EBRAINS

BrainScaleS-2

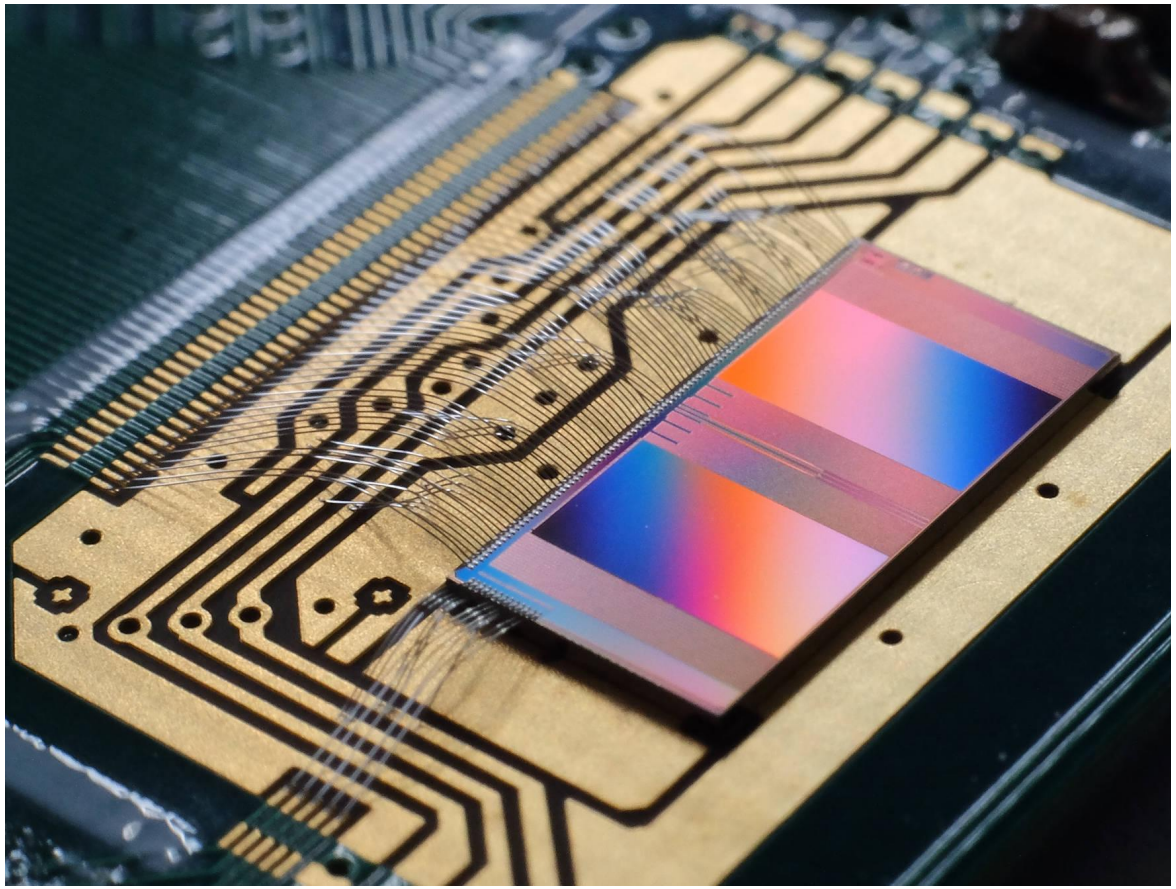
Jakob Kaiser (UHEI)

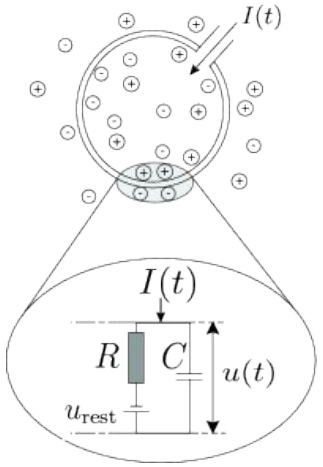
EBRAINS Tools for Teaching: Leveraging EBRAINS Open Science Tools for Neuroscience Education
EBRAINS Tutorials and Users Day, 12 March 2025 | Heidelberg, Germany

