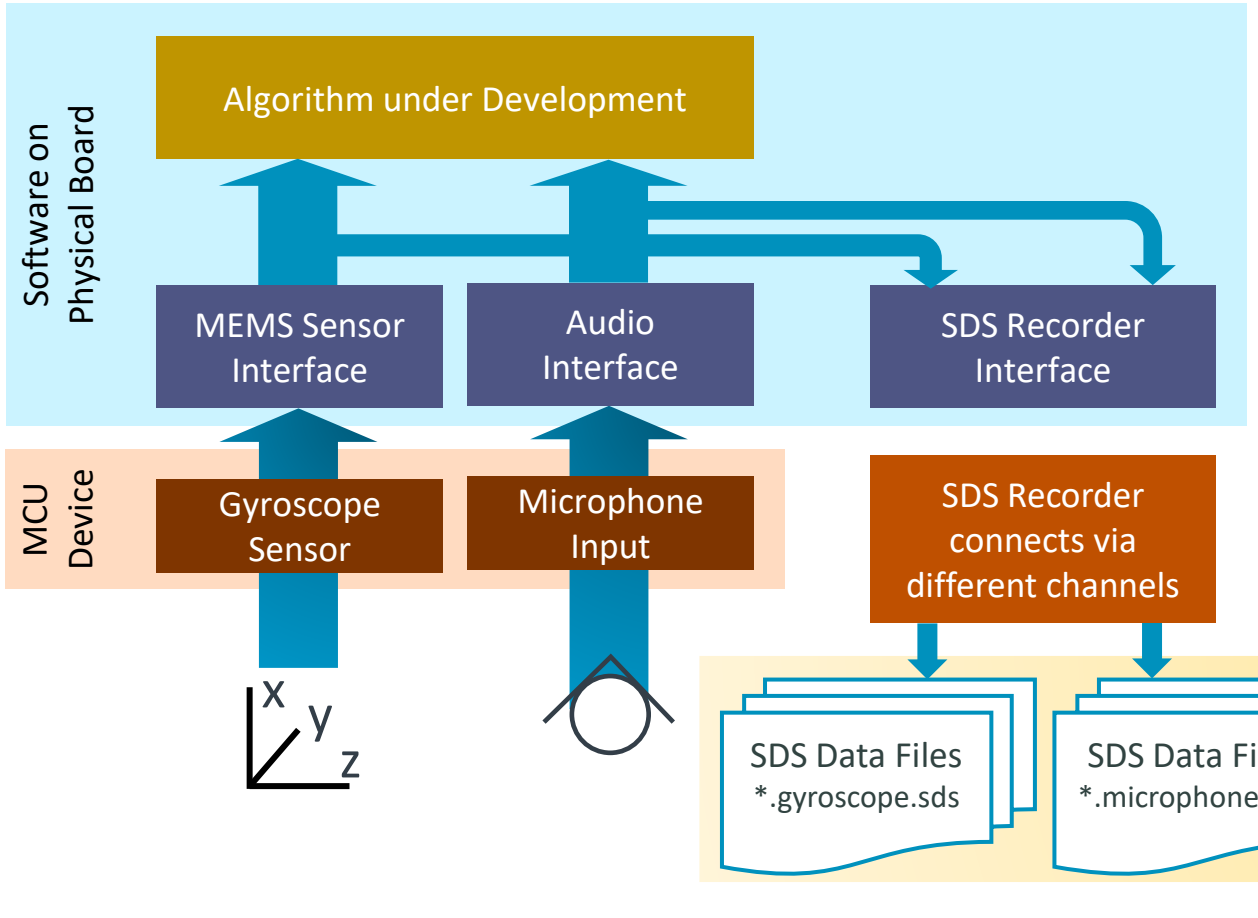


Uncritical functionality should have no direct access to device hardware

Record real-world data with Synchronous Data Streaming (SDS)

Simplify Development of Embedded Applications that utilize DSP or ML algorithms with Sensor/Audio Input

Microcontroller Hardware

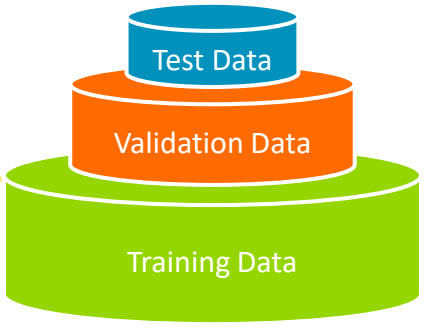


The SDS framework provides methods to record real-world data for analysis and development.

