



EBRAINS

BrainScaleS-2

Jakob Kaiser (UHEI)

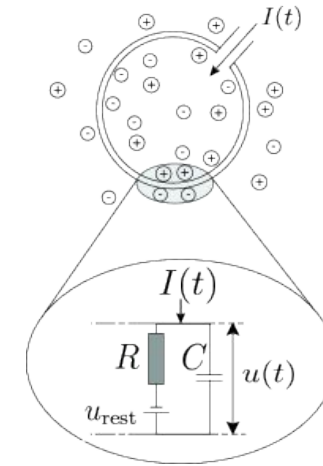
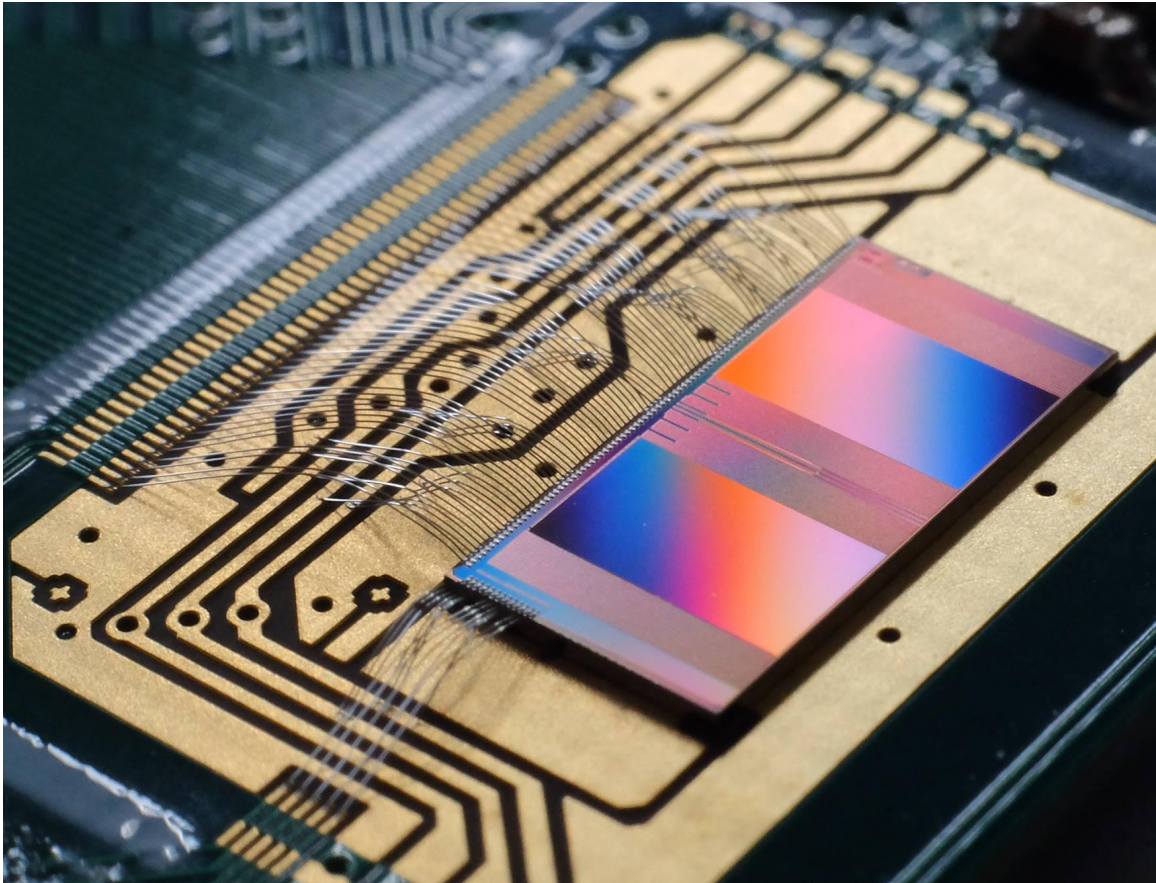
EBRAINS Tools for Teaching: Leveraging EBRAINS Open Science Tools for Neuroscience Education
EBRAINS Tutorials and Users Day, 12 March 2025 | Heidelberg, Germany