Co-funded by
the European Union

BrainScaleS-2 for Teaching

Physical emulation of neuron and synapse dynamics



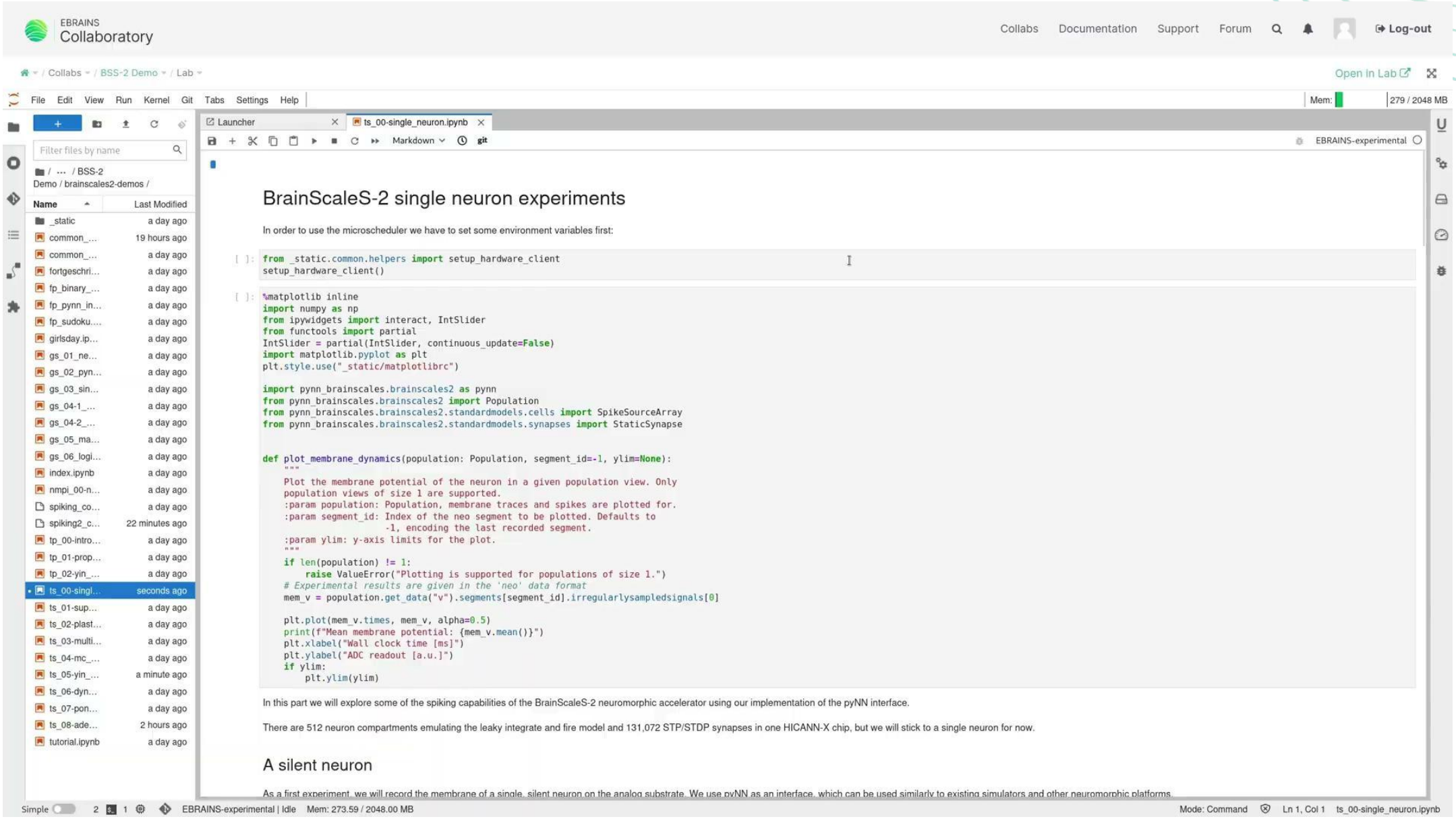


- Python interface (PyNN)
- Collection of educational jupyter notebooks
- Access via EBRAINS

