Applying The Blue Angel Criteria To Free Software

A handbook to certify software as sustainable

A KDE Eco initiative

eco.kde.org

Contents

About The Authors

KDE Eco tooling and documentation are provided by community members who have volunteered to contribute to this project for the benefit of all. Primary contributors include (listed in alphabetical order by first name): Arne Tarara, Cornelius Schumacher, Emmanuel Charruau, Karanjot Singh, Nicolas Fella, and Volker Krause. Thank you—your contributions make this handbook possible.

The text of this version of the handbook was written and/or compiled from the above documentation by Joseph P. De Veaugh-Geiss. Olea Morris edited the text. Lana Lutz and Arwin Neil Baichoo made the book and website design as well as the images therein beautiful. Paul Brown made significant improvements to the Okular blog post adapted for Part II: First Eco-Certified Computer Program: KDE's Popular Document Reader Okular. Wikipedia was a source of several texts which were included here in modified form: thank you to the community of Wikipedia writers and editors for making such a wonderful resource for all of us. See the end of each section for additional information about sources.

Acknowledgments

Thank you to the many contributors to the KDE Eco initiative in general (listed in alphabetical order by first name): Achim Guldner, Adriaan de Groot, Aleix Pol, Alexander Semke, André Pönitz, Björn Balazs, Carl Schwan, Chris Adams, Christopher Stumpf, David Hurka, Fabian, Felix Behrens, Franziska Mai, Harald Sitter, Jens Gröger, Johnny Jazeix, Jonathan Esk-Riddell, Kira Obergöker, Lydia Pintscher, Marina Köhn, Mathias Bornschein, Max Schulze, Phu Nguyen, Sami Shalayel, Stefan Naumann, Sven Köhler, and Tobias Fella. Your contributions are greatly appreciated.

People who are interested in contributing to KDE Eco are encouraged to join the mailing list or Matrix room. Contributors are also invited to join one of the KDE Eco sprints and in-person or online meetups. Learn more at out website.

This text has also benefitted from the informative conversations that took place at the following conferences and workshops: Akademy 2022, Linux App Summit 2022, FOSDEM 2023, rC3: NOWHERE 2021, SFSCon 2021/2022, Grazer Linuxtage 2022, Qt World Summit 2022, QtDevCon 2022, Fedora Nest 2022, Green Coding Berlin meetups, Sustainable Digital Infrastructure Alliance hackathon, EnviroInfo 2022, and Bits & Bäume 2022. Thank you!

License

Unless indicated otherwise, the contents of this handbook are released under the Creative Commons Attribution-ShareAlike 4.0 International (CC-BY-SA-4.0) license. For more information about documen-

tation licensing at KDE, see KDE's licensing policy.

Introduction

This handbook provides a brief overview of environmental harm driven by software and how the Blue Angel ecolabel, the official environmental label of the German government, provides a benchmark for sustainable software design.

The Blue Angel ecolabel is awarded to a range of products and services, from household cleaning products to small appliances to municipal services. In 2020, the German Environment Agency extended the award criteria to include software products. It was the first environmental certification in the world to to link transparency and user autonomy—two pillars of Free & Open Source Software (FOSS)—with sustainability.

At this point, you might wonder what sustainability has to do with software at all. How can something so seemingly immaterial as software have an environmental footprint? In this handbook, we'll take a closer look at the some of the ways software is contributing to the climate crisis, and how compliance with the Blue Angel award criteria for software eco-certification can help.

The book is divided into three parts:

- PART I: ENVIRONMENTAL IMPACT OF SOFTWARE
- PART II: ECO-CERTIFYING DESKTOP SOFTWARE
- PART III: FULFILLING THE BLUE ANGEL AWARD CRITERIA

While Part I explores the *why* and Part II the *what* of software eco-certification, Part III discusses the *how* by explaining what you need to know to measure the energy consumption of your software project and apply for the Blue Angel ecolabel. Specifically, in this section we provide a step-by-step guide to fullfilling the ABCs of the award criteria: (A) Resource & Energy Efficiency, (B) Potential Hardware Operating Life, and (C) User Autonomy.