- Input to Digital Signal Processing (DSP) development tools such as filter designers
- Input to Machine Learning (ML) model classification, training, and performance optimization.

SDS framework is an open-source project:
github.com/Arm-Software/SDS-Framework

Data conversion to ML training platform can be automated.

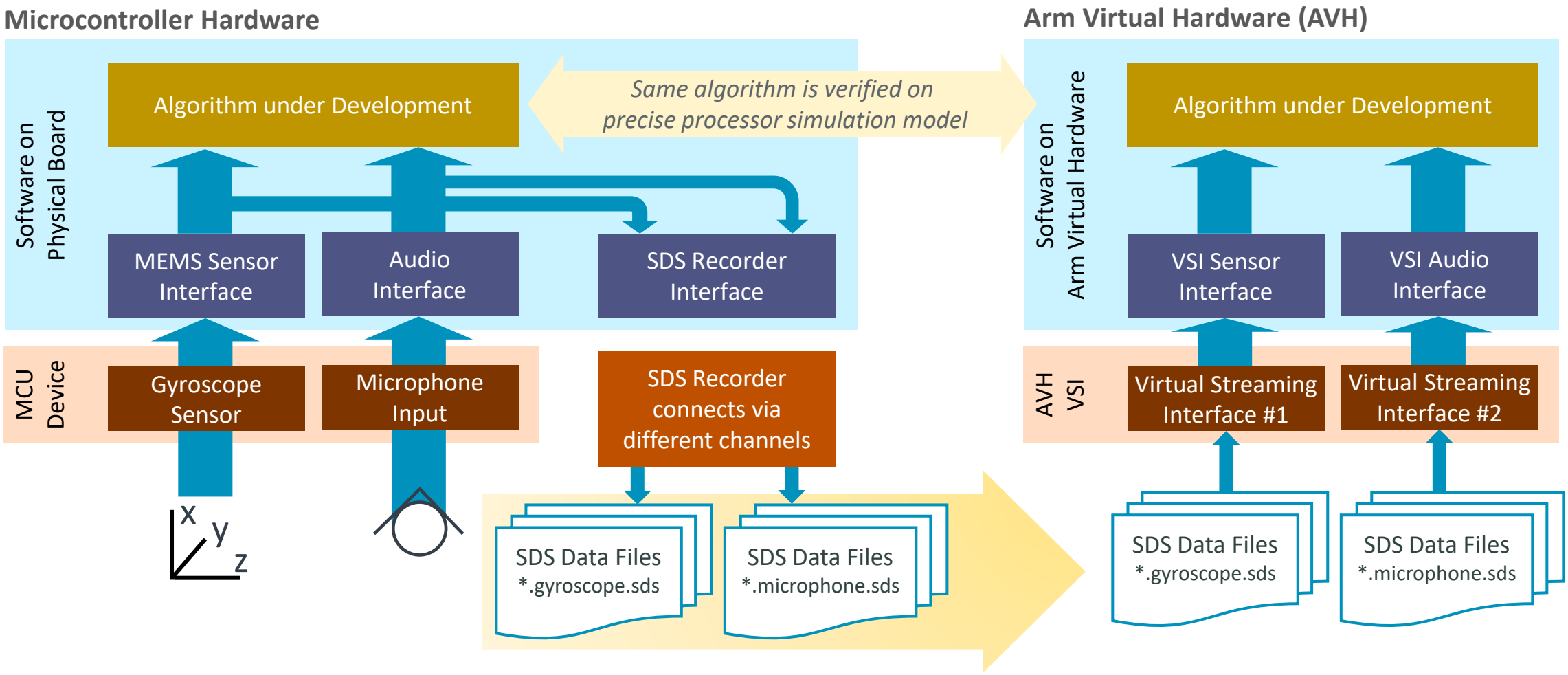


Capture physical sensor (real-world) data using the original Microcontroller target hardware