Co-funded by
the European Union

BrainScaleS-2 for Teaching

Physical emulation of neuron and synapse dynamics



- Python interface (PyNN)
- Collection of educational jupyter notebooks
- Access via EBRAINS

Use Cases: Workshops

The screenshot displays the EBRAINS Collaboratory web interface. At the top, there's a navigation bar with links for Collabs, Documentation, Support, Forum, and a search icon. A user profile icon and a 'Log-out' button are on the right. Below the navigation bar, the breadcrumb path is '/ Collabs / BSS-2 Demo / Lab'. The main interface is a JupyterLab environment. On the left, a file browser shows a directory structure for 'BSS-2 Demo / brainscales2-demos /'. The central pane shows a Jupyter notebook titled 'ts_00-single_neuron.ipynb'. The notebook content includes a title 'BrainScaleS-2 single neuron experiments', a paragraph about setting environment variables, and two code cells. The first code cell imports 'setup_hardware_client' from 'from_static.common.helpers'. The second code cell imports various libraries like 'numpy', 'matplotlib', and 'pyNN', and defines a function 'plot_membrane_dynamics'. The bottom status bar shows 'Simple' mode, '2' tabs, '1' kernel, and 'EBRAINS-experimental | Idle' with memory usage 'Mem: 273.59 / 2048.00 MB'.

EBRAINS Collaboratory

Collabs Documentation Support Forum Q ⌵ ⌵ Log-out

/ Collabs / BSS-2 Demo / Lab

Open in Lab

File Edit View Run Kernel Git Tabs Settings Help

Mem: 279 / 2048 MB

Filter files by name

/ ... / BSS-2 Demo / brainscales2-demos /

| Name | Last Modified |
|----------------|----------------|
| _static | a day ago |
| common_... | 19 hours ago |
| common_... | a day ago |
| fortgeschri... | a day ago |
| fp_binary_... | a day ago |
| fp_pynn_in... | a day ago |
| fp_sudoku... | a day ago |
| girlsdai.ip... | a day ago |
| gs_01_ne... | a day ago |
| gs_02_pyn... | a day ago |
| gs_03_sin... | a day ago |
| gs_04-1_... | a day ago |
| gs_04-2_... | a day ago |
| gs_05_ma... | a day ago |
| gs_06_logi... | a day ago |
| index.ipynb | a day ago |
| nmpl_00-n... | a day ago |
| spiking_co... | a day ago |
| spiking2_c... | 22 minutes ago |
| tp_00-intro... | a day ago |
| tp_01-prop... | a day ago |
| tp_02-yin... | a day ago |
| ts_00-singl... | seconds ago |
| ts_01-sup... | a day ago |
| ts_02-plast... | a day ago |
| ts_03-mult... | a day ago |
| ts_04-mc... | a day ago |
| ts_05-yin... | a minute ago |
| ts_06-dyn... | a day ago |
| ts_07-pon... | a day ago |
| ts_08-ade... | 2 hours ago |
| tutorial.ipynb | a day ago |

BrainScaleS-2 single neuron experiments

In order to use the microscheduler we have to set some environment variables first:

```
[ ]: from_static.common.helpers import setup_hardware_client
setup_hardware_client()
```

```
[ ]: %matplotlib inline
import numpy as np
from ipywidgets import interact, IntSlider
from functools import partial
IntSlider = partial(IntSlider, continuous_update=False)
import matplotlib.pyplot as plt
plt.style.use("_static/matplotlibrc")

import pynn.brainscales.brainscales2 as pynn
from pynn.brainscales.brainscales2 import Population
from pynn.brainscales.brainscales2.standardmodels.cells import SpikeSourceArray
from pynn.brainscales.brainscales2.standardmodels.synapses import StaticSynapse

def plot_membrane_dynamics(population: Population, segment_id=-1, ylim=None):
    """
    Plot the membrane potential of the neuron in a given population view. Only
    population views of size 1 are supported.
    :param population: Population, membrane traces and spikes are plotted for.
    :param segment_id: Index of the neo segment to be plotted. Defaults to
    :param segment_id: -1, encoding the last recorded segment.
    :param ylim: y-axis limits for the plot.
    """
    if len(population) != 1:
        raise ValueError("Plotting is supported for populations of size 1.")
    # Experimental results are given in the 'neo' data format
    mem_v = population.get_data("v").segments[segment_id].irregularlysampledsignals[0]

    plt.plot(mem_v.times, mem_v, alpha=0.5)
    print(f"Mean membrane potential: {mem_v.mean()}")
    plt.xlabel("Wall clock time [ms]")
    plt.ylabel("ADC readout [a.u.]")
    if ylim:
        plt.ylim(ylim)
```