PART I: ENVIRONMENTAL IMPACT OF SOFTWARE

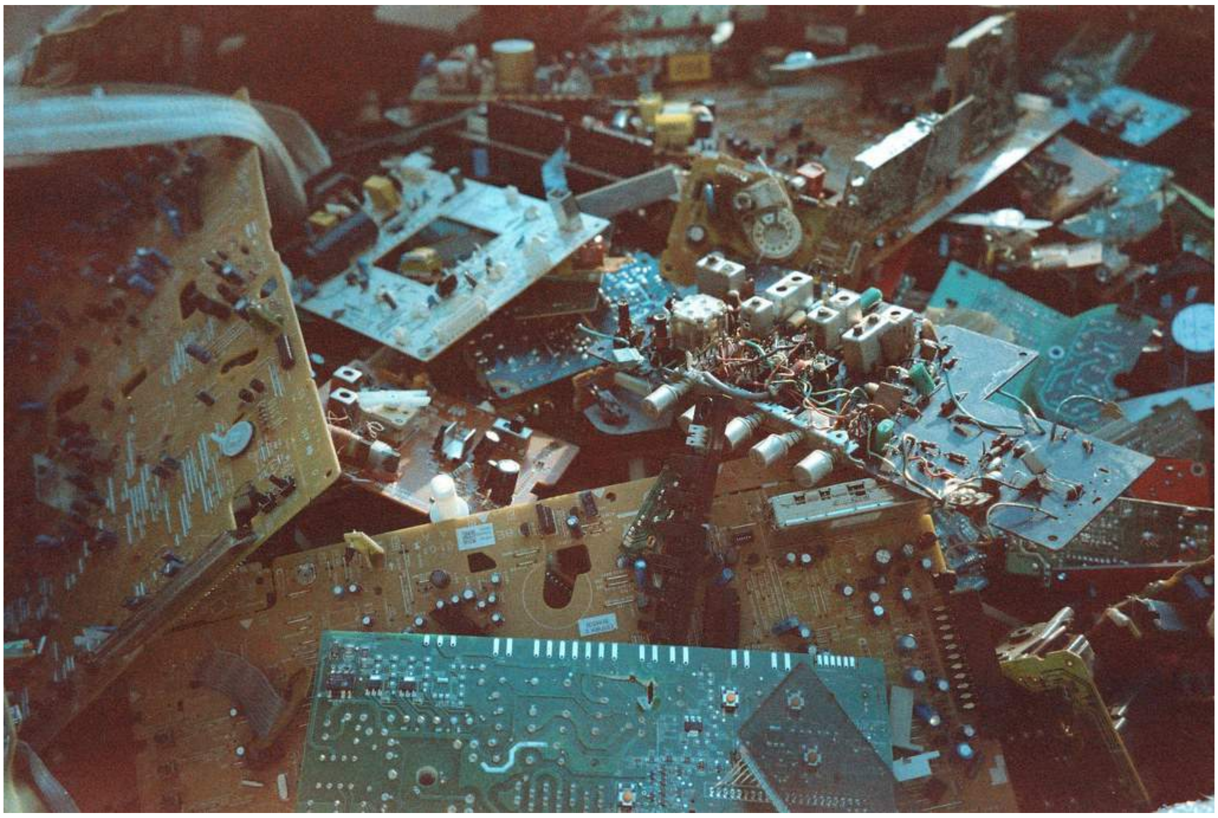


Figure 1: Photograph of e-waste. (Image published under a CC0-1.0 public domain license.)

In 2021, the Association of Computing Machinery (ACM), the oldest scientific and educational computing society in the world, released a Technology Policy Council report entitled "Computing And Climate Change". Among other findings, the brief explores the exponential increase in energy and resource consumption of internet-connected devices, both in production and use, as well as artificial intelligence. The report's estimates are staggering. In 2021 alone, the Information and Communication Technology (ICT) sector is estimated to contribute between 1.8–3.9% of global carbon emissions. To put this into perspective, this is on par with the global aviation industry, estimated to contribute 2.5% of all emissions. The report warns that if nothing is changed, by 2050 the carbon emissions attributable to the ICT sector will rise to more than 30% of all emissions globally.

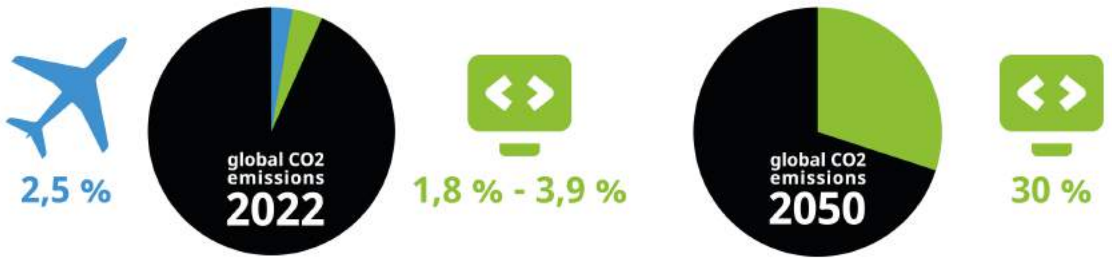


Figure 2: Two pie charts comparing (left) the greenhouse gas emissions of the aviation industry in blue with the ICT sector in green, and (right) projected estimates for emissions from the ICT sector by 2050 if nothing changes. Data is from ACM's 2021 Technology Policy Council report. (Image from KDE published under a CC-BY-SA-4.0 license. Airplane icon by Simon Child and IT icon by Sari Braga licensed under a CC-BY license. Design by Lana Lutz.)

In their conclusions, the authors acknowledge an inherent contradiction of digitization: while on the one hand, digital technology "can help mitigate climate change", it "must first cease contributing to it" (p. 1). ICT has revolutionized how we live, and it is often praised for bringing convenience and efficiency to our daily lives. Companies have leveraged digital technology for the efficient distribution of all sorts of consumer goods and the dematerialization of everyday products. Vehicles like cars, scooters, and bicycles are readily available for rental through apps, eliminating the need for individuals to own them in order to use them. Widespread video streaming capabilities means not having to produce or transport DVDs or Blu-ray discs to watch a movie, and driving to the rental store to pick one up on a Saturday night is a thing of the past. E-readers have replaced entire bookshelves. With the global SARS-CoV-2 pandemic making digitization an integral aspect of daily life, video conferences have been able to replace events once (almnost) exclusively held in person, including office meetings, international academic conferences, local piano recitals, and even first dates … all now possible from the comfort of one's own home with an internet-connected device.

For all the ways that technological developments have seemingly made our lives less material and less wasteful, and thus more convenient and more efficient, it might seem like the rapid pace of digitization does more good than harm when it comes to achieving sustainability goals.

But does it really?

The Internet and the devices we use to connect to it require infrastructure—real, physical hardware which demand energy and consume resources. Often overlooked are the environmental impacts of, for instance, the factories that produce these devices or the continent-spanning infrastructure that enables global communication. All of this requires energy in its day-to-day use. What's more, hardware that is no longer used either ends up in disposal centers for end-of-life treatment (which requires even more energy), or as e-waste that is both toxic to people and the environment. New devices are then produced and transported, in many cases unnecessarily.

Within this broader picture, the critical role that software plays in contributing to environmental harm may be overlooked. Indeed, in many cases it is software that determines the energy consumption and operating life of digital infrastructure.

> "A fact that is even more rarely appreciated is that the key to increasing energy efficiency and protecting natural resources lies not with the hardware but rather above all with the software." — Blue Angel Award Critera: Resource and Energy-Efficient Software Products (p. 5)

This handbook will take a closer look at some of the ways digital technology is contributing to environmental harm and the climate crisis. To be clear, this handbook is not anti-technology—undoubtedly, digitization has improved life in countless ways for vast numbers of people. But the ecological impacts of digital technology require us to think more deeply about the ways we use it and how we might use it more efficiently. And the good news is that through software design, developers can have an immediate and significant influence on many of the issues discussed here.

Throughout the text, and especially in later sections, the Blue Angel ecolabel for desktop software will serve as a benchmark for sustainable software design. But what does "Blue Angel" mean, anyway?

