An Introduction to Arm's Synchronous Data Streaming Framework

By Colin Funnell

Introduction

In the rapidly evolving world of Embedded Systems Development, the increasing use of ML has created the need to gather and process real-world data. The worlds of IoT and AI/ML both primarily concern themselves with data, and a lot of emphasis is placed upon its correct interpretation. Instead of developing processing models based on assumptions, real data from in-situ sensors can be used instead.

Arm's Synchronous Data Streaming (SDS) framework presents a ready-made and easy to use solution to address this need. In this article, we will look at the features of SDS and explore how it facilitates data transfer, processing, and visualisation.

Understanding Synchronous Data Streaming

The Synchronous Data Streaming (SDS) framework by Arm is a free to use software component available as a CMSIS Pack or through GitHub. (See Useful Links at the end of this document). The SDS framework is essentially a sophisticated data logger with record and playback features. It is easily inserted and integrated into a user application to provide high-performance data streaming between embedded systems and development environments.

By being inserted between an embedded device sensor and its on-chip processing, a tap-off point can be exposed. In one direction, this data is now readily available to be recorded along with timing information on a development PC.

In a playback context, your embedded sensor can be replaced by a data stream of previous recordings or synthetic signals. Without knowing the difference, your embedded core algorithm can be virtually exposed to many more data sample sets than a manual set-up on the bench in a lab can provide as stimulus.

It essentially creates pathways for real-time data capture and replay, ensuring that data flows at a regular rate, as if gathered by sampling. A PC acts as an intermediary.

SDS vs CMSIS Stream

With both having 'stream' in their name these two terms often get mixed-up, but they offer very different functionality.

CMSIS Stream:

- Primarily designed for signal processing applications.
- Focuses on digital signal processing (DSP) tasks.
- Optimises data flow between algorithm stages in a DSP pipeline.

CMSIS SDS:

- A broad framework for data capture, analysis and replay.
- Supports a variety of data types and sources.
- Ensures synchronous data transfer with minimal latency.

CMSIS Stream won't be discussed further here but it is useful to be aware that its data-graph nature makes it useful for complicated DSP tasks. It works along with SDS - one as an element that processes data and the other moving that data in and out of a development system.

CMSIS SDS is available as a pack download. It includes code to incorporate into your project and supporting Python code to run on your PC.

Connection with SDS (SDSIO / SDS-Server)

SDS supports multiple connection types, enhancing its flexibility and usability in different scenarios. Connection types include:

- Serial Ports: Standard RS232-style or USB serial connections.
- Sockets: Network-based connections of data transfer.
- File System: Local file storage instead of communication streams.

The SDS IO component is a C module provided in the CMSIS pack that you integrate into your project. It provides a data source/sink that then connects to either a communications channel or storage. These interfaces make use of the MDK-Middleware components for easy integration. Your time is better spent on your application instead of a side-quest of a better development system.

SDS-Server is a Python script that runs on a PC. It operates over either a serial or network connection to interact with the embedded target. The target itself can open multiple data streams that are captured in parallel by the SDS-Server. A unique set of data files are created for each session. It can easily build-up a large collection of data, so it is important to be able to tag data as you go along in true scientific experiment fashion. This is vital for the management of large datasets created during the product development and for continuing logging of data from live systems.

A serial port is handy for desk-based development, network is better for handling larger-scale testing. For devices that are tucked-away in their end-use environments, storing sensor data on a local SD card provides the means to gather a lot of data which can then be physically collected at a later date.

Viewing and Converting Data on a PC (SDS-View / SDS-Convert)

Once data is captured using SDS, it can be viewed or converted on a PC for further analysis. SDS provides tools and libraries to facilitate this process, enabling seamless data conversion and visualisation.

SDS-View is a Python script which takes a set of recorded data along with a YAML description of the sensor data timing to provide 2D or 3D plots. Plotting is done with the Python Matplotlib module, so the viewer code can be modified to give a wide range of data representations.

