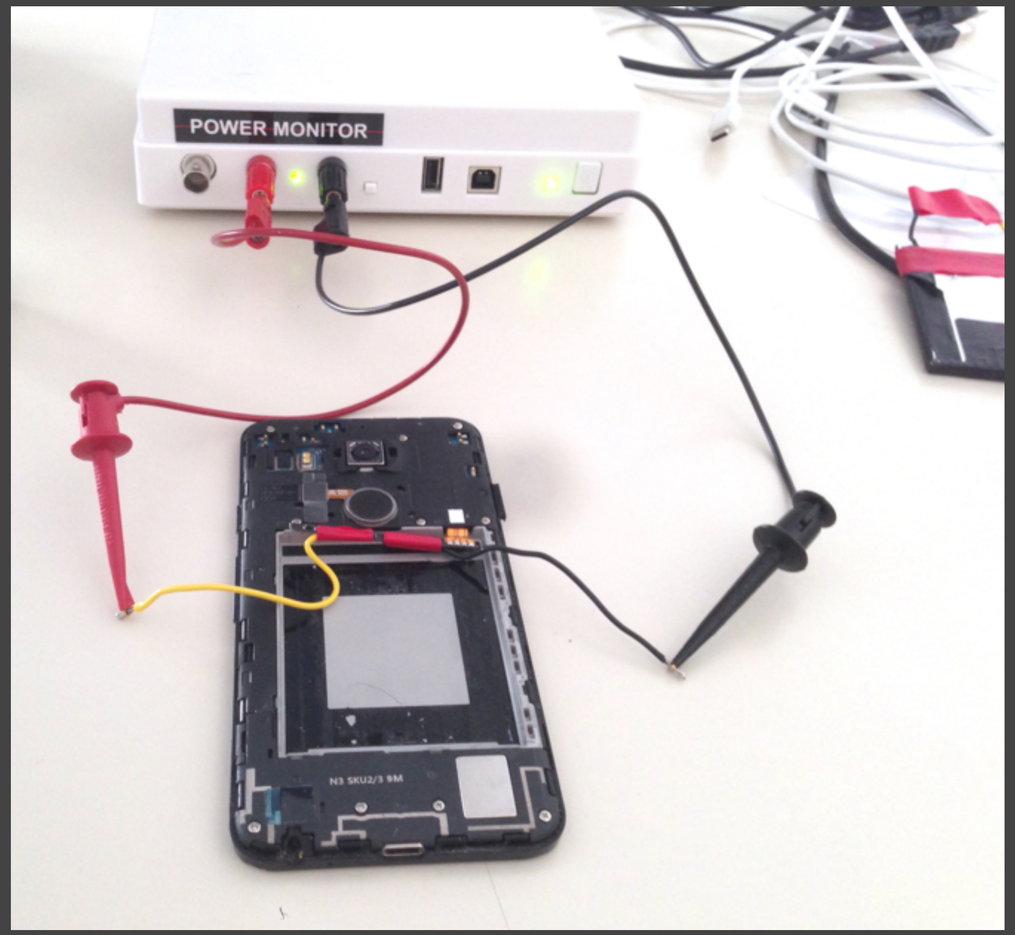
5. Tools to Measure Software Energy (lab)