The Blue Angel (German: *Blauer Engel*) environmental label is the official environmental label of the German government. In 2020, the German Environment Agency (German: *Umweltbundesamt*, or UBA) released the award criteria for certifying desktop software products, the first environmental certification in the world to link transparency and user autonomy with sustainability. Free & Open Source Software, or FOSS, has a real advantage in achieving both of these aims. By the end of this manual, we hope you'll have a better understanding how.

But in order to address a problem effectively, we must first identify what the problem is. So let's first understand what is meant by the digital carbon footprint and how software we use everyday is involved.

Material Footprint Of Digital Technology

Digital technology is often (and erroneously) associated with being immaterial. When we send an email or upload data to the cloud, it can be easy to imagine our transmissions disappearing into the ether. But there is a very real, very material aspect to digitization, encompassing not only our physical devices such as smartphones and laptops, but also the processing plants for mined metals necessary to make them run, container ships that transport mass-produced hardware, or the cables and data centers which connect them to global networks. The 2018 report "Lean ICT: Towards Digital Sobriety" describes the issue so:

> [T]he material footprint of digital technology is largely underestimated by its users, given the miniaturization of equipment and the "invisibility" of the infrastructures used. This phenomenon is reinforced by the widespread availability of services on the "Cloud", which makes the physical reality of uses all the more imperceptible and leads to underestimating the direct environmental impacts of digital technology. (p. 10)

To bring the tangible reality of the "cloud" back down to earth—underwater, even—we need to shift our perspective to the hidden infrastructure that provides the basis for our digital lives. As one New York Times article quipped, "people think that data is in the cloud, but it's not. It's in the ocean", referring to the underseas communication cables spanning the globe. Although data networks may be largely underwater, the carbon emissions of ICT have dire consequences for all natural environments world-wide. At COP27 in November 2022, United Nations Secretary General António Guterres underscored the urgency of the moment when he stated: "We are on a highway to climate hell with our foot on the accelerator".

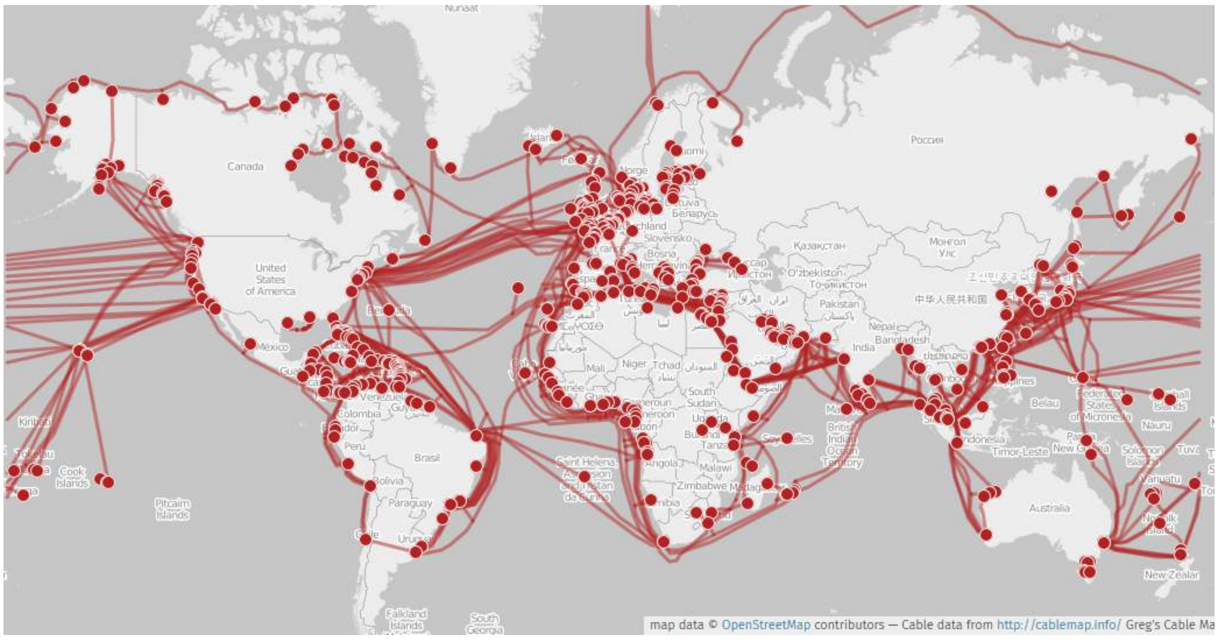

map data © OpenStreetMap contributors — Cable data from http://cablemap.info/ Greg's Cable Ma

Figure 3: World map of submarine communication cables. (Cable data by Greg Mahlknecht, KML file released under GPLv3; world map by Openstreetmap contributors.)

Within the ICT sector, what is contributing to this atmospheric rise of CO2?

Between 2012 and 2018, the energy demand of artificial intelligence (AI) increased 300,000 times, and it currently doubles every few months. It is estimated that training a single AI model (such as a model used in machine translation or language modeling) can require the energy equivalent of flying round trip from New York to San Francisco … 300 times (that's about 626,000 pounds of CO2)! Blockchain technology is another notorious contributor to exploding energy consumption—specifically, proof of work systems such as Bitcoin, which Harvard Business Review reports requires as much energy as entire countries like Sweden or Malaysia.

At the same time, we use more digital devices than ever before. The number of internet-connected devices, including laptops and smartphones, but also smart TVs, home assistants, and other IoT devices, is rapidly growing, and expected to surpass 75 billion by 2025. That's just about 10 devices for every person on earth (although the global distribution of these devices is far from even). Globally, smartphone adoption has increased rapidly, as well as the demands on resources needed to manufacture new and increasingly powerful devices. The production of these devices, including mining for the rare earth metals necessary to make them work, as well as their transporation, use, and eventual disposal all consume copious amounts of energy.