In this part we will explore some of the spiking capabilities of the BrainScaleS-2 neuromorphic accelerator using our implementation of the pyNN interface.

There are 512 neuron compartments emulating the leaky integrate and fire model and 131,072 STP/STDP synapses in one HICANN-X chip, but we will stick to a single neuron for now.

A silent neuron

As a first experiment we will record the membrane of a single, silent neuron on the analog substrate. We use pyNN as an interface, which can be used similarly to existing simulators and other neuromorphic platforms.

Simple 2 1 EBRAINS-experimental | Idle Mem: 273.59 / 2048.00 MB

Mode: Command Ln 1, Col 1 ts_00-single_neuron.ipynb

Use Cases: Advanced Lab Course

EBRAINS Collaboratory

Collabs Documentation Support Chat Community

Log-out

Open in Lab

File Edit View Run Kernel Git Tabs Settings Help

Mem: 593 / 2048 MB

fp_pynn_introduction.ipynb

Code

EBRAINS-experimental

```

pynn.Projection(src, pop, pynn.AllToAllConnector(),
                synapse_type=synapse, receptor_type="excitatory")

[4]: Projection("population1-population0")

The created network of populations and projections can now be emulated for a selected time.

[5]: # the duration is given in ms,
# this is in the hardware domain, not in the biological
# (the hardware is faster by a factor of approx. 1000
# compared to biology)
duration = 0.1
pynn.run(duration)

Thereafter, the recorded behavior of the neurons can be read out.

[6]: # read out the spikes of the neuron in 'pop'
spiketrain = pop.get_data("spikes").segments[0].spiketrains[0]
print(f"The neuron spiked {len(spiketrain)} times.")
print(f"The spike times were: {spiketrain}")

# plot its membrane potential
mem_v = pop.get_data("v").segments[0].irregularlysampledsignals[0]

import matplotlib.pyplot as plt
%matplotlib inline
plt.figure()
plt.plot(mem_v.times, mem_v)
plt.xlabel("time [ms]")
plt.ylabel("membrane potential [LSB]")
plt.show()

The neuron spiked 0 times.
The spike times were: [] ms

```

Simple 0 1 EBRAINS-experimental | Idle Mem: 494.23 / 2048.00 MB Mode: Command Ln 1, Col 30 fp_pynn_introduction.ipynb