Sustainable Software Engineering
CS4295

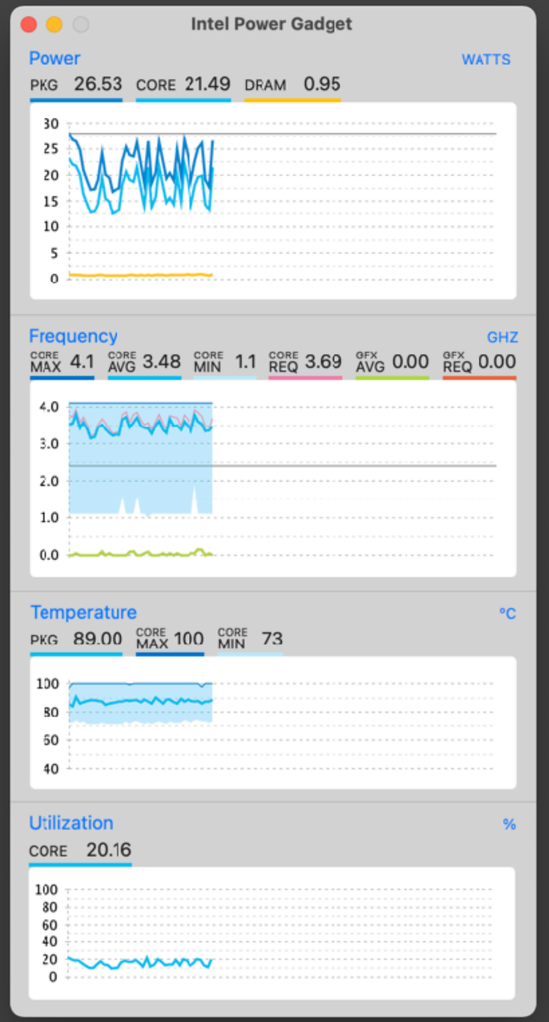
Luís Cruz
L.Cruz@tudelft.nl

1. Tools
2. Hands-on
3. Project 1

Hardware Power Monitors



Energy Profilers



Hardware Power Monitors

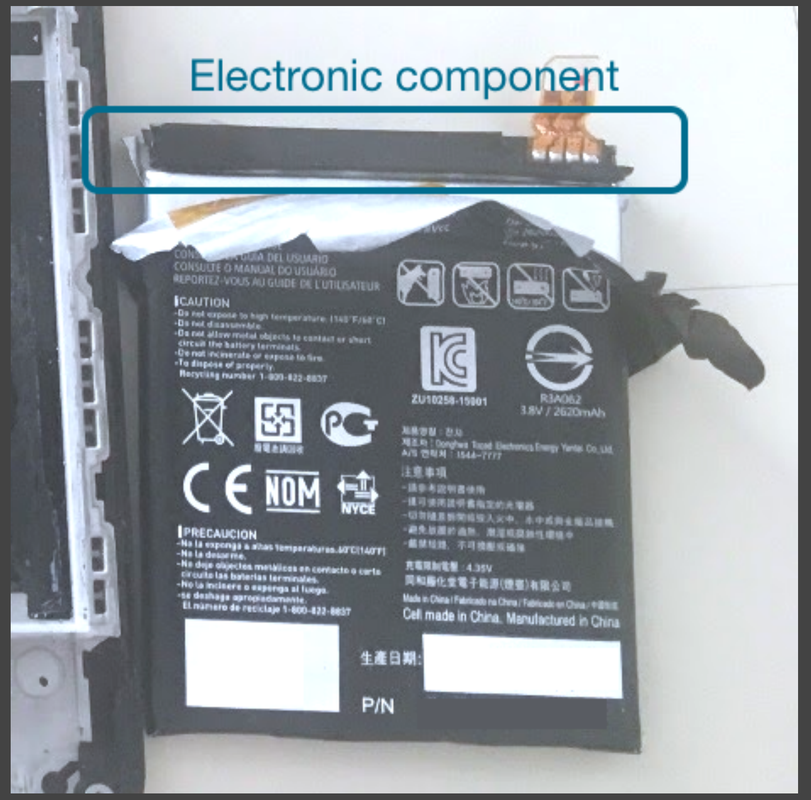
- Connects directly to the power source of the device/ component.
  - Some power monitors also replace the power source.
- Example:
  - **Monsoon Power Monitor** (for IoT and smartphones).
    - Can be fully automated using a Python API.
    - It measures and powers small electronic devices.
- There are many power/energy meters out there but for **software use cases** we need to be able to **control them using an API**.

Connecting Monsoon to a Smartphone

- 1. Disassemble the smartphone and **find the connectors** of the battery.
- **iFixit** usually has nice tutorials and blueprints. https://www.ifixit.com