Right figure taken from Gerstner et al. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. (Cambridge University Press, 2014)

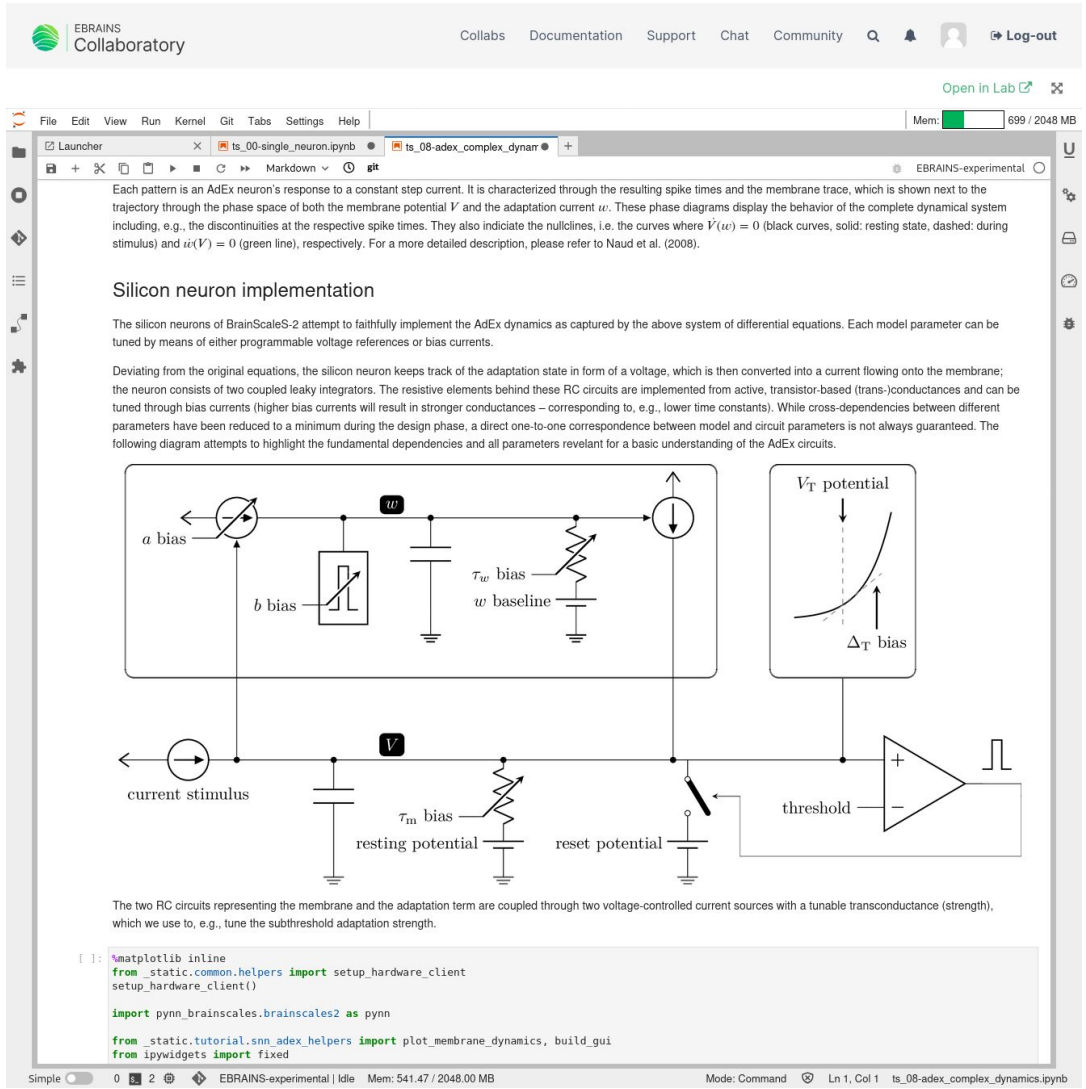