EBRAINS Collaboratory

Collabs Documentation Support Chat Community

Log-out

Open in Lab

File Edit View Run Kernel Git Tabs Settings Help

Mem: 332 / 2048 MB

fp_calibration.ipynb

Markdown

Python 3 (ipykernel)

```

• Measure the sampled value for voltages between 0 and 1.2 V. We recommend steps of 0.05 V. For each data point, save a mean ADC value. Hint: Use something like result["value"] for calculating your mean ADC values. Complete the cell below to do so:

Hint: You can use the measure_voltage function.

[ ]: import numpy as np

# Measure characteristic
voltages = ... # volt
results = np.zeros_like(voltages) # mean ADC values in LSB
errors = np.zeros_like(voltages) # std deviation of ADC values in LSB

with hxcomm.ManagedConnection() as connection:
    for voltage_id, voltage in enumerate(voltages):
        samples = ...
        results[voltage_id] = ...
        errors[voltage_id] = ...
        ...

• Plot the characteristic, i.e. plot the acquired value over the external voltage.

[ ]: import matplotlib.pyplot as plt
plt.figure(figsize=(12, 8))

# TODO
...

plt.xlabel("Voltage [V]")
plt.ylabel("MADC value [LSB]")

• Perform a linear fit in a suitable range (exclude possibly saturated values on top and bottom). Hint: In order to convert sample values to voltage later, you can use the results as "x data" and the voltages as "y data" for your fit function.

[ ]: # fit linear function to data
from scipy.optimize import curve_fit

# TODO

• Plot your linear fit and save the plot.
• Save the fit parameters for later, so you can convert ADC values to Volt.

You can use the fit parameters you obtained to implement a madc value to voltage conversion routine like so:

[ ]: def madc_to_voltage_conversion(m, b):
    def convert(value):
        return m * value + b
    return convert

madc_to_v = madc_to_voltage_conversion(*popt)

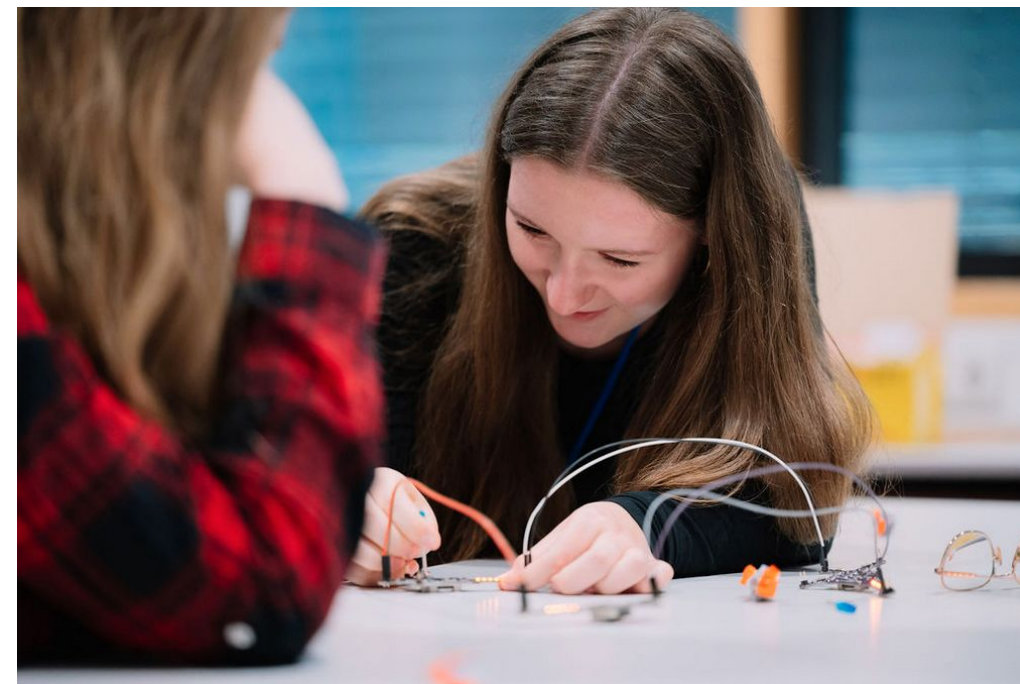
```

Characterizing Neuron Parameters

We now move on to a characterisation of Neuron Parameters, the goal is to relate the parameter values, which are specified in arbitrary units to the MADC measurement values. That way you will establish a correspondence between the values you specify and the membrane measurement values send back to you by the chip.

Simple 0 1 Python 3 (ipykernel) | Idle Mem: 255.57 / 2048.00 MB Mode: Command Ln 1, Col 1 fp_calibration.ipynb

Use Cases: Girlsday



Lu.i – An educational neuron PCB



Prerequisites

- EBRAINS account
 - Easy access to hardware
 - Tutorials available on github
- High number of participants → please contact us
 - Additional setups for reduced waiting times
 - Help with creation of guest accounts
- Prior knowledge
 - Depends on tutorial type



Demos



EBRAINS

Contact: Dr. Björn Kindler — bjoern.kindler@kip.uni-heidelberg.de