Enabled by CMSIS-Stream

SDS enables playback of real-world data for algorithm testing

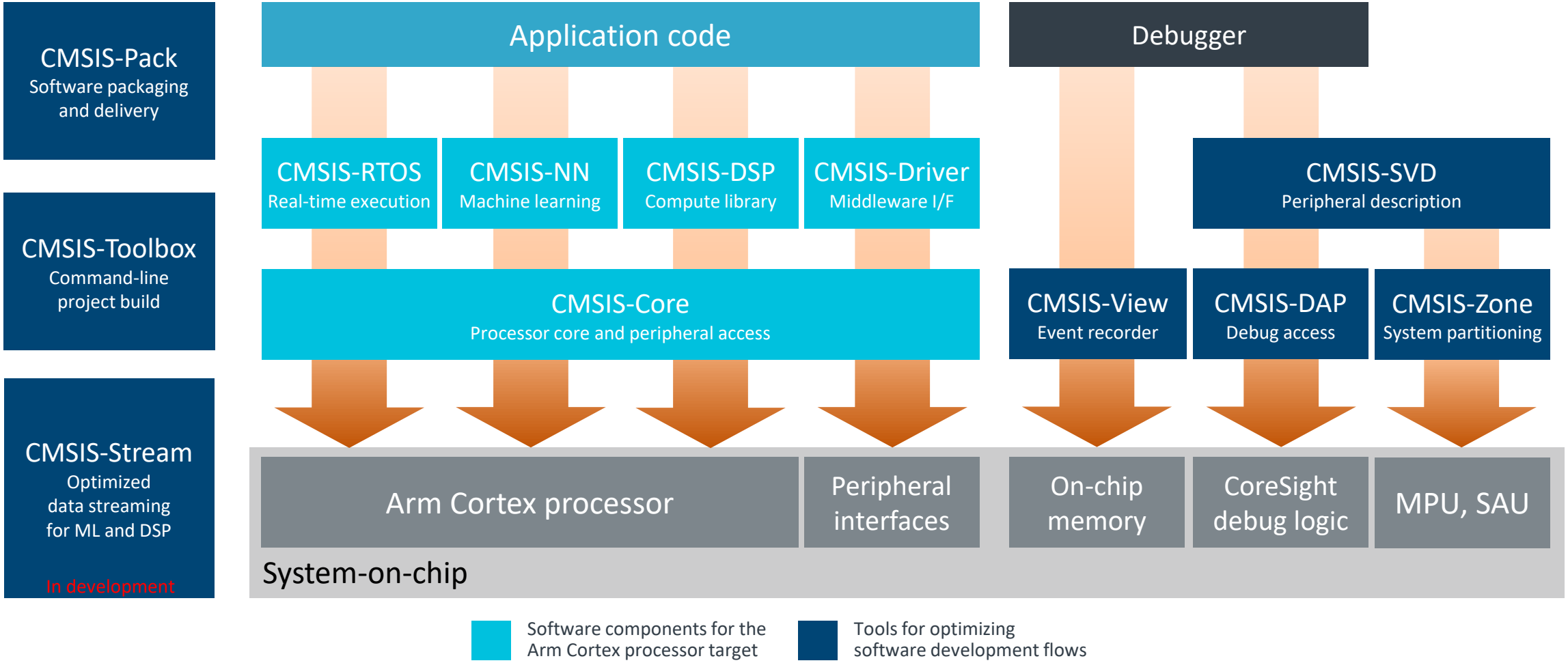
Combined with AVH it enables repeatable test automation in CI systems and MLOps cloud services



