Co-funded by the
Erasmus+ Programme
of the European Union

Department of
Computers and Informatics
FEEI TU of Košice

How To Measure Energy Impact — Tools & Techniques

Csaba Szabó
Technical university of Košice, Slovakia

The role of software

- Operating system (difference between Windows, Linux, macOS, Android, iOS)
- Working software
- Computer games
- Application systems
- Databases

The role of the user

- The user “drives” the software
- Needs individual training (unlike HW/SW)
- Does (s)he receive it? Where?
- Repairing bad configuration is often done by buying a new device…

Motivation example

- The guys are gone fishing without the girls
- They have to keep contact with them for various reasons via their mobile phones
- They use their mobile phones for outdoor navigation
- They also use their mobile phones to connect to the fish finder (sonar) several times per day
- How many power banks they need for one week to survive?

Measuring energy consumption

Incl. improvements

- System level
- Application level
- Component level
- Code level

- Process level

More details can be found in Intellectual output 1 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

Tools

- https://software.intel.com/en-us/articles/intel-power-gadget-20
- Google: Joulemeter1.2Setup
- https://developer.apple.com/library/content/documentation/Performance/Conceptual/EnergyGuide-iOS/MonitorEnergyWithXcode.html
- PowerTop for linux
- Trepn Profiler for Android
- etc.

Create your own energy measurement framework for Java!

- C/C++/C# (Intel Power Gadget - exists)
- jRAPL (uses RAPL)

- For an own: pack RAPL/IPG with JNI + extend with JVM features (memory) + use IOTOP or similar for disk operations (you will need to know hardware specs like memory/disk energy consumption — S.M.A.R.T.)

Techniques

Basic terms

- Measurement -> Calibration
- Software life -> Development
- Software life -> Evolution
- Software testing -> Black-box testing
- Software testing -> White-box testing

Measurement

- To measure is to observe one or more system attributes
- Measurement uses metrics and metrics values to state the results of such observation
- Metrics are used to compare two or more artefacts
- Energy consumption can be measured.

Development and evolution

- Energy consumption is not a very often requirement, at least not an accelerated one
- If exists, it is a non-functional requirement, which makes it measurable just after building the product
- There exist energy saving guidelines for different OS or target devices
- Energy efficiency is more often a requirement arising during usage

Black box testing

- A test technique applied in testware of (almost) final products.
- Start -> observe -> close -> evaluate. During observation, the complete software is running.
- Objective method since it is not altering the SUT.
- Can find existence of a failure but cannot locate it.

White box testing

- A test technique applied during module development and integration.
- Select -> configure -> run -> evaluate. Selection takes a part of the system code that will be tested.
- It creates a modified SUT to locate the source of the failure or error.

Energy consumption measurement

- Invasive or not (black or white?)
- Invasive: more precise for the price of change
- Non-invasive: how can we be sure we measure only what we want?

- Hardware measurement – using sensors or external devices
- Software measurement – using estimation based on host/target system configuration

Invasive EE measurement

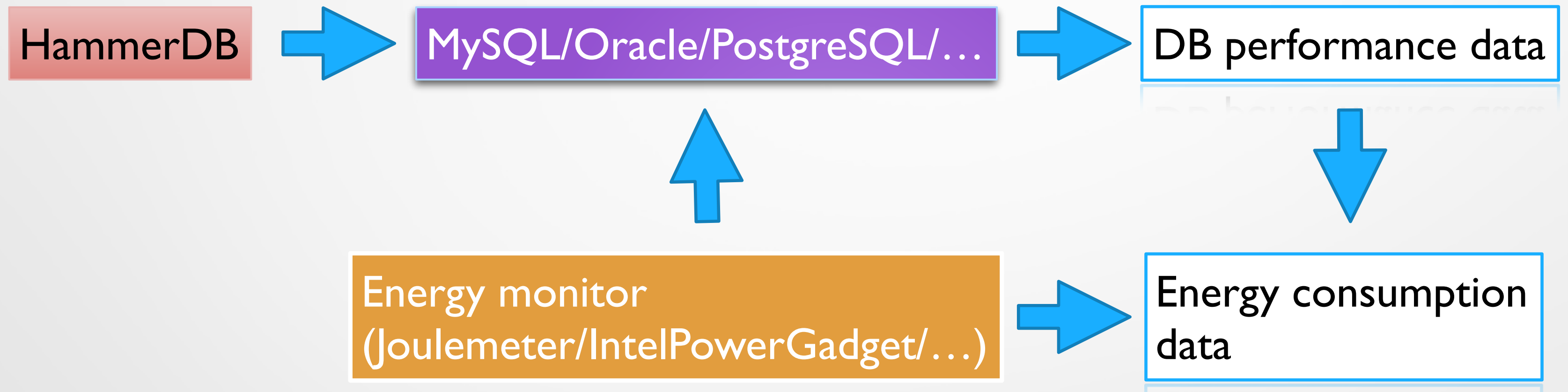
- Principle 1: running white-box tests that position the measured routine into a frame with known energy consumption. After running a required minimum number of tests, one can statistically estimate the energy efficiency of the routine.
- Principle 2: adding signal sources into the code to start/stop measurement