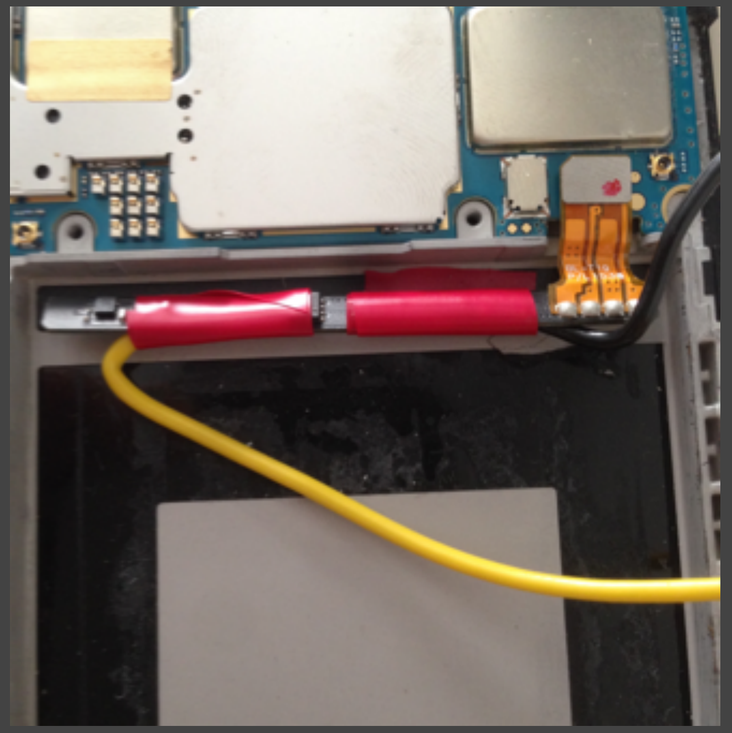
Connecting Monsoon to a Smartphone

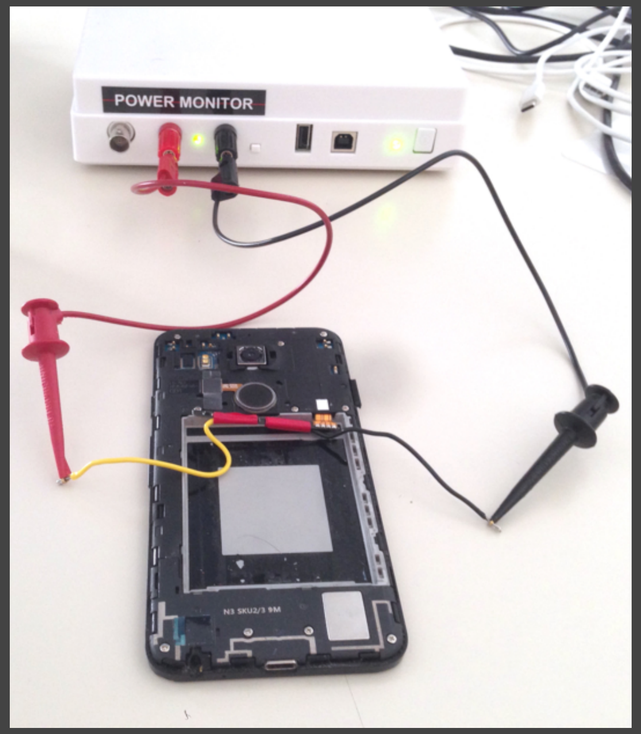
- 2. Extract the electronic component of the battery
- Modern batteries are connected through **4 terminals**:
  - **Positive**
  - **Negative**
  - **BTEMP**, battery temperature (used for safety)
  - **BST**, battery system indicator (provides info about the battery)
  - Hence, one cannot simply connect + and - pins



Connecting Monsoon to a Smartphone

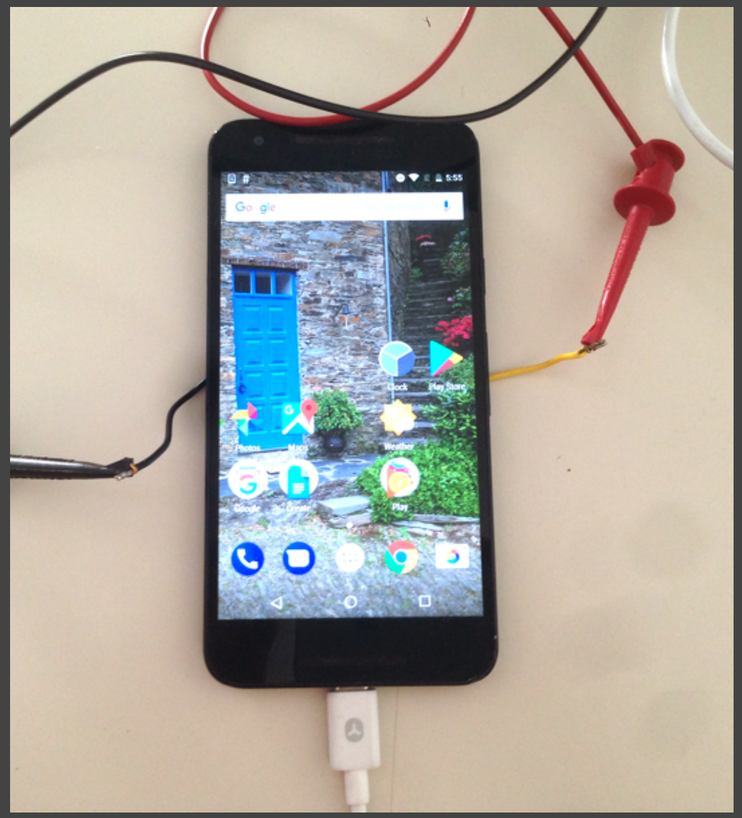
- 3. Connect the electronic component directly to the monitor.





Connecting Monsoon to a Smartphone

- 4. Use the library **PyMonsoon** to control the power monitor.
  - https://github.com/msoon/PyMonsoon
  - 4.1. Set the monsoon to desired Voltage. Choose the **typical voltage** of the **original battery**. For the Nexus 5X, **3.8V** was equivalent to its battery at around 60% capacity.
  - 4.2. Start measuring



Connecting Monsoon to a Smartphone

- 5. Automate User Interface interaction
  - The last thing you want to do is to manually interact with the smartphone while you measure energy consumption. Tests are **less accurate**, **less reproducible**, and, in this case, **the screen cannot not be easily accessed**.
  - Tools to automate interaction with Android phones:
    - To open, install, close apps: **adb**
    - To interact with the app: **Appium**, **Robotium**, **UIAutomator**, **espresso**, etc.
    - Alternative: **physalia** automates all adb interactions and PyMonsoon calls.

Issue: USB cable!

- **You need the USB cable** to automate the interaction with the phone.
- When you connect the USB cable, measurements are be completely **unreliable**.
- **Solution:**
  - Monsoon has a feature to control the USB connection (switch on/off)
    - **Option 1**: Right before starting measurements, the USB connection is stopped.
      - Works fine when when all the interaction instructions can be sent in advance and the time for the execution is already known.
    - **Option 2**: using USB, set up a **wireless ADB connection**. Stop USB connections afterwards.
      - How to: https://stackoverflow.com/a/3623727

Issue 2: your app is not exclusive

- Many activities run in a smartphone device. E.g., getting push notifications, checking nearby bluetooth devices, etc.
  - Moreover, brightness may change according to environment. Different screen brightness, different results.
- You need to **reduce tasks to the bare minimum**:
  - Set brightness to a fixed value; turn off notifications, kill all user-owned processes, turn off cellular data, bluetooth, location services, account syncs; uninstall all unnecessary apps, etc.

When it comes to desktop/cloud software, the sources of **noise** are different but **the same concerns apply**.