More information: watch the recording of the March '23 CMSIS Event

CMSIS Version 6

Consistent software framework for Arm Cortex-M and Cortex-A5/A7/A9 based systems

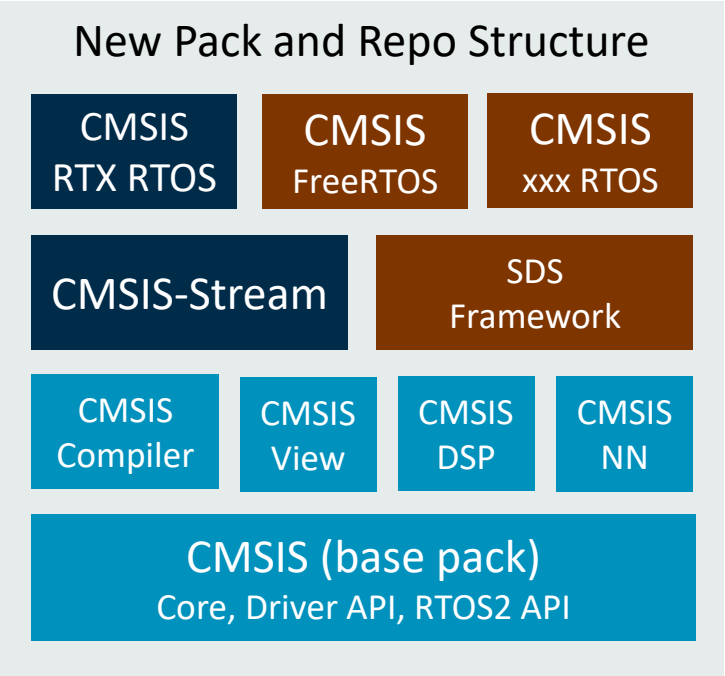


https://community.arm.com/arm-community-blogs/b/tools-software-ides-blog/posts/which-cmsis-components-should-i-care-about

CMSIS Version 6 enhancements (compared to version 5.9.0)

Overall goal: simplify software re-use across supported processors and toolchains

- **Core**: C Startup, new linker scripts (using C header files), fault exception template.
- **Driver**: GPIO for I/O pin control, simplified VIO for LEDs and switches/buttons.
- **RTOS2**: add FuSa RTS API extensions, deprecate TZ handling.
- **Compiler:** I/O retargeting (currently for GCC / AC6)
- **View**: complete initial release.
- **DSP**: incremental improvements in a separate pack.
- **NN**: incremental improvements in a separate pack.
- **Stream**: new component, derived from ComputeGraph (relates to SDS-Framework)
- **CMSIS-Toolbox v2.0** feature complete (i.e. with linker script support).



NOTE: **RTOS**: version 1 deprecate and remove.

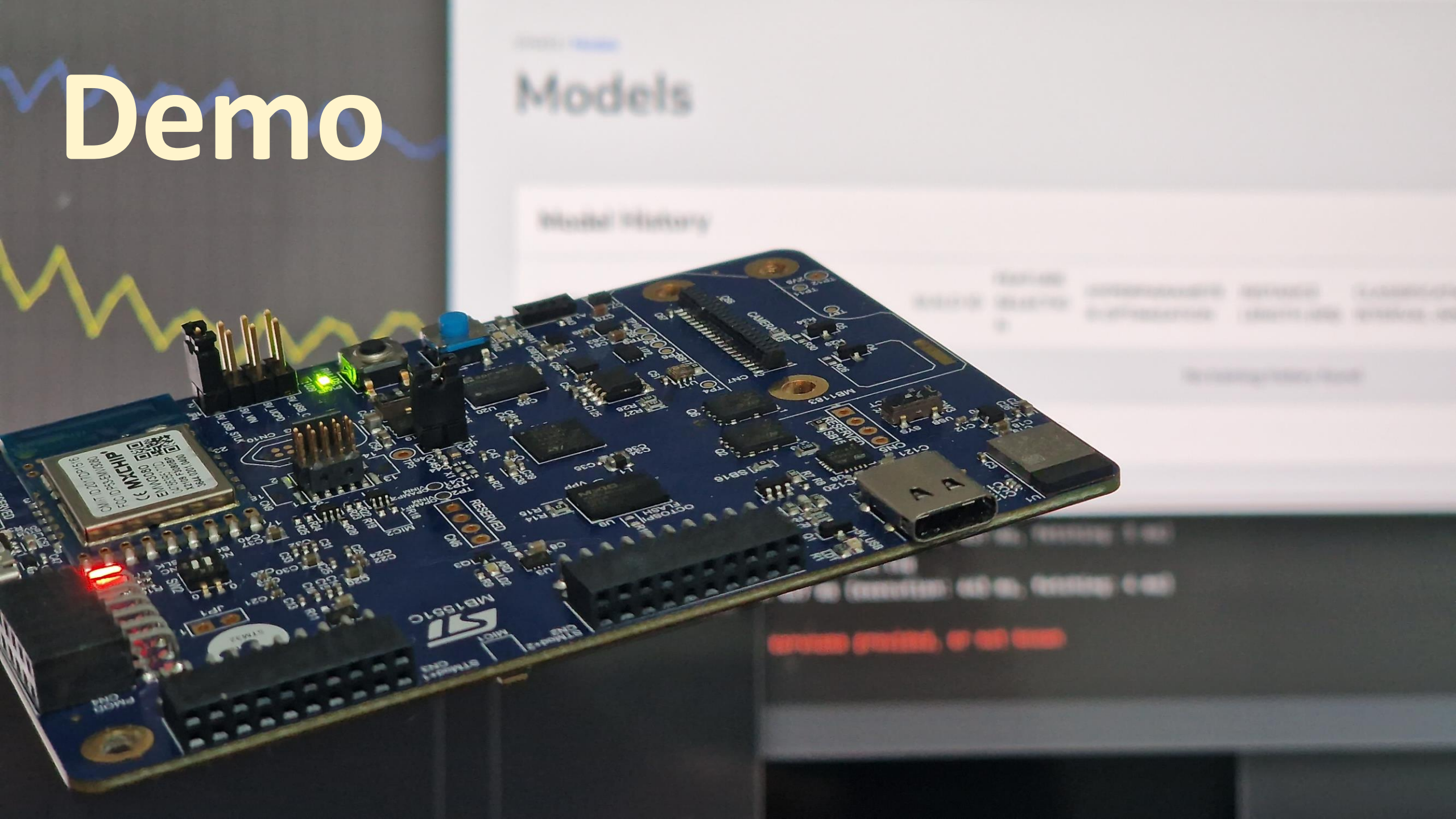
NOTE: Tools are no longer included in the CMSIS base pack