It is crucial to underscore here, however, that energy consumption is not the same as carbon emissions. Carbon emissions depend on the particular mix of fuels used for generating electricity in a particular region, referred to as the electricity or power generation mix. As an example, for the European Union's energy supply in 2016, the power generation mix included 32.9% oil, 23.9% gas, 14.9% coal, 13.7% nuclear power, and 14.5% renewables. With the energy crisis of 2022, the energy mix in the EU has changed—in some cases for the better long-term, in some for the worse short-term. Relative carbon emissions will depend on this mix: for example, energy consumption from 100% carbon-neutral energy sources contribute no direct CO2 emissions.

Relative Harm: When Less Is Not More

Digitization is often associated with "dematerialization": printing concert or travel tickets on paper is no longer necessary, as they can be simply be downloaded and presented on one's smartphone; photographs are not collected in over-stuffed shoeboxes, but on a small tablet or hard drive; thousands of films and TV series are streamed on laptops, making movie collections a thing of the past. In many cases, one device, a smartphone, is used for all of the above—and much, much more.

Each of those material objects were once a major part of our daily lives … but today, they are simply no longer necessary. That must be better for the earth, no?

While digital devices might reduce some forms of waste, estimating the true environmental impact of digital technologies requires accounting for the entire life cycle of an item. This includes the costs of producing and transporting digital devices (to and from the shop, as well as to the landfill), or the costs of remediating environmental harm caused by e-waste. This is especially true when considering the collective carbon footprint of our digital technologies, since in some cases the production of devices, together with their transportation and end of life treatment, contribute more greenhouse gas emissions than the devices' use over their *entire operating life*. To illustrate this, consider Apple's 2019 Environmental Responsibility Report, which estimates that Apple contributed 25.2 million metric tons of CO2 in 2018 (p. 9). Most of this—a whopping eighty percent (!!!)—comes from production (74%), transportation (5%) and end-of-life treatment (<1%). Only 19% comes from the actual usage of the devices. This is the case not only for Apple. Similar numbers will be found for comparable devices.

So when are the costs of manufacturing a digital device to replace all of those analogue objects worth it? The book "Smarte Grüne Welt" (English: *Smart Green World*) by Steffen Lange and Tilman Santarius (2018) explores the difficulty of accounting for relative environmental harm when trying to answer such questions. Consider this excerpt, in which the authors explore the environmental impact of printing paper books vs. manufacturing e-readers (pp. 29–31; unofficial translation):

> Making electronic devices is obviously more energy-intensive and resource-intensive than printing a single book. For example, the production of an e-reader, usually weighing less than 200 grams, accounts for about 15 kilograms of different materials (especially non-renewable metals and rare earths), 300 liters of water, and 170 kilograms of the greenhouse gas carbon dioxide. However, it is not just the quantities of input and output materials that are decisive, but also their environmental impact. There are great differences between e-readers and books, especially in the toxicity of materials and manufacturing processes. It is true that the paper industry in many countries (still) has very negative environmental effects, for example when chlorine or acids poison local waters. However, the environmental effects of the electronics industry are sometimes devastating: e-readers and other IT products include brominated fire retardants, phthalates, beryllium, and numerous other chemical substances that are severely harmful to health and the environment. Not to mention the social consequences, such as the sometimes miserable working conditions under which cobalt, palladium, tantalum, and other resources of digital devices are initially extracted in dictatorships such as the Republic of Congo or in other countries of the global South—and then disposed of there at the end of life as environmentally-harmful e-waste.
>
> Despite all of this, the e-reader may be better than the book. This ultimately depends on two factors: How many books are read on the e-reader over its lifetime—and how many people share an analog book? In order for the high environmental costs of the e-reader's production to pay off ecologically, a certain number of books must be read on it. This is the case after 30 to 60 books—depending on the thickness of the book and depending on the environmental indicator. If you read less than this number of books on an e-reader, it is better to choose the paper form. If you go beyond this, each other book on the e-reader is ecologically better than its analog counterpart. Furthermore, how objects are used is crucial [...]: If it is assumed that someone buys a book and does not let anyone else look at it, then a file on the e-reader is around five times more energy-efficient than a book. This advantage, however, disappears when several people share a book.

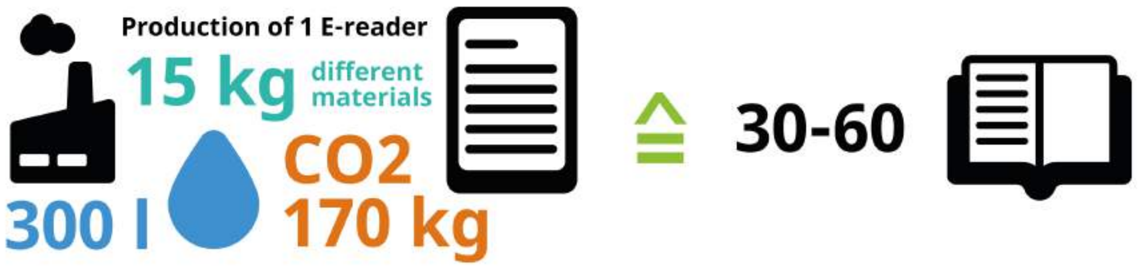


Figure 4: To produce one e-reader, 15 kilograms of different materials, 300 liters of water, and 170 kilograms of the greenhouse gas carbon dioxide are needed. If you read fewer than 30–60 books on the e-reader device, it may be environmentally less harmful to choose the paper form. (Image from KDE published under a CC-BY-SA-4.0 license. Design by Lana Lutz.)

So, does replacing physical objects with digital technology result in a reduced environmental impact? Well, it depends. For example, if you buy an e-reader, will you read 30–60 books before discarding the device? A Gallup poll found that in 2021, 57% of US Americans read fewer than 5 books a year, and 15% read between 6-10 books. This means for over two-thirds of the US population, an e-reader would need to be used for five to ten years in order to be the more eco-friendly choice. But how many consumers upgrade to the next, shiny new device well before then?