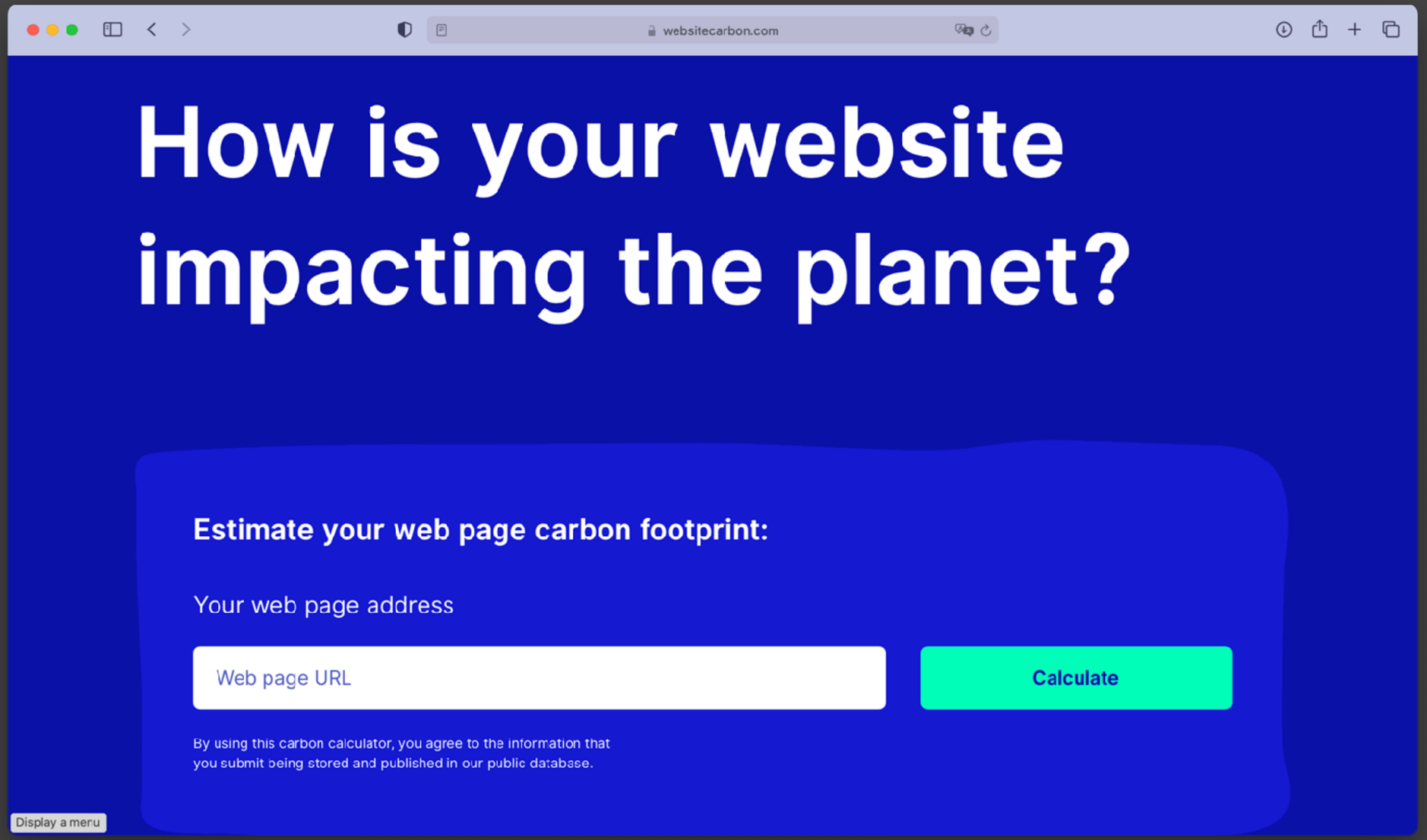
Each case is different – think it through!