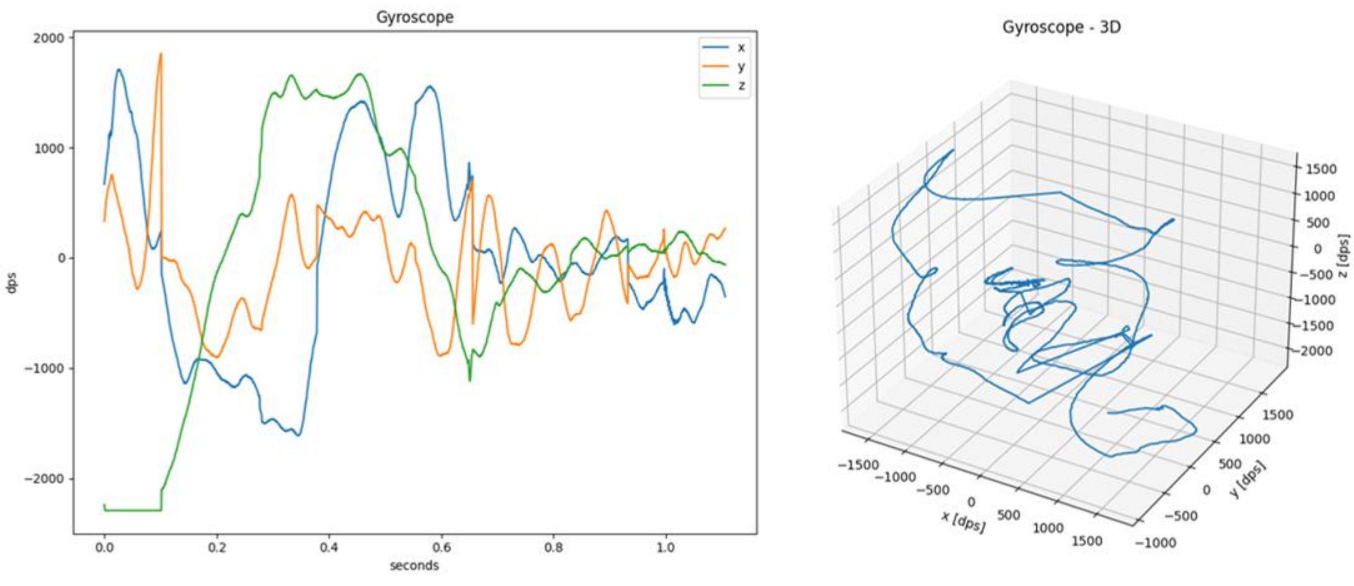

Figure 1 - SDS-View outputs

The viewer itself may not provide the analysis and manipulation you require of your data. For this, the SDS-Convert script is provided which creates a standard RIFF/WAV file using linear PCM encoding. Information in a sensor description YAML file allows it to be converted to the correct sample rate. It makes your data universally available to a wide range of software. PC tools such as Audacity which would normally be used for sound sample processing can now be harnessed for filtering and manipulation.

Data can also be converted into CSV data format but can only handle a single channel of data.

Since data can be at the heart of a system, it is important to ensure what you're working with is valid. SDS-Check is a script which performs checks with the internal timestamps for data consistency.

CMSIS Neural Net (NN) and ML-Ops with SDS

The wider Arm ecosystem has multiple technologies which can assist the modern developer on AI/ML projects.

CMSIS-NN provides a collection of optimised Neural Net kernels that can be used with most ML frameworks. Although they can be used independently by a developer, its power is shown by being able to utilise different realisation schemes based on the target silicon without the need for bespoke customisation. It'll make best use of what's available. A separate driver and compiler are also available to translate models for use with the latest Ethos NPUs from Arm. These units go beyond providing extensions which assist in processing data to highly performant data processors in their own right.

All the above is supported by Arm Virtual Hardware (AVH) – high-speed software simulation that can be run locally or in the cloud. Much ML work can be tested and verified without the need of final hardware designs. Through modern cloud-based workflow such as Arm's MDK-6 with GitHub runners, the ML-ops cycle can be highly automated.