It's also worth asking whether devices will remain supported for the amount of time it would take to potentially be the less harmful choice. At the end of 2022, Wikipedia's list of discontinued e-readers included seventy-one devices. According to the list, the average lifespan, i.e., from intro year to end year, was 1.5 years, considerably shorter than the minimum five-year period—let alone a decade! How many of those e-readers were still functioning, but ended up in a landfill because of discontinued software support? This kind of hardware obsolescence is an important contributor to environmental harm, whether in the form of e-waste or the carbon emissions associated with producing the device. As Lange and Santarius write: "it is questionable whether all e-readers sold, before they break or become technically obsolete again, are used so intensively on average that an overall ecological benefit is achieved" (p. 31; unofficial translation).

A "Tsunami Of E-Waste"

At seven meter's tall, the "WEEE Man" is a giant. Taking its name from the 2003 Waste Electrical and Electronic Equipment (WEEE) directive, which sets collection, recycling, and recovery targets for e-waste in the EU,[1] the statue is made from 3.3 metric tons of electrical waste, or the average amount of

[1]See Seiwert & Zaczyk 2021: p. 13 for details; see also Appendix A 2 on p. 46 for a Python script to automate some of these tasks.

e-waste that one UK individual creates in a lifetime.

Figure 5: Image of “WEEE Man” statue, which is made from 3.3 metric tons of electrical waste, or the average amount of e-waste that one UK individual creates in a lifetime. (Photograph by James T.M. Towill and published under a CC-BY-SA-2.0 license.)

E-waste is considered the “fastest-growing waste stream in the world”, with 44.7 million metric tons generated in 2016—equivalent to 4,500 Eiffel Towers, which, when stacked, is 17 times higher than Mount Everest. In 2018, an estimated 50 million metric tons of e-waste was reported, motivating the UN to refer to a “tsunami of e-waste rolling out over the world”. The numbers continue to rise: in 2021, an estimated 57 million metric tons of e-waste was generated globally. Less than 20 percent of it is collected and recycled, and although it makes up only 2% of trash in landfills, it contributes to almost 70% of the toxic waste found there.

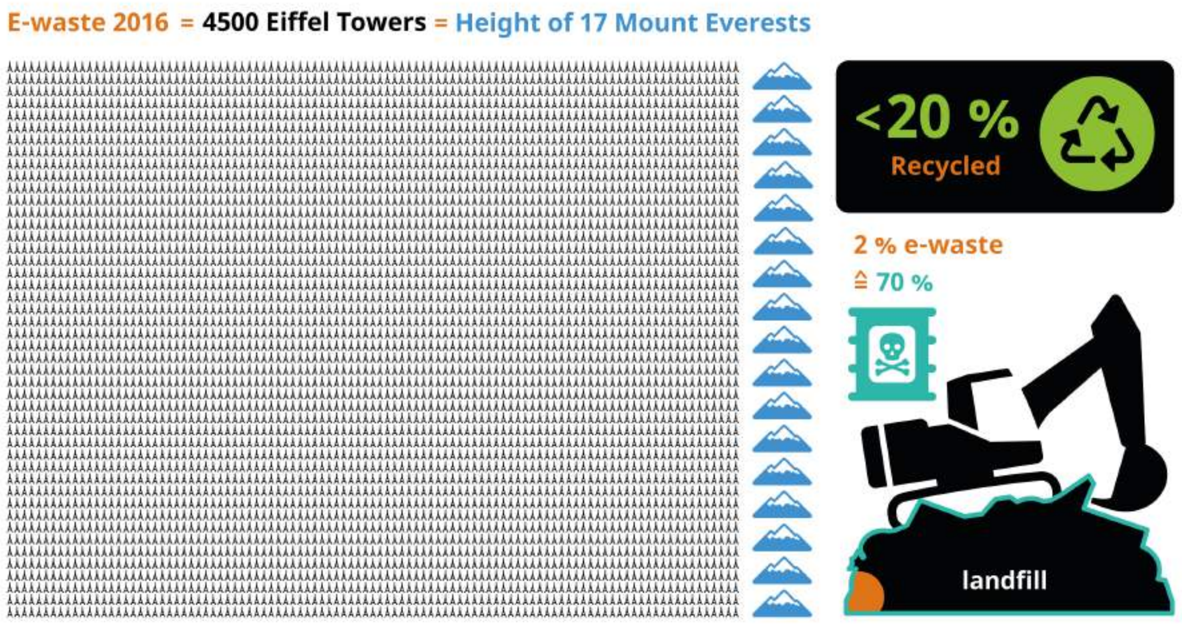

E-waste 2016 = 4500 Eiffel Towers = Height of 17 Mount Everests
<20 %
Recycled
2 % e-waste
≙ 70 %
landfill

Figure 6: In 2016, 44.7 million metric tons of e-waste was generated. This is estimated to be equivalent to 4,500 Eiffel Towers, which, when stacked, is 17 times higher than Mount Everest. Less than 20% of e-waste is collected and recycled. Although e-waste makes up less than 2% of trash in landfills, it contributes to almost 70% of the toxic waste found in them. (Image from KDE published under a CC-BY-SA-4.0 license. Eiffel Tower icon by Daniela Baptista, moutain icon by Samy Menai, recycling icon by Kosong Tujuh, Excavator icon by Peter van Driel, Poison icon by Adrien Coquet, all licensed under a CC-BY license. Design by Lana Lutz.)

The environmental impacts of e-waste are enormous. Electronic scrap components like CPUs contain potentially harmful materials such as lead, cadmium, beryllium, or brominated flame retardants. The end of life treatment of e-waste can also involve significant risk to the health of workers and their communities. What's more, scavengers risk their health for the discarded precious metals in laptops and smartphones "laced with lead, mercury or other toxic substances". The process of dismantling and disposing of e-waste has led to a number of environmental impacts in developing countries. Liquid and atmospheric emissions end up in bodies of water, groundwater, soil, and air—and thus, also in land and sea animals, in crops eaten by both animals and humans, and in drinking water. This pollution is a crucial aspect of digital technology's environmental harm.

Figure 7: A young man is pictured burning electrical wires to recover copper at Agbogbloshie, Ghana, as another metal scrap worker arrives with more wires to be burned. (Image by Muntaka Chasant, published under a CC-BY-SA-4.0 license.)