Energy Profilers

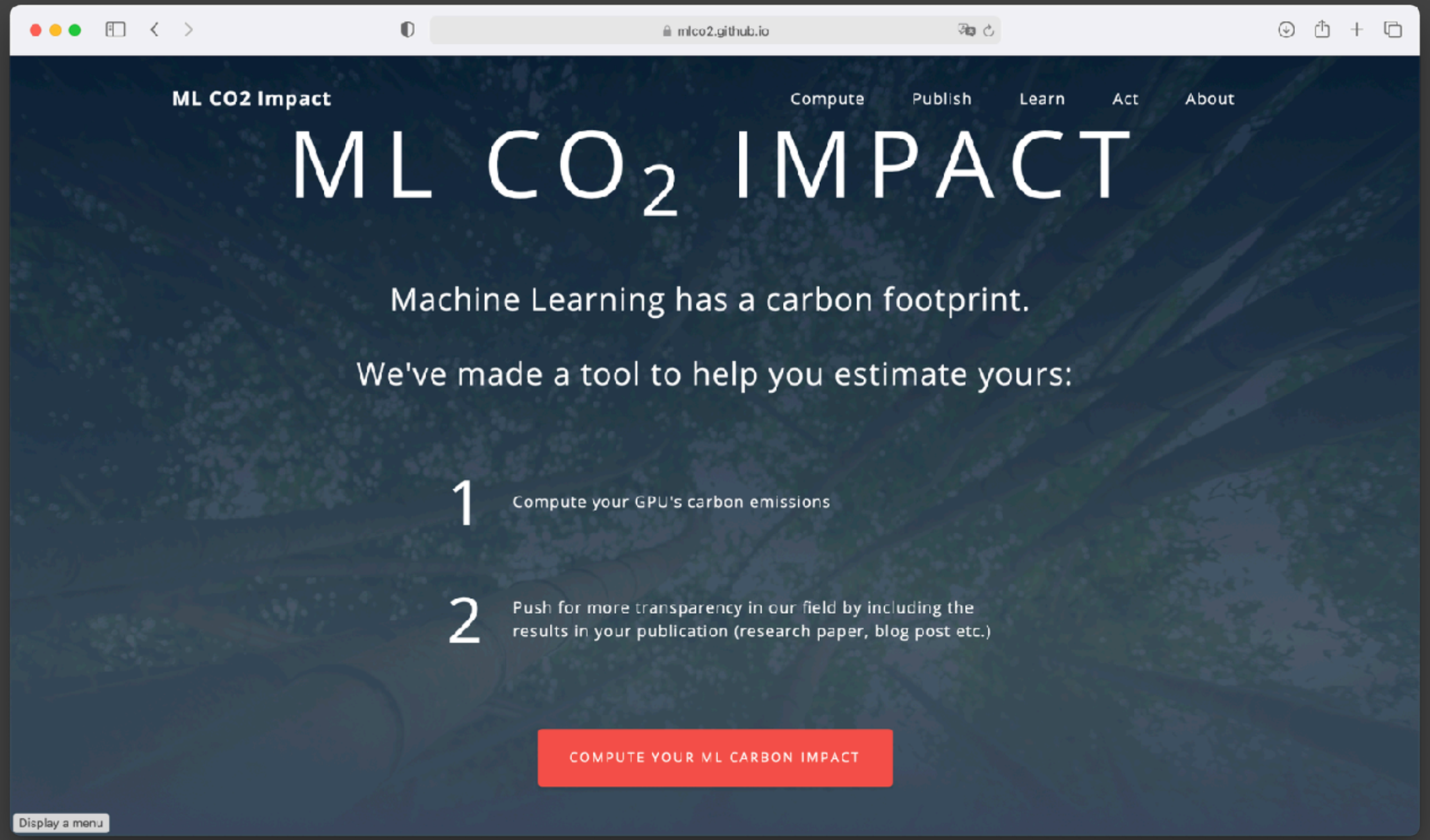
- **Simple setup**! Quite reliable (if you choose the profiler wisely).
  - Recently, they are starting to rely on internal power sensors.
- Still **sensitive to noise** from concurrent processes/tasks! ⚠️

Examples of Energy Profilers

https://www.websitecarbon.com



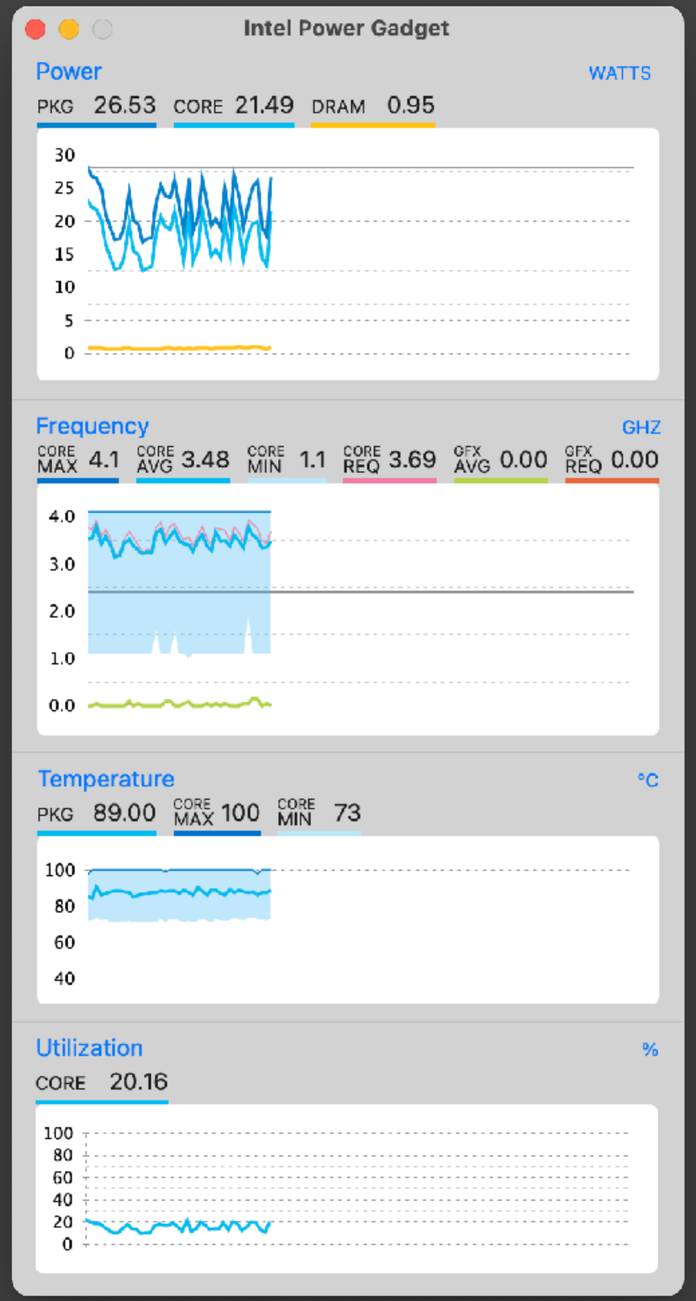
https://mlco2.github.io/impact/



Intel Power Monitor

- **Install**: https://software.intel.com/content/www/us/en/develop/articles/intel-power-gadget.html
- **To collect**: `Logging > Log to File`