- CMSIS-DSP and CMSIS-NN already separate
- CMSIS (base pack) is just called CMSIS pack

- Validation with various tools and software integrations

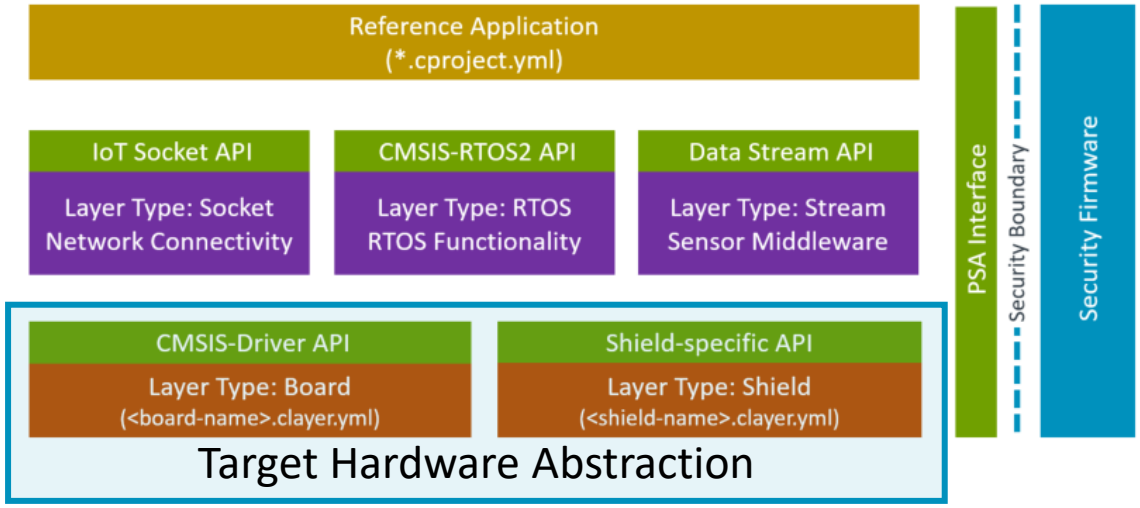
- Further improvements depending on feedback

Demo



Current Layers in more detail …

https://github.com/Open-CMSIS-Pack/RefApp-Framework



Layer-Type: ./Board + ./Shield - Target Hardware Abstraction

- Initializes used device, board, and shield (optional) hardware
- Initializes the application (calling function app_initialize)
- Starts an RTOS kernel (optional)
- Transfers control to application (entry app_main)
- Standardized interface drivers for Ethernet, I2C, SPI, UART provide connectivity for user application and layers socket and stream.
- Optionally includes CMSIS-View components (Event Recorder and Exception Fault Analysis)

Access to target abstraction via CMSIS_target_header (provides no direct device hardware access)

Note: access to device peripherals via CMSIS_device_header (should we define a CMSIS_board_header?)