Non-invasive measurement

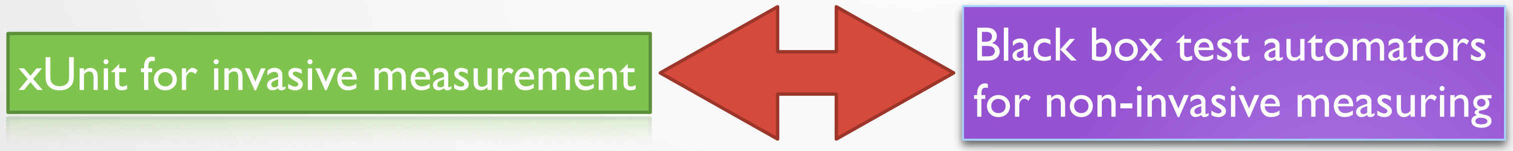
- Principle 1: We measure energy consumption of the system with and without running the measured application. After a suitable statistical evaluation, the energy efficiency of the application can be calculated.
- Principle 2: We calibrate the energy consumption estimation model on the host system using known processes. Then, we observe the behavior of the selected process on the host. Observed values are turned to energy consumption estimations based on CPU load, memory usage etc.

- One could set up a measurement process that combines the previous principles.

Measuring the DB


HammerDB
MySQL/Oracle/PostgreSQL/…
DB performance data
Energy monitor (Joulemeter/IntelPowerGadget/…)
Energy consumption data

Applying testing frameworks


xUnit for invasive measurement
Black box test automators for non-invasive measuring

Example technique

1. Setup the environment
2. Start the energy monitor
3. Develop (think, code, test, fix) for 15 minutes
4. Have a 5 minutes break (stop energy usage monitoring, set up the next one, get a coffee)
5. Finish (for this time) if there is no further idea
6. Repeat (jump to label 2)
7. Analyse collected data (energy efficiency of your development process) inside the team

More details can be found in Intellectual output 1 and 2 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

Example development game

- Let develop a JavaFX FXML application that implements a classical mirror. It will use the computer's built-in camera to capture the video that will be then displayed in the application window.
- We will use IntelliJ IDEA for JavaFX, Oracle's Scene Builder for visual FXML GUI design, and the OpenCV library for image processing.