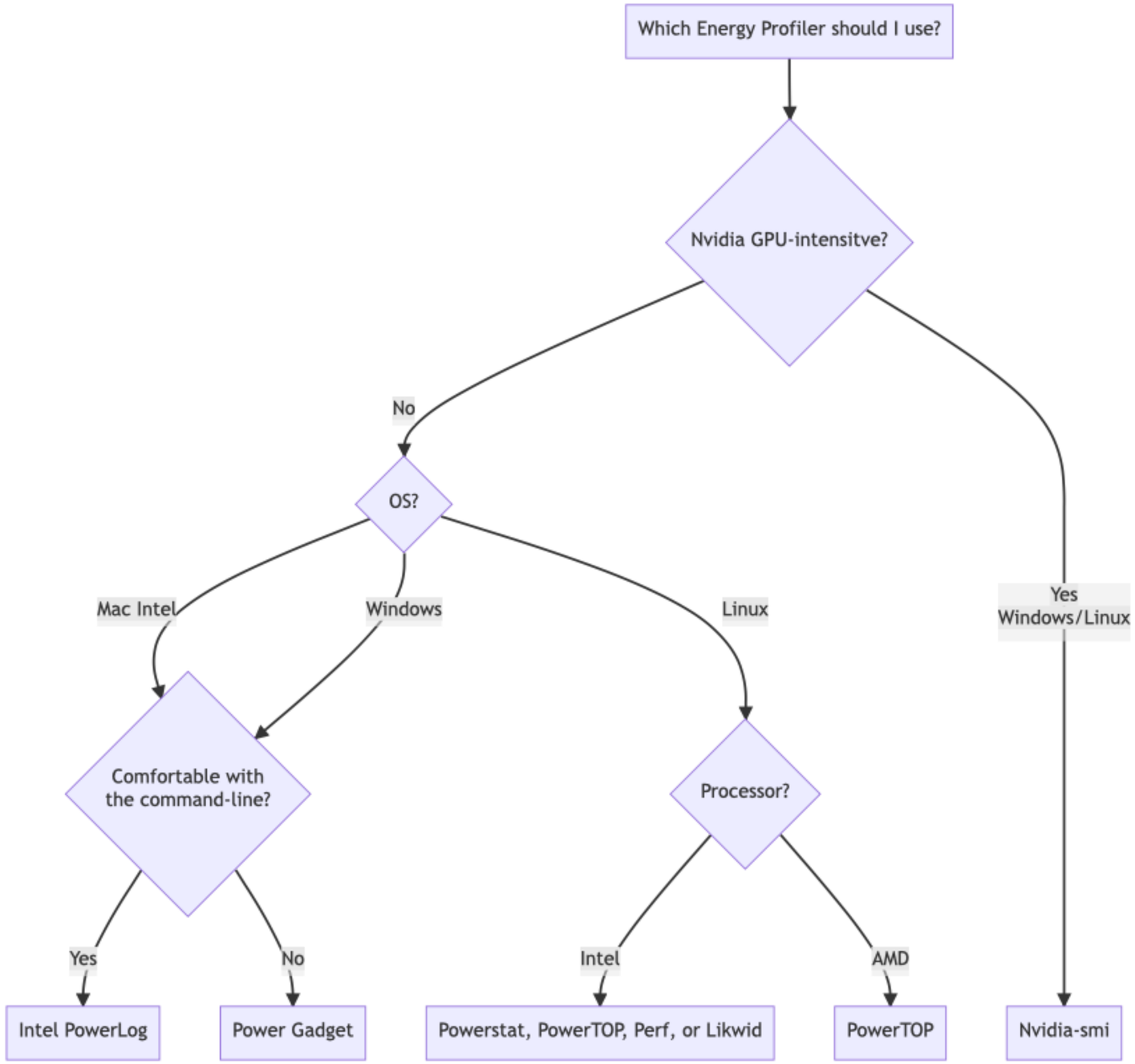
- It will store a **CSV file** with all the collected power data. (File location is specified in the settings)
- Based on Intel **RAPL**. Works with Intel-based Windows and Macs.
- Alternative-twin for M1-based Macs: **Mx Power Gadget**. https://www.seense.com/menubarstats/mxpg/