Co-funded by the European Union

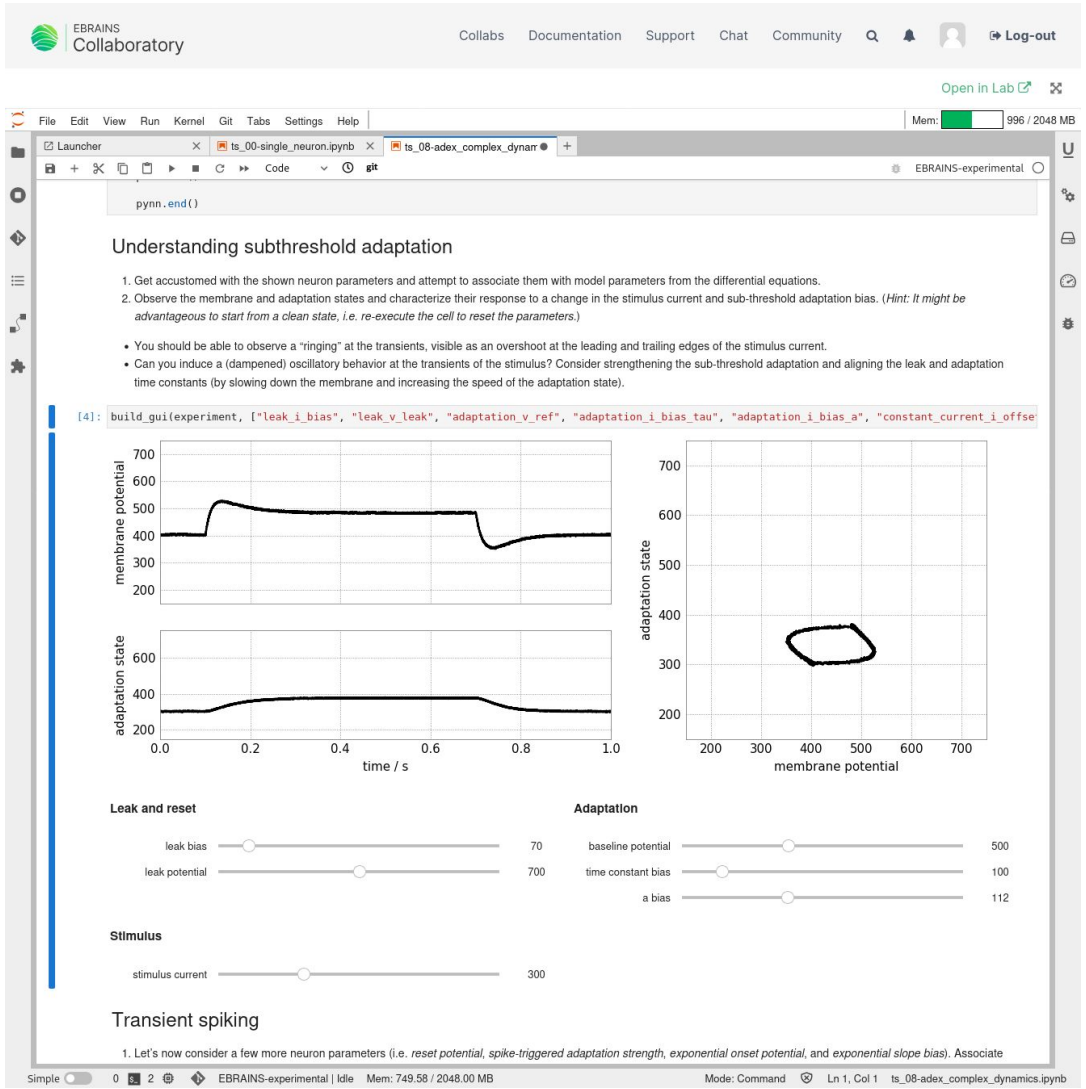
Use Cases: Workshops



Use Cases: University Lecture

Tutorial by Alessio Franci in “Principles of neuroengineering”