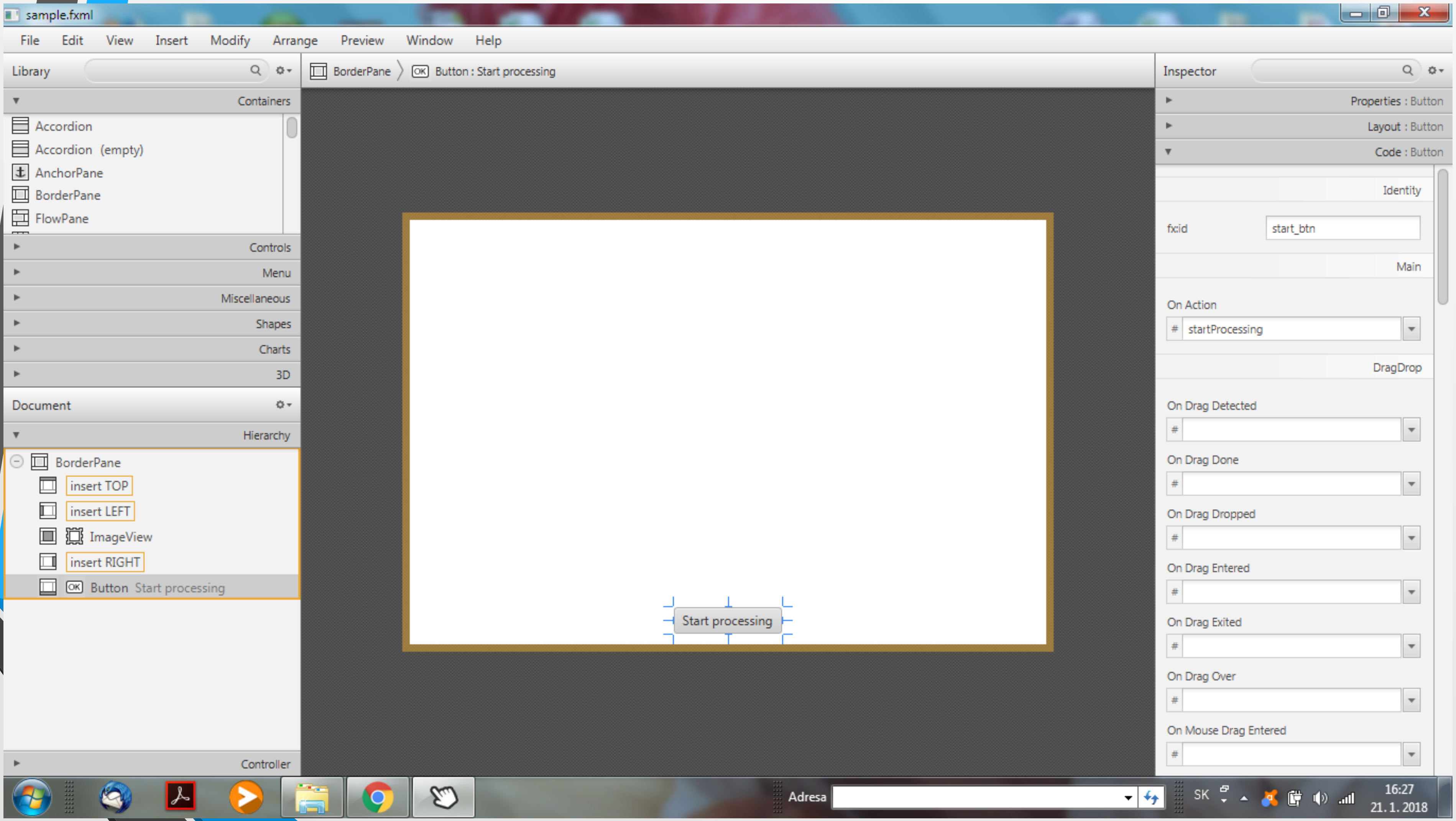
Our search engine found the following reusable sources

- https://opencv-srf.blogspot.sk/p/opencv-lessons.html
- https://opencv-srf.blogspot.sk/2010/09/object-detection-using-color-seperation.html
- https://www.codeproject.com/Tips/717283/How-to-Use-OpenCV-with-Java-under-NetBeans-IDE
- http://opencv-java-tutorials.readthedocs.io/en/latest/03-first-javafx-application-with-opencv.html
- https://github.com/opencv-java/getting-started/tree/master/FXHelloCV/src/it/polito/elite/teaching/cv

The development process

- We found a reusable solution, it will save time and energy to use it.
- We follow the tutorial.
- If required, we introduce changes to fit the tutorial to our requirements.
- This is a small task, to which a tutorial exists, but because of the non-native library used, it can be qualified as Medium.
- Total time to solve the problem is also short (even when not considering the option downloading the project from Git)