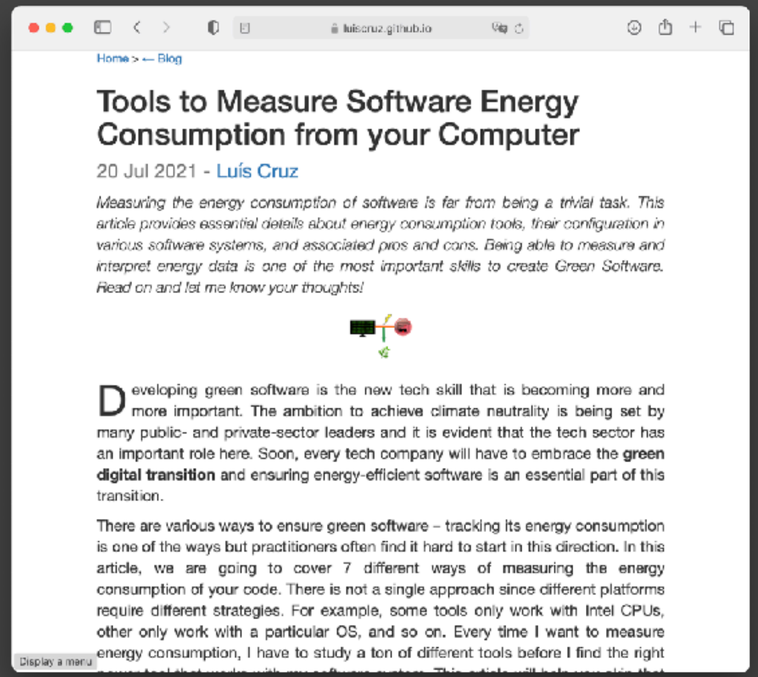
Intel Power Monitor - Log file

- **Total Elapsed Time (sec)**. The total time in seconds in which power data was being collected.
- **Cumulative Package Energy_0 (Joules)**. The total energy consumption of the processor.
- **Cumulative DRAM Energy_0 (Joules)**. The total energy consumption of the volatile memory.
- (Note: IA is only the CPU cores, package is the whole CPU Package; some reports include “GT Energy” – i.e., energy consumption from the GPU).

104	Total Elapsed Time (sec) = 10.030457
105	Measured RDTSC Frequency (GHz) = 2.400
106	
107	Cumulative Package Energy_0 (Joules) = 140.248840
108	Cumulative Package Energy_0 (mWh) = 38.958011
109	Average Package Power_0 (Watt) = 13.982298
110	
111	Cumulative IA Energy_0 (Joules) = 89.629333
112	Cumulative IA Energy_0 (mWh) = 24.897037
113	Average Package IA_0 (Watt) = 8.935718
114	
115	Cumulative DRAM Energy_0 (Joules) = 9.915833
116	Cumulative DRAM Energy_0 (mWh) = 2.754398
117	Average Package DRAM_0 (Watt) = 0.988572



🔗 https://luiscruz.github.io/2021/07/20/measuring-energy.html



Home > ← Blog

Tools to Measure Software Energy Consumption from your Computer

20 Jul 2021 - Luís Cruz

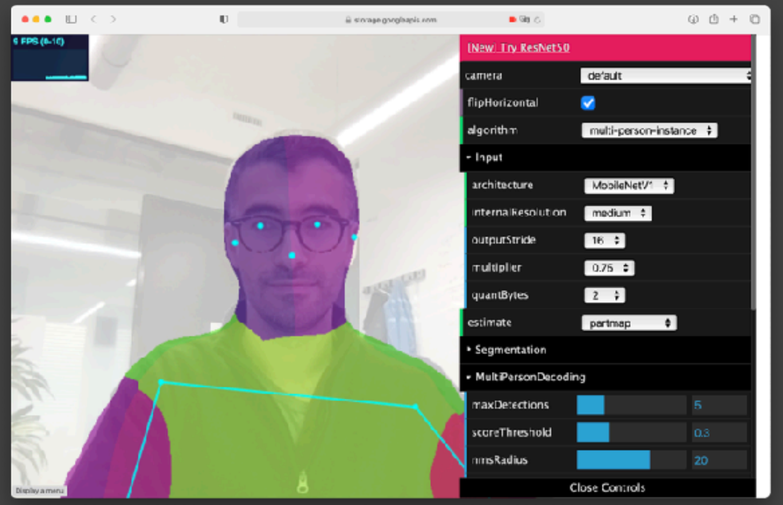
Measuring the energy consumption of software is far from being a trivial task. This article provides essential details about energy consumption tools, their configuration in various software systems, and associated pros and cons. Being able to measure and interpret energy data is one of the most important skills to create Green Software. Read on and let me know your thoughts!

Developing green software is the new tech skill that is becoming more and more important. The ambition to achieve climate neutrality is being set by many public- and private-sector leaders and it is evident that the tech sector has an important role here. Soon, every tech company will have to embrace the **green digital transition** and ensuring energy-efficient software is an essential part of this transition.

There are various ways to ensure green software – tracking its energy consumption is one of the ways but practitioners often find it hard to start in this direction. In this article, we are going to cover 7 different ways of measuring the energy consumption of your code. There is not a single approach since different platforms require different strategies. For example, some tools only work with Intel CPUs, other only work with a particular OS, and so on. Every time I want to measure energy consumption, I have to study a ton of different tools before I find the right

(Missing Apple m1 tools: mxpg, powermetrics)

Hands-on 1



- **Install** your energy profiler (e.g., Power Gadget).
- **Collect** the energy data of using **Coral BodyPix** for **30 seconds**.
  https://storage.googleapis.com/tfjs-models/demos/body-pix/index.html
- **Report** the **total energy consumption**.
- **Extra-mile**:
  - Compare the energy consumption in different browsers.
  - Check the spikes and drops in **Power** and **Temperature**.

Retrospection
Hands-on 1

- Are the measurements repeatable?
- What were the confounding factors?
- How can we automate this process?

Energy testing
(Different from energy monitoring)

1. Create a **reproducible scenario** of the execution of your software. Preferably this should be an automated script – e.g., using a unit test framework.

2. **Execute the scenario** in a version of your software. Use the energy profiler to measure the energy consumption.

3. Improve your software in parts of the code that you suspect have low performance.

4. Execute the same scenario with the **new version**. **Compare the energy data** in this version with the previous one.
   Energy is lower, **test passes**; energy is higher **test fails**.

Hands-on 2

- Create a reproducible scenario. (Usually easier with command-line interfaces)
- Automatically start/stop energy profiling.

Project 1

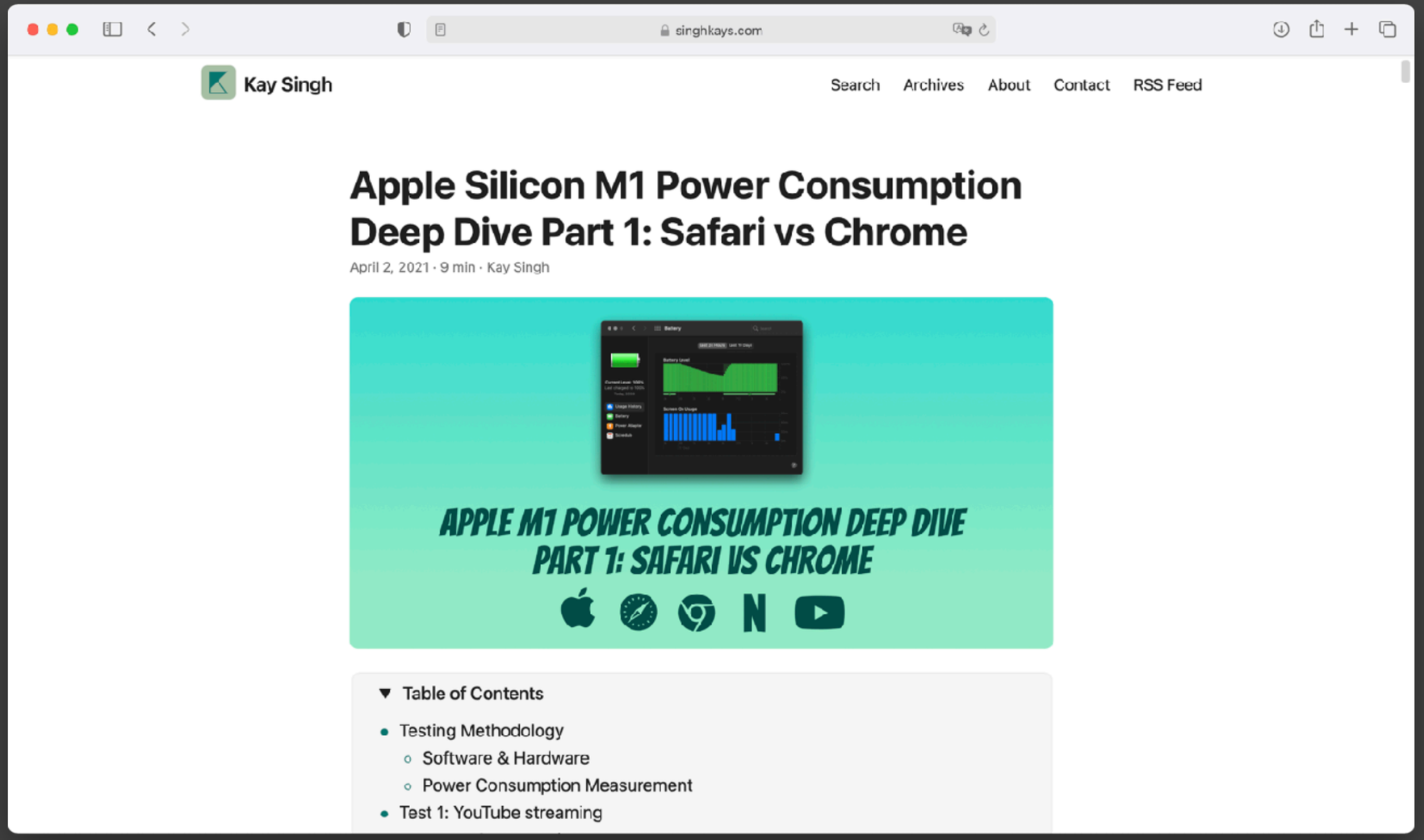
- Deadline: 25 Feb (new; before: 24 Feb)
- **Compare energy consumption** in common software use cases.
  - Examples:
    - **Different versions** of the same app;
    - **Same use case** but different apps
    - Same version, same app, but different **user settings** (e.g., enable/disable GPU optimisation)
    - Same version, same app, but different **running environment**
- Submission via **PR** (markdown).
  - Blog-style report (**markdown**, approx **2500 words**).
  - Bonus if you can **automate** the experiment and/or if there is a **replication package**.

Pierre Lagarde. VLC Energy Optimization with GPU

https://devblogs.microsoft.com/sustainable-software/vlc-energy-optimization-with-gpu/

Kay Singh. **Apple Silicon M1 Power Consumption Deep Dive Part 1: Safari vs Chrome**

https://singhkays.com/blog/apple-silicon-m1-video-power-consumption-pt-1/



(For project 1, you don’t need to dive deep into hardware details)