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 — git — 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)
```

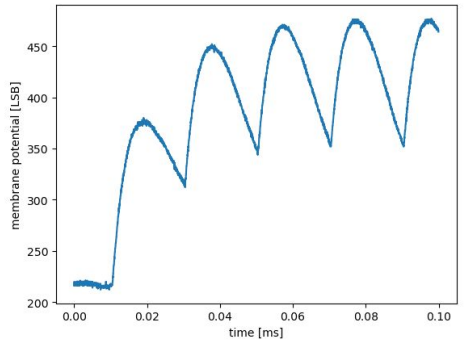
Thereafter, the recorded behavior of the neurons can be read out.

[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 — git — 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

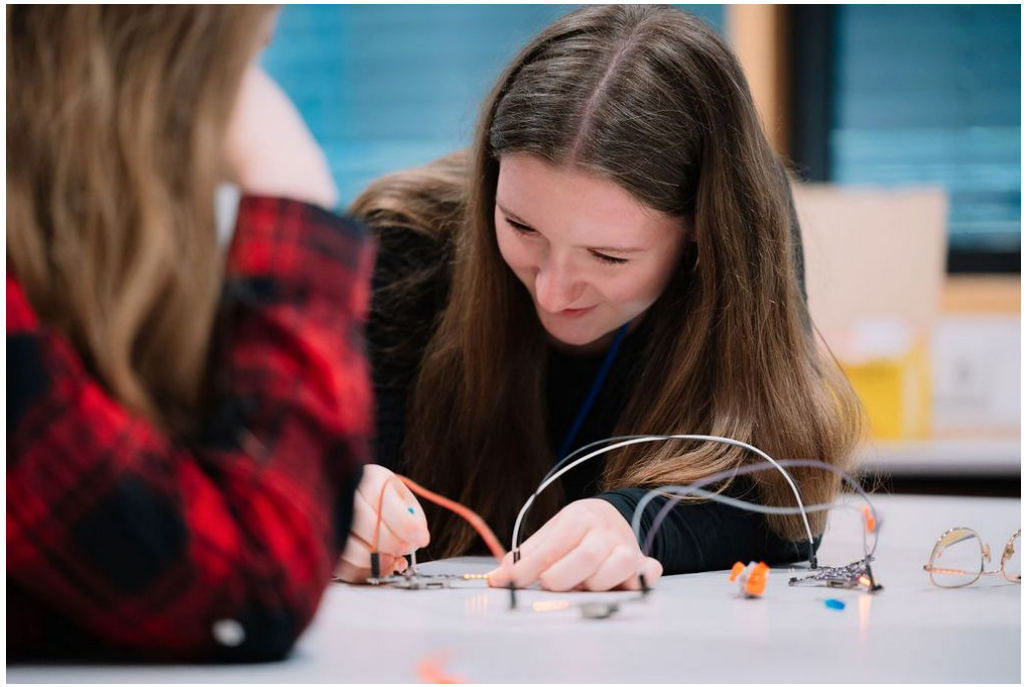
We now move on to a characterisation of Neuron Parameters, the goal is to relate the parameter