Scene Builder



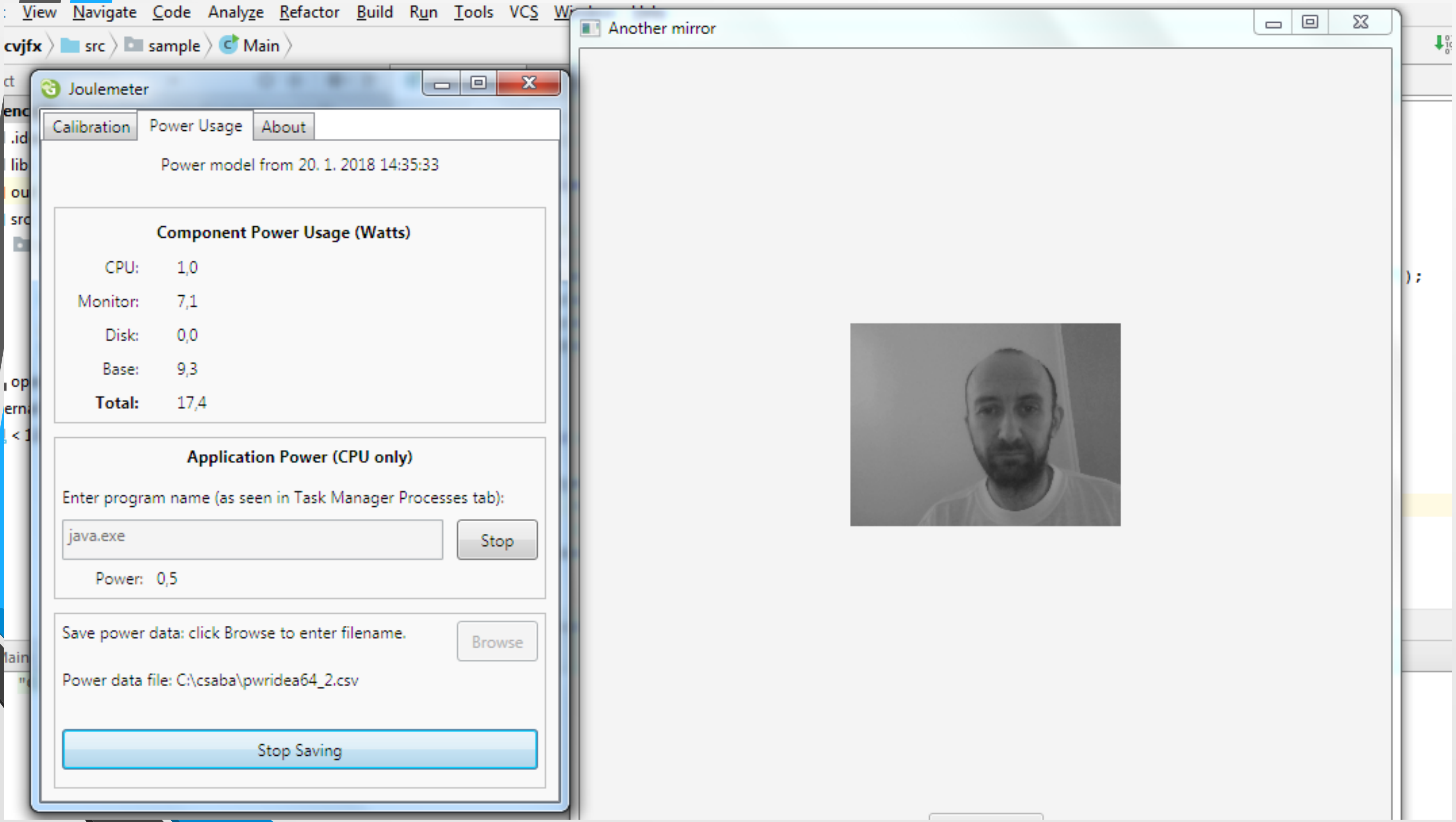
fx:id: start_btn

On Action: # startProcessing

Finals

- During development it is normal to compile and run an application many times, which will have an effect on energy consumption. The goal of our measurements is to point out this energy and combine to the energy consumption of the product itself.