Layer-Type: ./Socket – connects to network via Ethernet (using various stacks), WiFi (using chipsets), or Virtual Socket

- Optimized BSD socket for resource constrained MCU supports standard network connectivity, could be extended to Matter

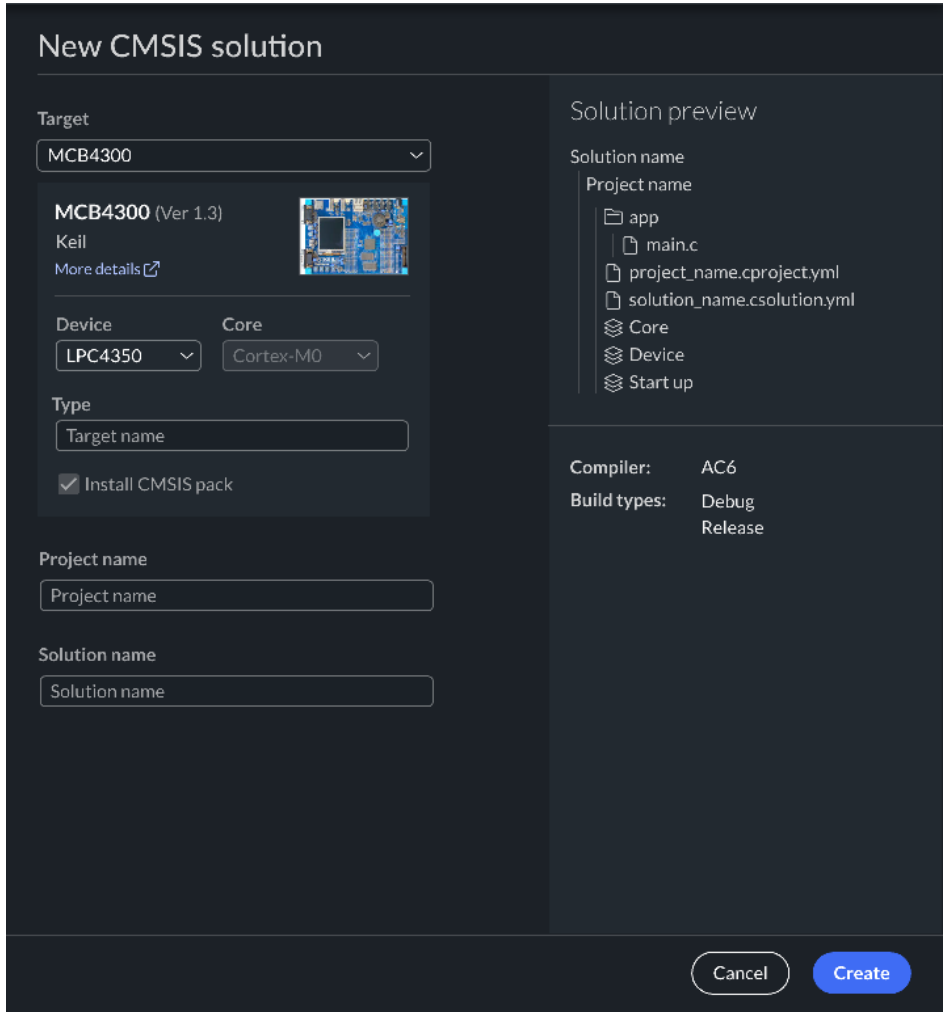
Layer-Type: ./RTOS – access to various RTOS kernels with CMSIS-RTOS2 abstraction (could be also part of Reference Application)

Layer-Type: ./Stream – provides interfaces for machine learning applications; currently in development

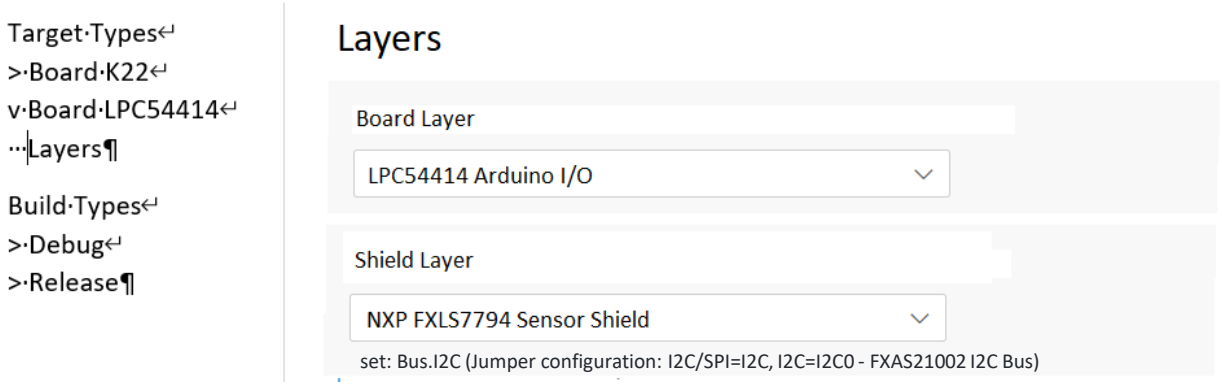
PSA Interface: - todo, a first implementation is based on STM32U5

