EMBEDDED MICROPROCESSOR BENCHMARK CONSORTIUM

Measuring ML Energy Efficiency of Embedded Devices

An overview for ULPMark™-ML and tinyMLPerf

Peter Torelli, President, EEMBC
peter.torelli@eembc.org

slides version 2021.04.26.002

What is EEMBC

Formed in 1997, EEMBC is a non-profit consortium of semiconductor manufacturers, integrators, and academics that develops industry-standard performance and energy benchmarks for embedded applications.

This project is a collaboration between EEMBC and MLCommons

https://www.eembc.org

Objective

What you will learn:

- How to measure the energy consumed by a neural net model on an embedded platform.

Prerequisites:

- You should already have ported the benchmark firmware to your test platform and understand how to use the Host Runner software to measure performance (see "Part 1" slides).

Terms

Framework - the ecosystem of hardware, software and firmware used to execute benchmarks and take measurements

Host PC - the computer that runs the benchmark software

Runner - host application software used by a developer to run the benchmark

EMON - Energy monitor, provides and/or measures energy consumption

DUT - Device under test, the thing we are measuring

IO Manager - A device used to electrically isolate the DUT from the Host PC

Important link

https://github.com/eembc/benchmark-runner-ml

Contains links to the runner, datasets, videos, and firmware source.

Basic concept

*Iterate inference in a **loop** with the **same input dataset** for a **significant period of time** and **measure energy consumption**.*

Loop: required because a single inference is too fast for the tolerance of the system

Same input dataset: cannot universally feed data to any DUT; benchmark is designed to be architecture agnostic; same dataset removes noise due to data-dependent execution

Significant period of time: runtime must exceed min. tolerances of measurement hardware and must be long enough to amortize any start-up costs in the DUT; DUT must achieve steady state

Measure...: min. 0.1uA accuracy required by the EMON; EMON *internal* accumulating sampling rate must exceed power delivery capacitance time-constant so as to not miss di/dt spikes

It takes a lot of components & steps to do this in a portable and controlled way! That's benchmarking...

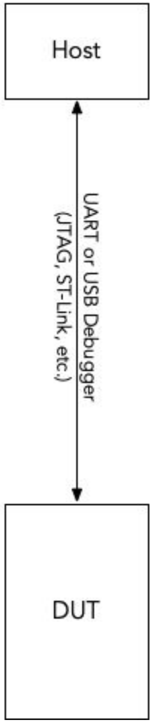
Performance mode

DUT connects directly to the Host PC

Basic text protocol between Host PC and DUT over UART allows downloading data and configuring execution

Baud rate can be set by altering the firmware and the Host Runner initialization file

The basic requirement for tinyMLPerf



EMBC EMBEDDED MICROPROCESSOR BENCHMARK CONSORTIUM

Energy mode

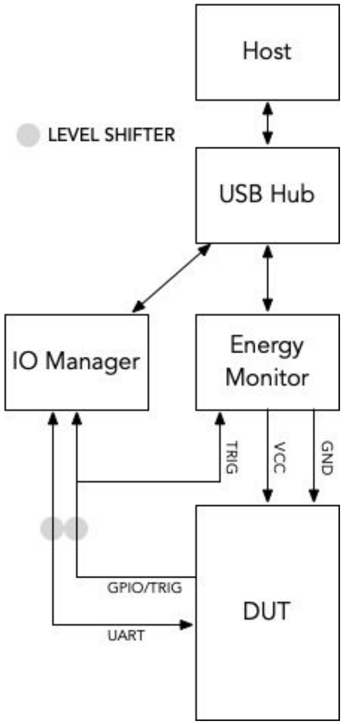
DUT is isolated from Host PC

Why? ... Without isolation the DUT would source and sink current and voltages wouldn't match, making energy measurement impossible

Energy Monitor supplies energy at native MCU/SoC voltage instead of the DUT USB interface using on-board power supply

IO Manager passes UART commands to DUT through level shifters to match USB voltage with DUT IO voltage (baud = 9600*)

** Why 9600? Historical reasons. 1) compatibility: many devices only support low baud; 2) cost: Arduino UNO is cheap & uses software serial; 3) independence: Arduino isn't an EEMBC member; 4) original bmarks sent few hundred bytes of config, not 10's of KB. May revisit in future versions.*

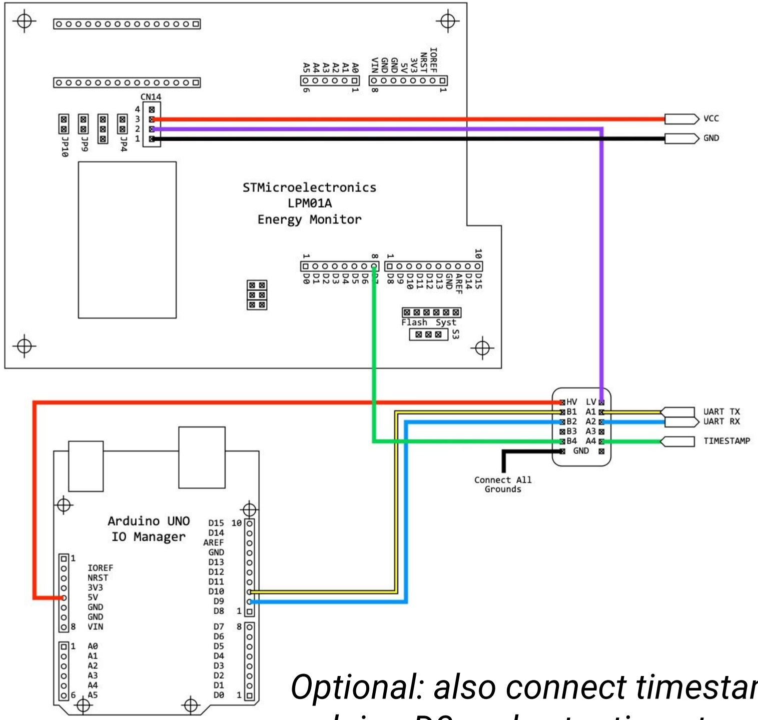


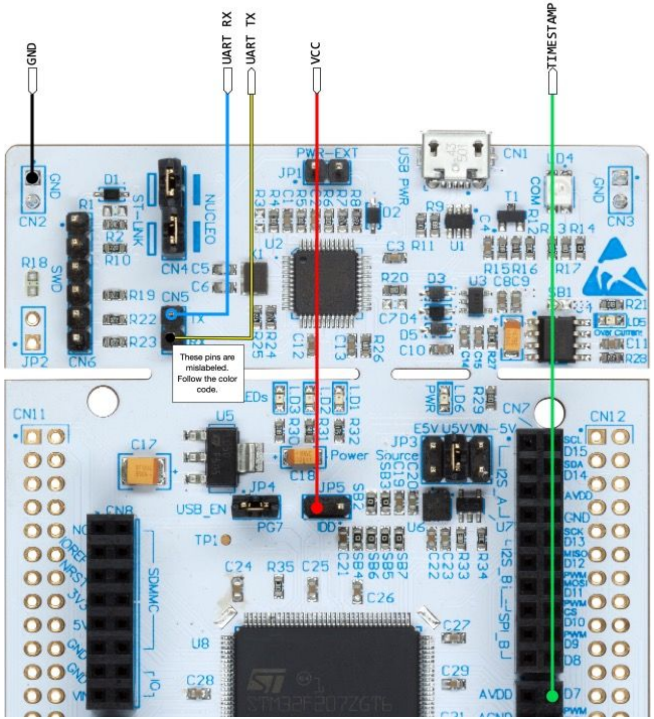
Setup steps

1. Finish performance mode porting
2. Configure a GPIO pin as PP or OD and reset for 1us in "th_timestamp", then set
    a. This hold time works for a range of energy monitors
3. Set #define EE_CFG_ENERGY_MODE 1
    a. This should switch the UART to 9600 baud and switch the timestamp from reporting the MCU timer to pulling a GPIO pin low & recompile
    b. You could also use another GPIO to switch modes to avoid having to recompile & flash, but the # of available IOs is not a guarantee
    c. (2021-04-08 not all ref. implementations have this code yet, see ad01 & vww01)
4. Connect the following hardware components in the next slides
5. Select "ML Energy Mode" in the Host Runner

That's it.

Hardware connectivity for LPM01A

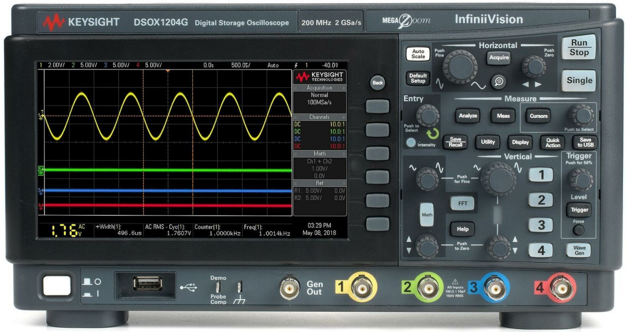




Optional: also connect timestamp to arduino D3 and extra timestamp info is printed during the run.

Nice-to-have lab tools

For debugging UARTs



For debugging timestamps

Runner GUI

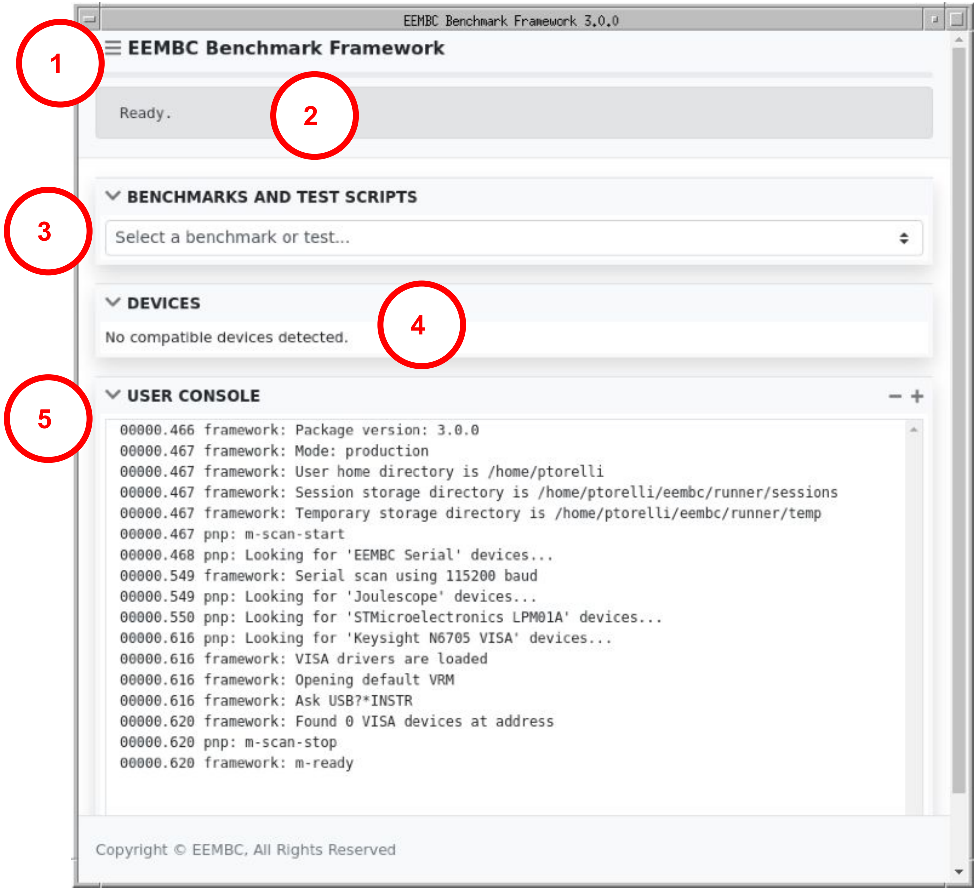
1. Load emon data / reload session / exit

2. Status bar

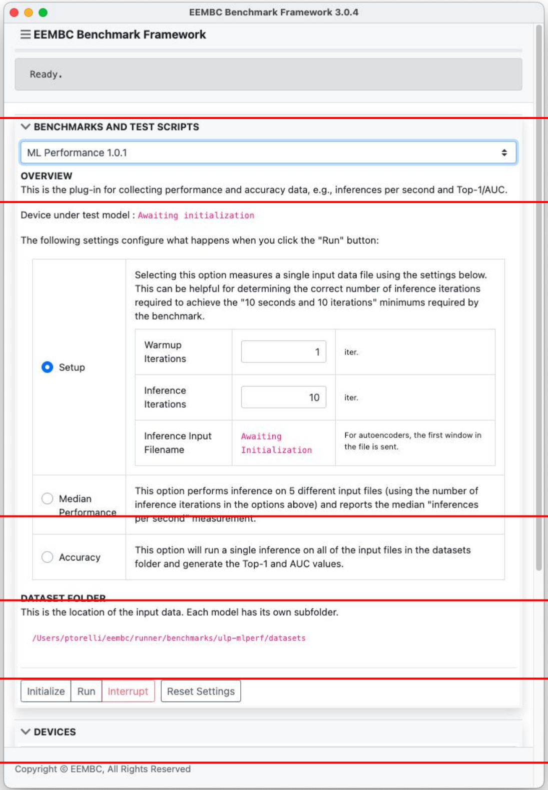
3. List of available benchmarks (varies based on benchmark, others exist; comes with ML installed)

4. Device detection window for mounting / unmounting

5. User console displays messages, manual command entry (detailed docs WIP)



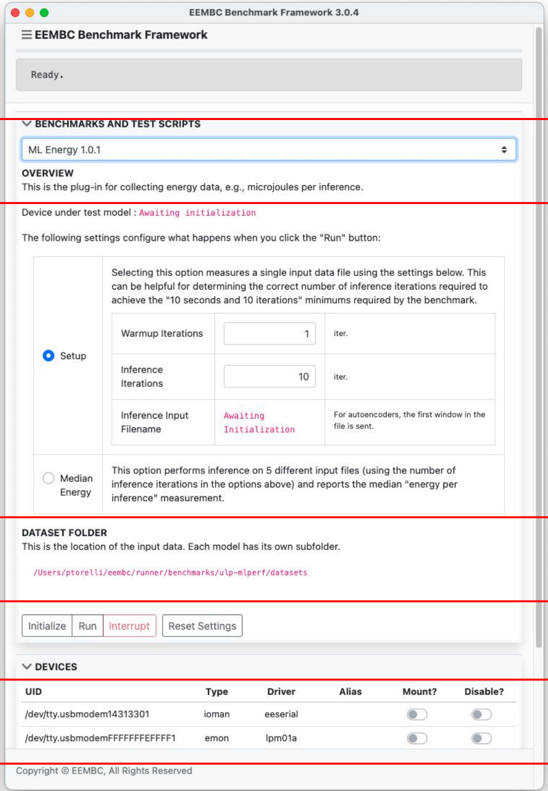
Host runner differences, performance vs. energy



Different plug-in

No "Accuracy" Mode

DUT vs. EMON+IO



What does the runner actually do?

It communicates with the EMON, IO Manager and DUT to coordinate the execution of the benchmark

It uses a simple text-based protocol via serial port, USB, or VISA to serialize asynchronous events

Every component has a command-set

A benchmark is just a series of commands that trigger a linear progression of events and measurements

It can run GUI mode or "headless" mode for automation (documentation WIP)

Detect - identify compatible hardware connected to the system

Initialize - query the hardware for configuration information (e.g., NN model)

Run - execute the instructions in the main benchmark script (turn on the emon, start tracing, listen for timestamps, download data to the DUT, etc...)

Post-process - read the log and energy files to determine correct execution of the benchmark and extract measurements

Reload - reload and analyze data from previous session (sessions stored in $HOME/eembc/runner/sessions)

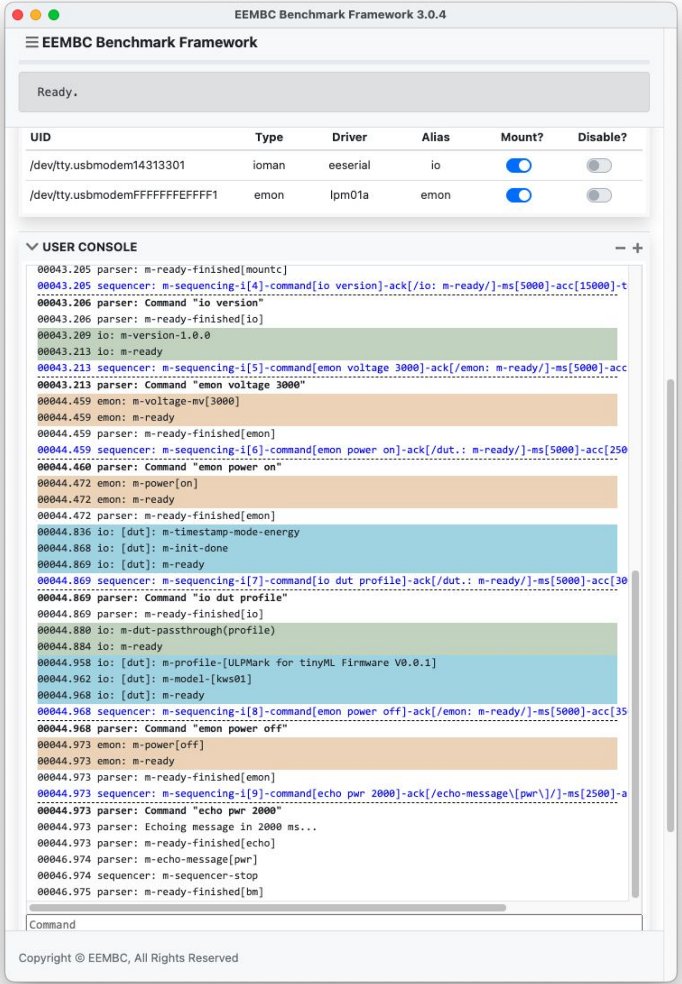
What happens in initialization?

All of the run scripts are pre-written to describe the correct benchmark initialization and execution

Here is initialization (not shown, mounting the devices)

1. ask IO Manager version (for compatibility)
2. set voltage
3. power-up & wait for device to boot
4. ask for neural net model
5. power down

Note the sequencer commands: each command waits for a completion regex



EMBC EMBEDDED MICROPROCESSOR BENCHMARK CONSORTIUM

What happens during a run?

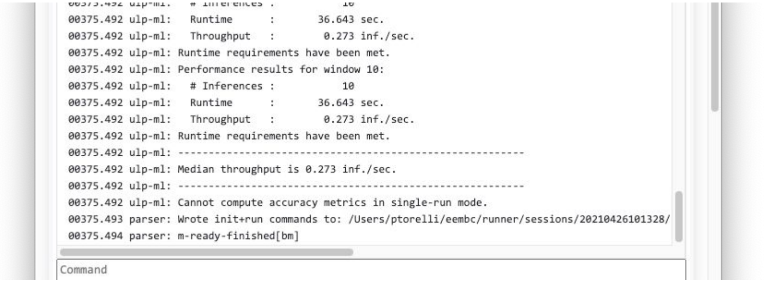
1. Device is powered on & energy trace started
2. A timestamp is issued (for synchronization purposes)
3. The input file is downloaded from the Host PC to the DUT via repeated "db" commands
    a. See the firmware GitHub README for explanation of "db" sequence[1]
    b. Commands NOT printed to console (too many)
4. The "infer" command is sent to DUT
5. *N* warm-up inferences
6. Timestamp
7. *M* measured inferences
8. Timestamp
9. Power down
10. Post-process

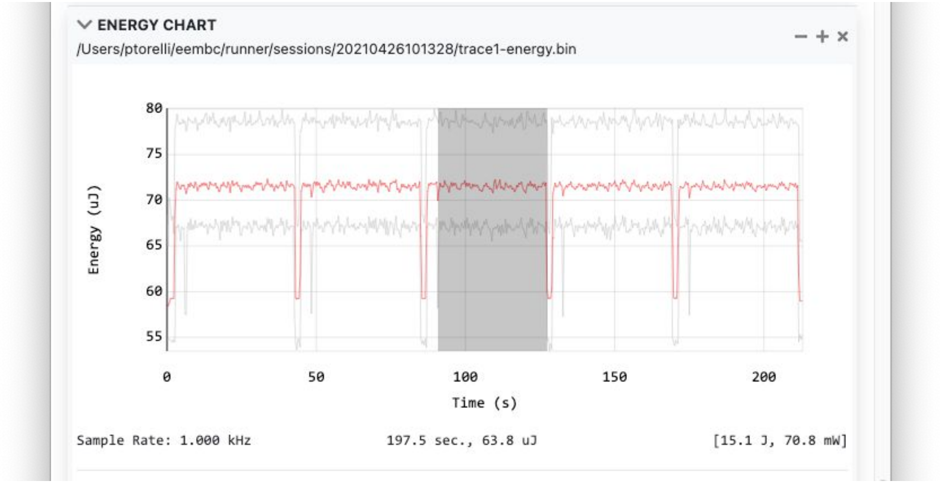
```c
void
ee_infer(size_t n, size_t n_warmup)
{
    th_load_tensor(); /* if necessary */
    th_printf("m-warmup-start-%d\r\n", n_warmup);
    while (n_warmup-- > 0)
    {
            th_infer(); /* call the API inference function */
    }
    th_printf("m-warmup-done\r\n");
    th_printf("m-infer-start-%d\r\n", n);
    th_timestamp();
    th_pre();
    while (n-- > 0)
    {
        th_infer(); /* call the API inference function */
    }
    th_post();
    th_timestamp();
    th_printf("m-infer-done\r\n");
    th_results();
}
```

[1] https://github.com/eembc/testharness-ulpmark-ml

Interpreting results

The "Median Energy" option reports the median uJ/inference of five runs with five different inputs. Each run must take at least 10 seconds or ten iterations.





EMBC EMBEDDED MICROPROCESSOR BENCHMARK CONSORTIUM

Run rules

RUN RULES			
Run Rule Type	**ID**	**Rule**	**Justification**
General	1.1	Scores are only valid if collected with the EEMBC Host Runner software.	To ensure the test was run according to specification.
General	1.2	Energy, performance, and verification (EPV) must use the same firmware.	To ensure the three different scores are consistent.
General	1.3	Verification score must be within Accuracy % and AUC margins-of-error (TBD, based on joint task-force analysis).	To ensure optimizations have not degraded the model's accuracy.
General	1.4	The DUT must be "typical" power SKU, e.g. a median of a large sample of publicly available parts.	To avoid picking ultra-rare parts for higher scores.
General	1.5	The DUT hardware must be publicly available.	To ensure that the hardware is available to anyone.
Electrical & Environmental	2.1	Only one power supply is allowed into the DUT for energy mode.	To prevent escapee current paths that would artificially reduce power.
Electrical & Environmental	2.2	Any energy used during the benchmark must be drawn from the energy monitor.	To prevent attempts at hiding the amount of energy used (e.g., via supercapacitor).
Electrical & Environmental	2.3	Minimum 21C ambient temperature.	To prevent thermal-related optimizations (e.g., leakage sensitivity).
Electrical & Environmental	2.4	The board may be modified by cutting traces, removing jumpers, or desoldering bridges to remove ancillary components that may increase	To allow the user to remove unused platform hardware to obtain a more realistic score.
Firmware	3.1	Only "th_" functions may be modified in the firmware's source code.	To prevent the user from modifying the behavior of the benchmark.

https://www.eembc.org/ulpmark-ml/run-rules/

Submitting your score

Use the score submission form at this page:

https://www.eembc.org/ulpmark/ulp-ml/submit.php

You will need to create a generic account first:

https://www.eembc.org/user/create.php

And confirm via email.

The Submission page is very similar to the JSON submission form for tinyMLPerf with a few notable exceptions...

Submission disclosure

The firmware for the EE_CFG_ENERGY_MODE=1 must be supplied; any user should be able buy a board, configure the framework, and generate the same score.

- Since the tinyMLPerf requires the source to be available, this is a no-brainer, but EEMBC does not require source to be published

A document is required explaining how to connect the board to the test framework; such as GPIO pins, VCC, jumper configuration, etc.

Q&A

Questions

2021-04-08: What other EMONs do you support if the DUT draws > 50mA?

A: JS110 and N6705 are natively supported. See this section on the github page for how the schematic changes: https://github.com/eembc/benchmark-runner-ml#energy-mode-hardware