Result



Power model from 20. 1. 2018 14:35:33

Component Power Usage (Watts)

CPU:	1,0
Monitor:	7,1
Disk:	0,0
Base:	9,3
Total:	17,4

Application Power (CPU only)

Enter program name (as seen in Task Manager Processes tab):

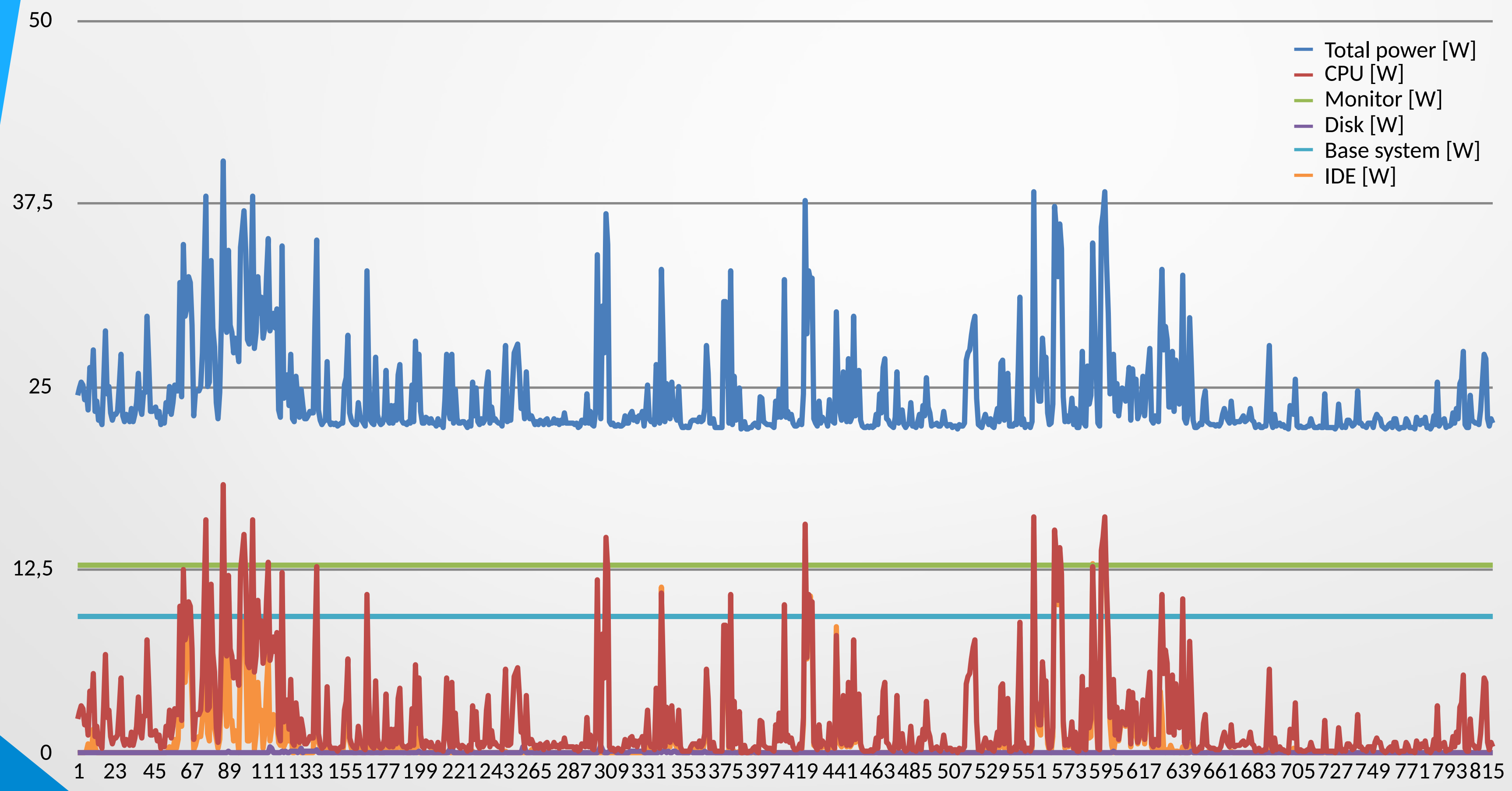
java.exe

Power: 0,5

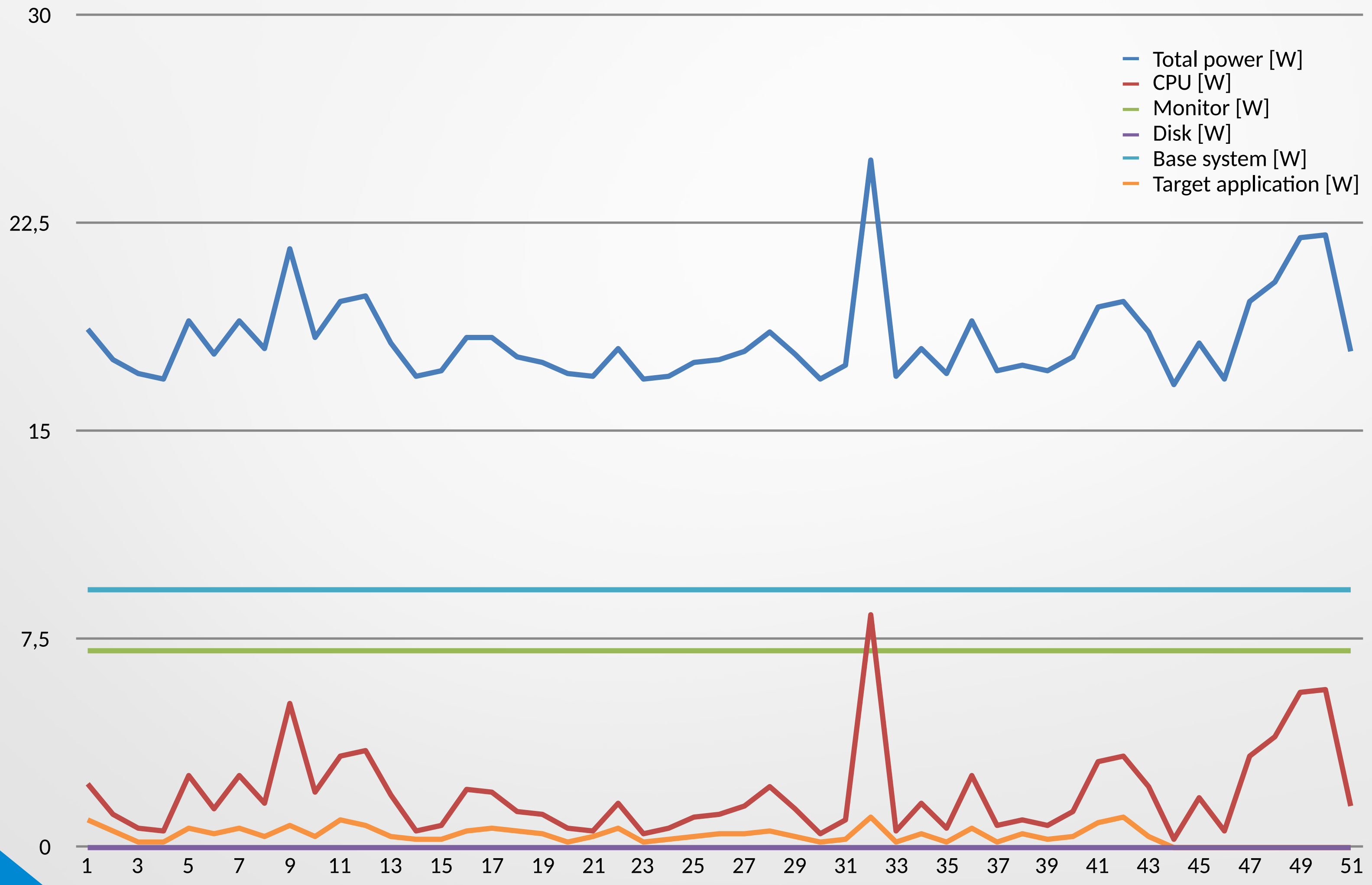
Save power data: click Browse to enter filename.

Power data file: C:\csaba\pwridea64_2.csv

Measurement results – development



Measurement results – product



Conclusion

- Target application energy consumption: 2.19 Ws
- IDE energy consumption: 74.72 Ws
- Total host system energy consumption: 1983.28 Ws

kpi

TECHNICAL UNIVERSITY OF KOŠICE
Faculty of Electrical Engineering
and Informatics

Liked it? ☺

Csaba.Szabo@tuke.sk

Disclaimer

This presentation contains parts of Intellectual output 1, 2 and 4 of the ERASMUS+ project No. 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software as reference to older Intellectual outputs of the specific project.

Co-funded by the Erasmus+ Programme of the European Union