Layers are combined based on the `connections:` described in the cproject.yml / clayer.yml files.

Potential IDE workflow for retargeting reference examples



- Add a new board
  - Review the selection of layers
  - Potentially select different layers



- Ready to Go!

CMSIS-Toolbox

github.com/Open-CMSIS-Pack/cmsis-toolbox/releases

Command-line tools – tool foundation for CLI and IDE software development flows

- Package creation and validation
  - **packgen** - create a software pack from a Cmake based software repository
  - **packchk** - semantic validation of a software pack description and the archive content
- Package management including discovery of components, devices, boards and examples
  - **cpackget** - download, add and remove packs and local repositories to CMSIS_PACK_ROOT
- Project management for constructing projects from local files and software components
  - **csolution -** manage complex applications with *.yaml user input files and content from CMSIS-Packs and output cbuild files for project build
- Project build management
  - **cbuildgen (aka CMSIS-Build)** - convert a single project context to a CMake build
- Build orchestration from solution *.yaml input to build artifact including pack installation
  - **cbuild** - convert dependant project context from the same configuration
- Package index utilities
  - **vidx2pidx** - create a flat index file from a vendor index file; a public index is maintained here: www.keil.com/pack/index.pidx; vendor index: www.keil.com/pack/keil.vidx

22'Q4 | 23'Q1 | 23'Q2 | 22'Q3 | 22'Q4 | 23'Q1

CMSIS-Toolbox 1.5.0

- Support for GCC, Arm Compiler 6, IAR
- Examples that work across toolchains
- Basic support for layers
- Initial integration into VS Code

CMSIS-Toolbox 2.0.x

- Automatic linker script generation
- Generator support finalized
- Context-map: project use across solutions
- Improved Layer handling

CMSIS-Toolbox 2.1.x

- Improved CMake integration with pre/post build steps and project dependencies
- CPRJ format deprecated

We are committed to CMSIS…

… and we will make it work for you – but we need your help

- Open-CMSIS Bi-Weekly Workshops: starting Tue 18. April (15:00 GMT)
  - **18. April:** How to create scalable software packs to maximize software re-use
  - **2. May:** Structure of Device Family Packs (DFP) and Board Support Packs (BSP)
  - **16. May:** CI test process for validation of reference applications
  - Review and evolve existing API interfaces – we need to structure taxonomies
  - Any other topics that relate to improving software re-use with packs
- PSA Certified Working Group meetings: 20. April, 18. May (16:00 GMT)
  - Approval of the PSA Certified Firmware Update 1.0 specification
  - Identifying future requirements for firmware update, and evolving the spec
- CMSIS-Stream technical details: Wednesday, 10. May (15:00 GMT)
  - Introduction to infra-structure, tools, and SDS-Framework
  - Discussion of MLOps integration and feedback on potential gaps

To get an invite to these virtual meetings send email to:

cmsis@arm.com

Next steps

How can we work together in building a portfolio of targets and applications?

- Review RefApp-Framework and provide feedback
- How to get to a standardized PSA Interface? Proposals welcome
- Can IAR add support for I/O retargeting with IAR compiler to:
  - https://github.com/ARM-software/CMSIS-Compiler/tree/main/source
- Application development has started
  - https://github.com/Open-CMSIS-Pack/AWS_MQTT_MutualAuth_SW_Framework/tree/develop
  - TFLmicrospeech for Stream layer
  - Considering Arm Model Zoo examples
  - Working on CI test process for validation with Arm Virtual Hardware
  - Working on CI test process for integration test
  - Working with NXP on sensor examples
- How can we engage with more partners?

arm

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה