# An Experimental Setup to Evaluate RAPL Energy Counters for Heterogeneous Memory Experiments Reproduction Guide

Lukas Alt

February 2024

In this document, we describe how the experimental setup and the measurements for the RAPL validation can be reproduced. First, the DIMM risers need to be installed in the designated memory slots, and the MCC128 needs to be mounted on the Pi while following applicable ESD guidelines. Then, all wiring between the risers, the current-sense amplifiers, the MCC128, and the MCP2221 need to be established according to the provided schematic. Some photos of the setup can be found in `experimental_setup` folder of the paper artifact.

## Software Setup on the Raspberry Pi

The following steps need to be followed to set up a Raspberry Pi 3B or later for power and energy measurements.

- Power the Pi using a recommended power supply. We used the official one. Powering via a standard USB port usually does not provide enough current and can lead to an undervoltage.
- Install any Linux installation on the Pi. We used Raspberry Pi OS for this thesis.
- Install python3, e.g., `sudo apt install python3 python3-dev python3-pip` and `sudo apt install screen`
- Install required python packages: `pip install pandas numpy pyserial RPi.GPIO`
- Copy the `sourcecode/power-collector/` folder from the thesis artifacts to the Pi.
- To capture power consumption traces, run `screen -dmS power-measurements python3 power-collector.py`. If the accumulated energy (at a higher internal sampling rate) should be measured only, run `screen -dmS power-measurements python3 power-collector.py` instead.

- Both scripts require a configuration containing which voltages and signals from which DIMM are connected to the input pins of the data acquisition device.
  Different profiles can be configured in `power-collector/scripts/channel_-config.py`. The profile can then be configured in the global variable `channel_config` in the collector scripts.
- The sampling frequency can be controlled by setting the `scan_rate` variable, which defines the number of samples per second, in the collector scripts.
- Both scripts act as agents and do not need to be explicitly controlled on the Pi once they are started.

## Software Setup on the Machine under Test

The following steps need to be followed to set up the host system for lining up the power measurements taken by the Pi with the execution of applications on the host system.

- Ensure a Linux installation with a recent kernel is available and that root access is available.
- Install `numactl`, `libnuma`, `python3`, `python3-dev`, `python3-devel`, `hi-dapi`. All installed `RHEL 8` packages can be found in `experimental_-setup/system-info/yum_packages.txt`.
- Install the python packages `pip install numpy pandas PyMCP2221A hidapi==0.14.0 libusb pyserial`. In our case, the installation of `PyMCP2221A` was challenging and required a couple of other dependencies. A list of all installed Python packages can be found in `experimental_-setup/system-info/pip_packages.txt`.
- Copy the `sourcecode/` folder from the artifact to the system.
- Use `make` to compile the RAPL measurement tool inside `sourcecode/rapl-trace/`.
- Switch to the `workloads` folder and run `make`.
- Update the path to the `workloads` folder in the `workloads/run.sh` script.
- Run the `measure-all-optane.sh` and `measure-all-dram.sh` scripts to run the RAPL validation experiments we have evaluated in our experiments for PMM and DRAM, respectively.
- Run `python3 measure-power-rapl.py` to collect RAPL measurements and reference measurements of the specified workload. A description of the parameters can be requested by specifying the `-h` flag.

The RAPL validation scripts will create a folder for each workload with the reference measurements on the Pi and the RAPL measurements on the system under test. For further processing, both folders need to be merged, and a `config.json` file that contains the mapping of the different voltage channels to the pins and risers needs to be created. The existing configs in the `data/icelake` folder serve as a baseline. A Jupyter notebook that creates the figures from this raw data can be found in `data/Visualize RAPL Results.ipynb`.

## Required Changes for DDR5 Measurements

Our setup can be modified to measure the power consumption of DDR5 DIMMs. As the DDR5 interface is incompatible with DDR4, our riser cards do not work here, and DDR5-compatible alternatives can be used. Currently, we only found the DDR5-R riser from Adex Electronics[1], which, however, only supports 4800 MT/s memory. Also, validate that the current-sense resistors are rated for the voltage and current through them (see paper section 4.5). If too much heat is dissipated at the shunt, the riser, the memory module, or even the mainboard can be damaged. Registered DDR5 DIMMs are powered using a single 12 V supply, and unbuffered DDR5 memory is powered using 5 V. Thus, only a single current-sense amplifier (ideally with a gain ¿= 200) and two data-acquisition channels are required for measuring the power consumption of a single DDR5 DIMM. If the bus voltage (e.g., 12V for RDIMMs) exceeds the maximum input voltage of the DAQ device, a voltage divider (see schematic) is required to step down the voltage to fit into the supported range.

## IT Setup and Configuration

Due to the required space for the hardware instrumentation, the server could not be operated in the rack in the server room. Thus, we moved it to another room where we set up our measurements. As no network connection to the cluster infrastructure was available in this room, a local network using an ethernet switch was set up. The Pi was used to bridge WiFi to this local network, and we connected the server to the switch via a USB to Ethernet adapter as the Ice Lake system did not have an RJ45 public network port. The management port was also connected to the switch to access the IPMI.

Because the server was not connected to the cluster anymore, the distributed file system and the local operating system of the server could not be used. Thus, we had to install our own operating system, which did not work at first as the server did not detect any USB boot devices. After installing Rocky Linux on an old SATA SSD and connecting this SSD via a SATA to a USB adapter to the server, the system correctly detected it and booted successfully. We installed PAPI v7.0.1 libnuma v2.0.16, and libmemkind v1.14.0 from sources.

Do not hesitate to contact the author if questions or problems arise during the reproduction of the experiments.

[1] `https://adexelec.com/ddr5-r`