Where SDS really comes into its own is with the Machine Learning development cycle. Here there is a need to gather as much training data as possible, convert it into a useable model, verify that the reduced model for embedded systems provides suitable results - before and after deployment. With SDS, pre- and post-processed data can be gathered for evaluation, verification or further training.

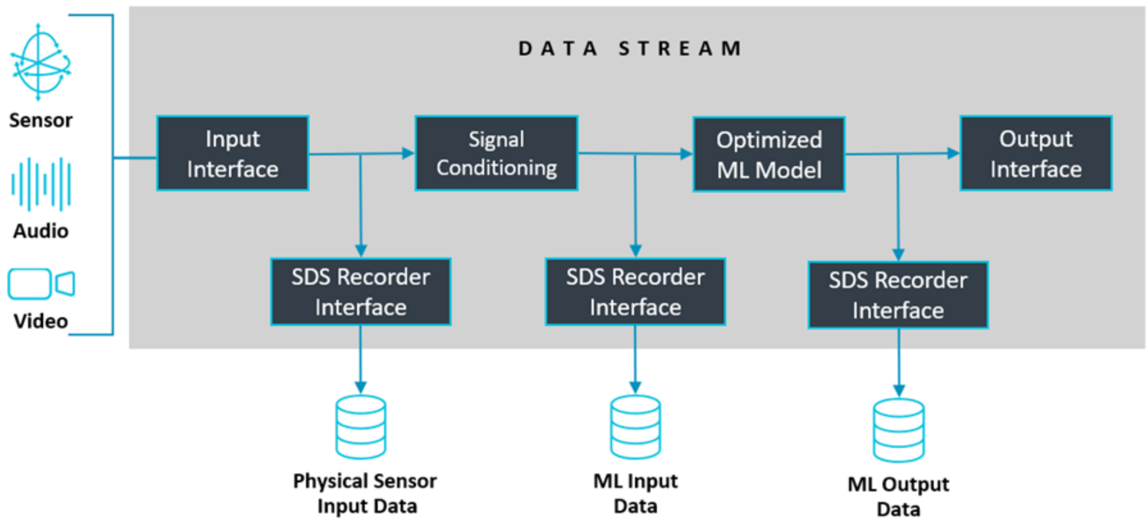

Figure 2 - Taps in an AI/ML inference device

Replay and Simulation

SDS not only supports data capture from embedded systems to host applications but also allows playback of recorded data through the sensor streaming interface. The SDS Playback interface allows data flow back to the embedded system, as if it were data from a sensor.

This is invaluable for detailed regression testing. A system can be exercised with a whole range of pre-recorded data to show correct operation. A captive unit can now be virtually exposed to all sorts of conditions.

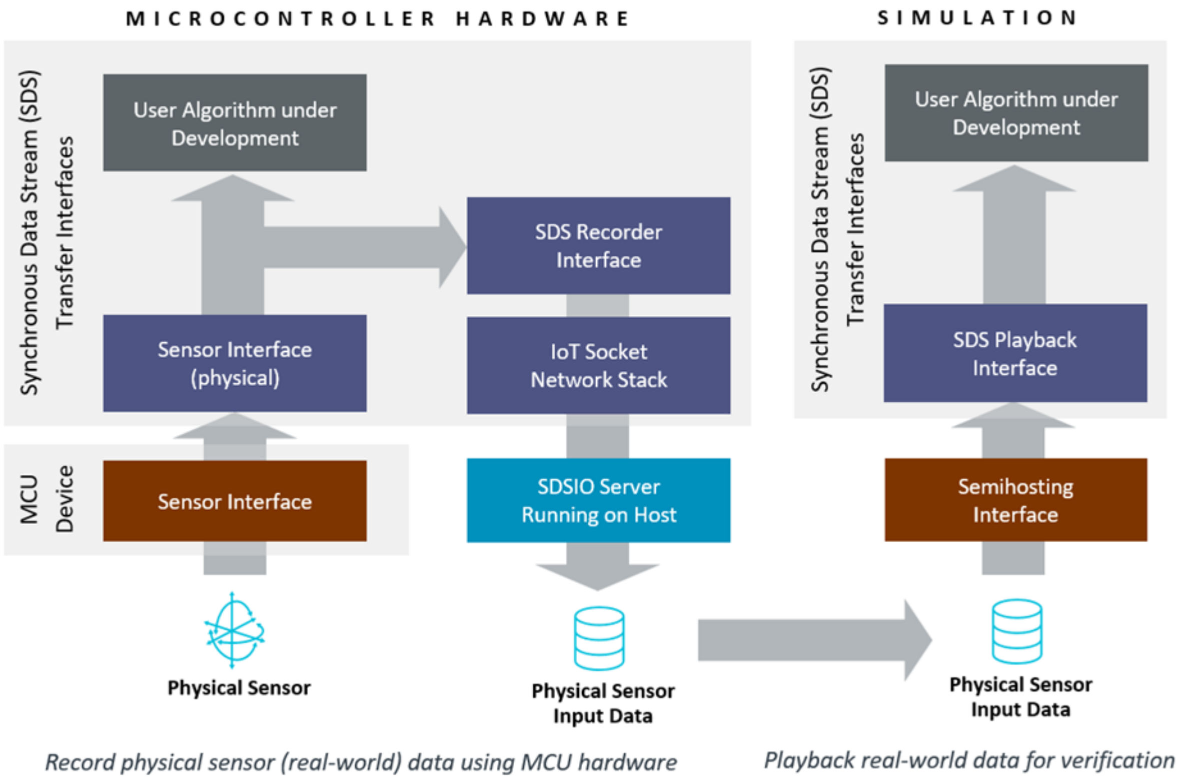

Figure 3 - SDS data transfer between a real and virtual device

The above diagram shows 'semihosting' – a bridge between simulation and host - a topic for another day.

Taking this further, Arm Virtual Hardware (AVH) can be used as a simulator to design and evaluate software before any hardware is created. With a collection of sensor data, optimisation can be done to squeeze any data processing algorithms and check performance. The target hardware and core can be chosen much more selectively with a wealth of knowledge in-hand – knowing it hasn't been over-specified.

SDS 2.0.0

During the writing of this article the SDS CMSIS pack has moved up to version 2. It has been generally enhanced with the following:

SDS file I/O interface extended with USB bulk communication

Concurrent recording and playback of multiple data streams

Test framework template as reference application:

- Data Test for communication and bandwidth verification
- Algorithm Test for algorithm development and testing
- Verification using different hardware with Board layers
- Verification using simulation models with AVH FVP

Improved Python utilities:

- SDSIO-Server for capturing SDS data files via multiple interfaces

SDS-Check to validate SDS data files

Extensive documentation:

- github.com/Arm-Examples/SDS-Examples show multiple use cases on different hardware

Version 2 makes more use of the development principles found in the MDK-6 workflow. It makes use of software layers, and the template example is in the. csolution format.

In Summary

In short, Arm's Synchronous Data Streaming framework is a powerful tool for sensor data-centric designs. As a tap-point to the sensor data flow, it opens-up many development paths to intercept, store and replay data easily and conveniently.

It can bridge the gap between real and virtual devices – providing the latter with realistic situations. As a step in regression testing, pre-recorded scenarios can be played out repeatedly and precisely.

Access to data is essential for AI/ML workflows and SDS provides a standard method to interface to this. Developers can spend more time in comfort at their desks and concentrate on what the data has to say instead of trying to approximate matters in the lab.

Useful links

SDS CMSIS pack:

https://www.keil.arm.com/packs/sds-arm/versions/

Arm SDS-Framework on GitHub:

https://github.com/ARM-software/SDS-Framework

Introduction to MLOps with GitHub runners:

https://learn.arm.com/learning-paths/servers-and-cloud-computing/gh-runners/

Further Information

For more information visit our Knowledge Base, website: www.hitex.co.uk or get in touch: info@hitex.co.uk. You can also connect with us: LinkedIn

Sign up to receive Tech Tips directly to your inbox. Never miss a trick!