values, which are specified in arbitrary units to the MADC measurement values. That way you will establish a correspondence between the values you specify and the membrane measurement values send back to you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

Co-funded by the European Union

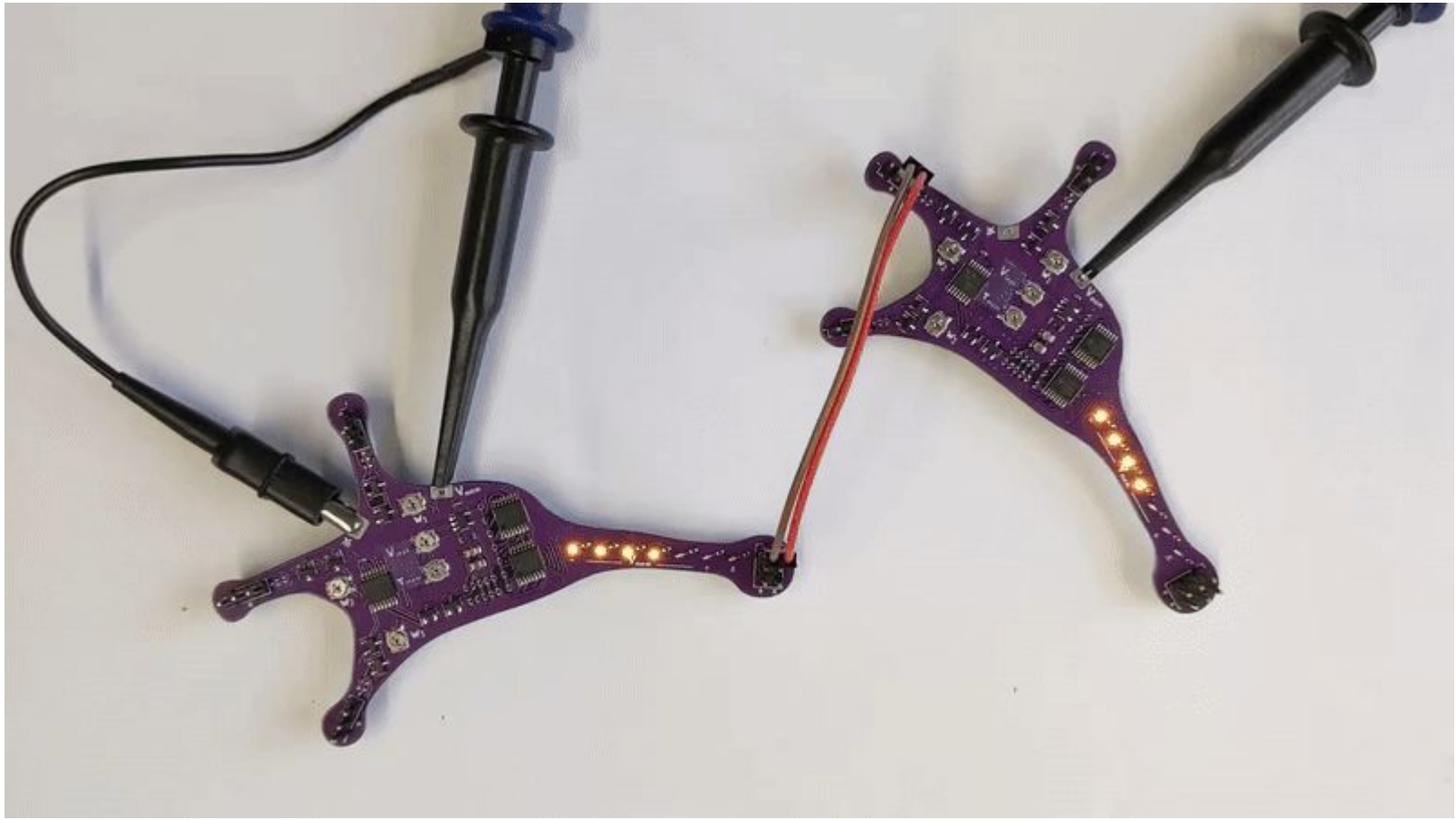
EBRAINS

Use Cases: Girlsday



Co-funded by the European Union

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

Co-funded by the European Union