Look To The Software

What is the cause of all this e-waste, and why do digital devices that still work end up in landfills? Software engineering has an important but often unseen role in driving our digital consumption patterns. Manufacturers regularly encourage consumers to purchase new devices, often unnecessarily; indeed, they may even enforce it through software design. And because of licensing restrictions on software use and certain forms of vendor dependencies, end users can do little about it. In short, these are largely economic—and not technological—reasons for functioning hardware becoming e-waste.

Programmed obsolescence and software lock-out result in unusable hardware. Abandonware released under a proprietary license can, at best, leave users vulnerable to viruses and other malware, and, at worst, simply stop working, without an alternative to keep their devices running. Underlying infrastructure which people depend on to run an application—such as software license servers used for access control by software vendors—can go offline, sometimes permanently. In these cases, users may not be able to continue using "outdated" hardware even if they wanted to.

Feature creep and other forms of software bloat may render less-powerful hardware obsolete, even though costumers never requested the extra functionalities or might wish to remove them if they could.

> “Processing power has doubled about every two years since 1970. This means that functions are processed twice as fast and thus less energy is required for the same functions. A similar improvement in efficiency cannot be observed in the field of software. [...] The availability of more and more powerful hardware has resulted in software becoming more and more bloated from version to version so that more resources are required **for only minimal or even no enhancement of the functionality**.” [emphasis added] — Blue Angel Award Critera: Resource and Energy-Efficient Software Products (p. 5)

In these cases, vendor dependencies and user restrictions attributable to software design and licensing mean that still-functioning devices are tossed in the garbage heap while more resources are consumed to produce and transport new ones.

And when not rendering functioning hardware unusuable, software design which requires license servers or results in feature creep, etc., also result in higher energy consumption when using the software. For example, one study published by the German Environment Agency found that two applications achiving the exact same result can have drastically different energy profiles.

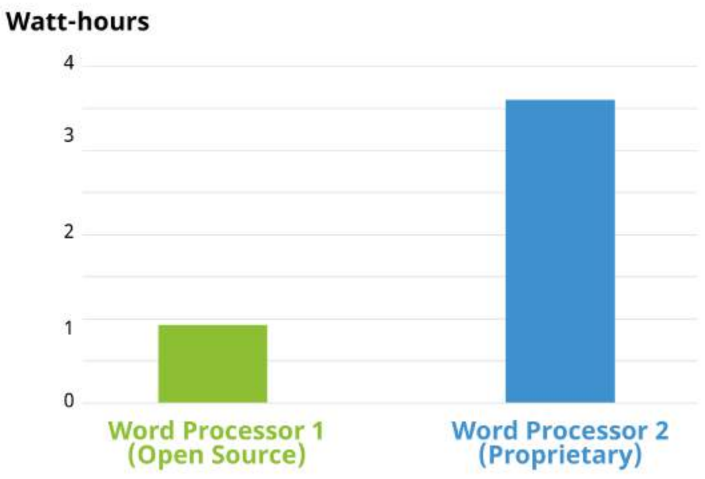


Figure 8: Bar plot comparing two word processors during execution of a Standard Usage Scenario (based on Figure 1 on page 24 in the German Environment Agency report). Word Processor 1 is an Open Source program. This word processor consumes four times *less* energy than Word Processor 2, a proprietary program. (Image from KDE published under a CC-BY-SA-4.0 license.)

The report compared two word processing programs: Word Processor 1 is identified as Open Source, while Word Processor 2 as a proprietary software product. Both computer programs ran the same sequence of commands through a Standard Usage Scenario (SUS) script, corresponding to “the most representative use of the respective software over a defined period of time” (p. 23). We’ll return to usage scenarios in Section III of this handbook. For now, what’s important to note is the massive difference in energy use. Running Word Processor 2 consumed 4 times the energy compared to Word Processor 1— again, and this cannot be stressed enough, to achieve the same result!

Looking closer at the two word processors’ power use over time, it’s also clear how the two software products behave quite differently … and perhaps contrary to what you might expect. Consider the plot below, in which the power consumption while running the usage scenario’s sequence of commands is shown. At the 440-second mark, the script calls for both word processors to save the document and then stops calling for further action. As you can see, Word Processor 1 goes idle (as one might expect). By contrast, Word Processor 2 continues working, consuming power even though the script has ended.

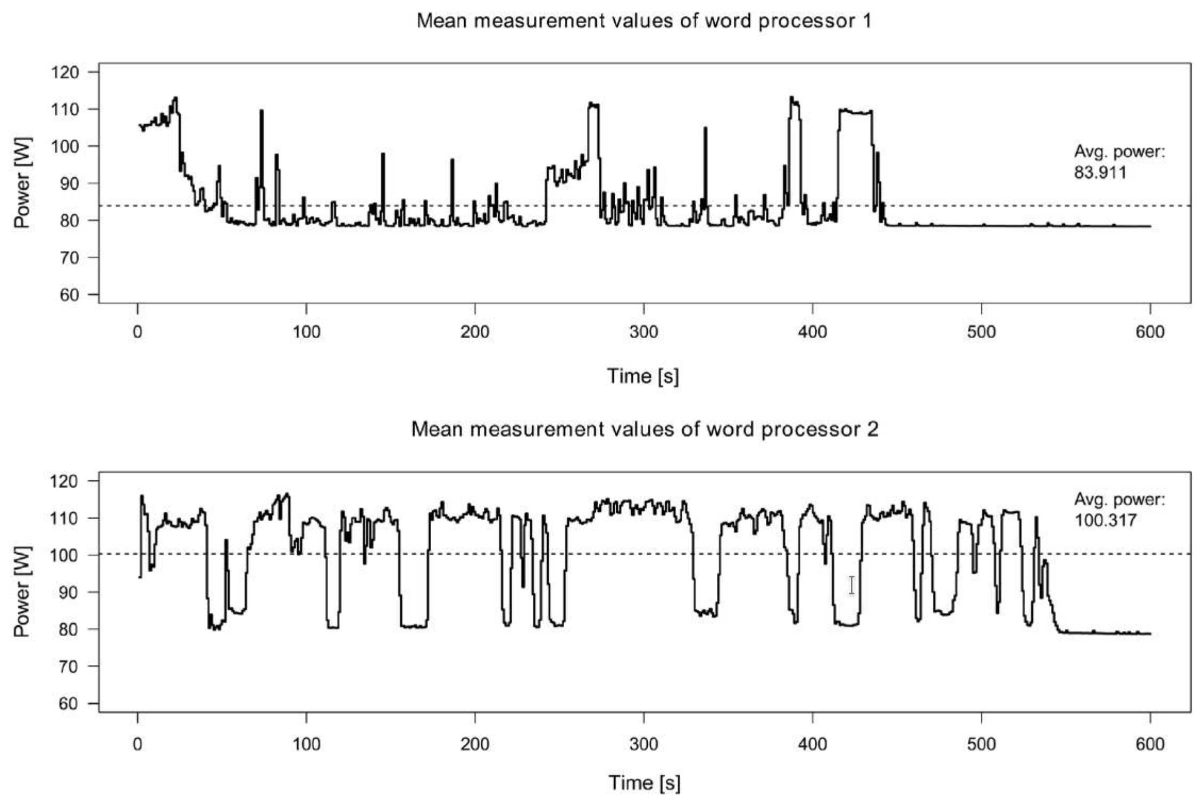


Figure 9: Plot comparing two word processors over time when running a Standard Usage Scenario (SUS). Word processor 1 (*top*) goes into an idle state when not doing anything, seen most clearly after the document is saved at the 440-second mark and no other actions are called by the SUS script. By comparison, Word Processor 2 (*bottom*) rarely goes idle, even after saving the document and no other actions are called. (Screenshot from Gröger et al. 2018 article published under CC-BY-NC-ND license.)

It's worth asking what the additional activities from 440 to 600 milliseconds is for: Are the actions of Word Processor 2 necessary for the functionality of the software? Is the word processor collecting and transmitting user data? If so, do users have a way to opt out of these types of analytics? Indeed, user autonomy, such as the capability of turning off of unwanted data use, can make a big difference on the energy profile of a software product. Data mining, third-party tracking, personalized engagement-maximizing algorithms, and advertising are significant drivers of energy consumption. Collecting and analyzing user data and training algorithms on it require computing power and infrastructure.

In 2021, researchers in the EU estimated the environmental costs of tracking which users cannot opt out of, what they refer to as "unwanted data use" in the report "Carbon footprint of unwanted data-use by smartphones: An analysis for the EU". The carbon footprint of this tracking and advertisements by smartphones—between 3 and 8 million metric tons a year in the EU alone—is "equal to the carbon footprint of between 370 and 950 thousand EU citizens" (at its worse roughly the annual footprint of a city like Turin, Italy or Lisbon, Portugal). The report also points out that circa 60% of European smartphone users indicate they would opt out of tracking and block advertisements when possible. That's an awful lot of energy consumption for something most users don't want in the first place!

"The Very Contrary Is The Truth": Jevons' Paradox

Increases in software efficiency alone do not necessarily translate to lighter environmental footprints. For example, the "rebound effect" (also known as the "take-back effect") describes how efficiency improvements can lead to changes in usage that decrease or even negate the original gains.

Imagine that a change in software results in a 5% improvement in energy efficiency. However, because of the increased energy savings, you might end up using the software more, leading to less energy savings overall. Let's say that due to your increased usage, the overall energy usage of the software drops by just 1%. In this case, the rebound effect is 80% ((5-1)/5): in other words, those original efficiency gains have decreased by 80%, practically negating any savings from the improved efficiency!

If the rebound effect goes over 100%, meaning more energy ends up being used than before, this is referred to as Jevons' Paradox, or "back-fire". The concept comes from the English economist William Stanley Jevons, who in 1865 recognized that technological improvements in coal-use actually increased coal consumption across industries. Jevons concluded that:

> "It is a confusion of ideas to suppose that the **economical use of fuel is equivalent to diminished consumption**. The very contrary is the truth." [emphasis added]

A practical interpretation of this paradox is that efficiency gains must be combined with conservation practices in order to have a meaningful effect, lest one ends up consuming more than before. The ACM report references at the beginning of this section makes a similar point: "Computing-enabled efficiencies must be coupled with slashed energy demand to reduce ICT sector carbon emissions" (p. 1). In other words, both software engineering AND user behavior are crucial elements to consider when combatting the environmental harm driven by software.

Is All Of This Worth It?

Looking at the bigger picture, software's contribution to environmental harm and global greenhouse gas emissions may be less significant in comparison to other industries. Is it worth focusing on the environmental impacts of software, a problem that might seem relatively small in the grand scheme of things?

There are a few things we might consider here. The first is a rejection of the "not as bad as" fallacy, also known as "Appeal to Worse Problems". The general argument is that software's contributions to global CO2 emissions may not be as bad as another industry's, and therefore it is not worth focusing on. What's wrong with this argument is that although another industry may be worse, this does not negate the fact that software is responsible for causing serious environmental harm. What's more, reducing software's environmental impact can also help to reduce other industry's impacts at the same time: the "not as bad as" fallacy suggests a false choice between addressing one problem or the other, when in fact ecologically-friendly software design is just one piece of a bigger puzzle.

Let's not be like XKCD's White Hat, and appeal to worse problems as an excuse not to do anything:

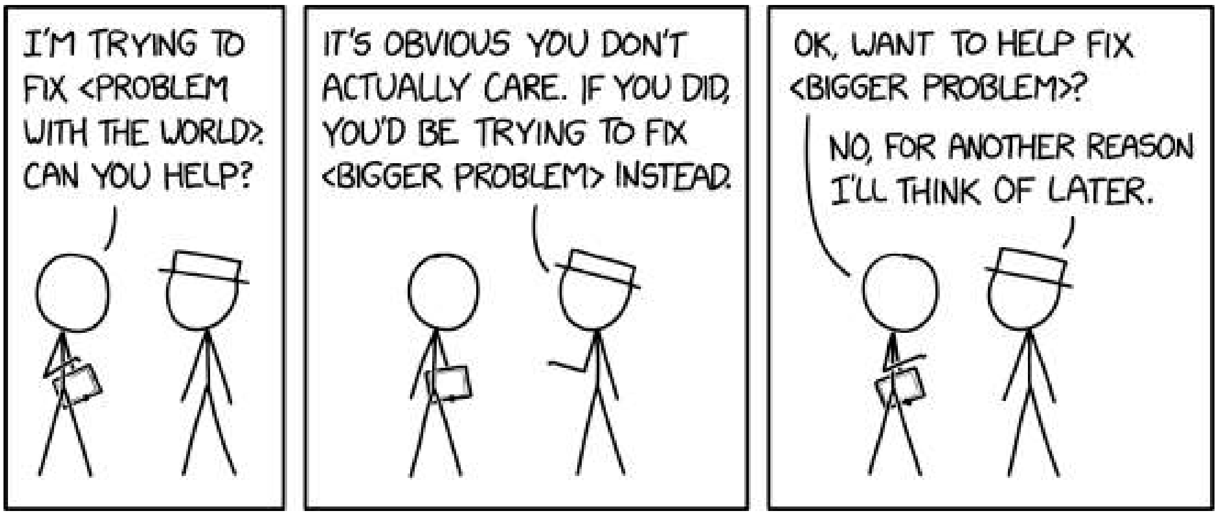

I'M TRYING TO FIX <PROBLEM WITH THE WORLD>. CAN YOU HELP?
IT'S OBVIOUS YOU DON'T ACTUALLY CARE. IF YOU DID, YOU'D BE TRYING TO FIX <BIGGER PROBLEM> INSTEAD.
OK, WANT TO HELP FIX <BIGGER PROBLEM>?
NO, FOR ANOTHER REASON I'LL THINK OF LATER.

Figure 10: XKCD comic "2368: Bigger Problem". (Comic published under a CC-BY-NC-2.5 license.)

Second, focusing only on fixing the "biggest problem" is not necessarily the most effective strategy—it's also important to weigh the likelihood of success when addressing a problem, as well as the time and resources required to do so. Free & Open Source Software, with its focus on user autonomy and transparency, provides unique opportunities for users, communities, and organizations to directly address intertwined social and ecological issues. FOSS can be adapted, updated, and maintained at lower cost and without vendor dependencies or artifical restrictions.

Third, it's hopefully clear by now that software *does* have significant impacts on energy consumption and the production of waste, both of which have consequences for the environment. When considered at scale, changes in software design can have savings comparable to the annual energy consumption of entire cities. For instance, watch Detlef Thoms (04:20–06:10) do back-of-the-envelope calculations to go from one CPU-second reduction, equivalent to about a 10 watt-second savings, to 95 thousand megawatt hour savings simply by scaling up—this is roughly the annual energy consumption of over 30-thousand two-person households.

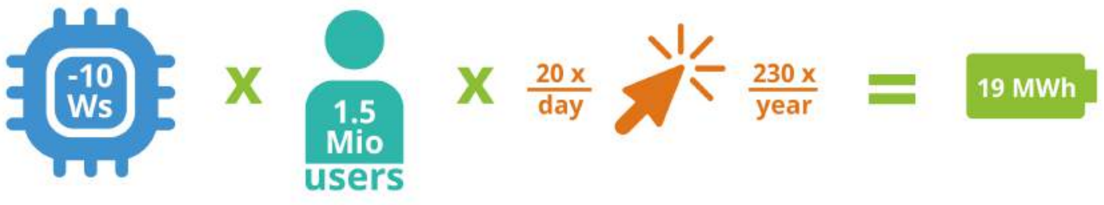


Figure 11: One CPU-second reduction is roughly equivalent to 10 watt-second savings. If 1.5 million users use the software, and there are 20 transactions a day with the savings over 230 work days, that is about 19 megawatt-hours savings. (Image from KDE published under a CC-BY-SA-4.0 license. CPU icon by Azland Studio and Cursor icon by Alice-vector licensed under a CC-BY license. Design by Lana Lutz.)

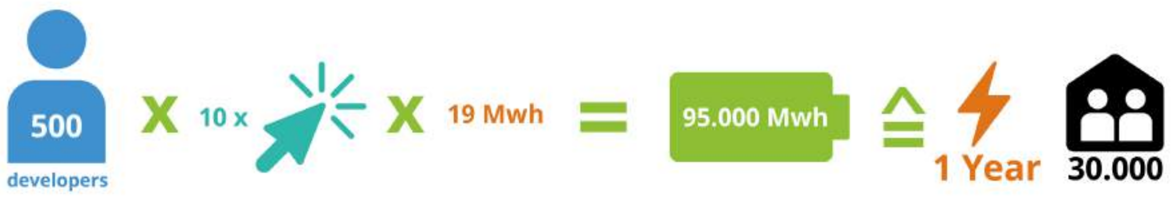


Figure 12: If 500 developers make 10 one CPU-second reductions, this is equal to 95 thousand megawatt-hours of savings, or the energy consumption of 30-thousand two-person households over one year. (Image from KDE published under a CC-BY-SA-4.0 license. Cursor icon by Alice-vector licensed under a CC-BY license. Design by Lana Lutz.)

As Detlef Thoms states in the video: “Often, it is a quite manageable set of decisions which lead to significant differences in power consumption”.

Finally, in order to make claims about relative harm, it is necessary to first have estimates about actual effects. Since research in the area of software’s energy and resource consumption is still quite new, we often do not have data to make data-driven claims. With this handbook, and with the Blue Angel ecolabel as a guide, KDE hopes to help change that.

Changing our software may seem like a small gesture in addressing an issue as complex as climate change. It’s also clear that simply changing our individual consumption patterns may not be sufficient on its own (what’s worse, evidence suggests that major contributors to global greenhouse gas emissions—such as ExxonMobile—have embraced a rhetoric of consumer individual responsibility in order to deflect from their own role in the crisis). It's true: a zero-emissions future will require fundamental shifts in how we live, and that responsibility can't be managed at an individual level. But consider what anthropologist Margaret Mead once observed:

> "Never doubt that a small group of thoughtful, committed citizens can change the world; indeed, it's the only thing that ever has."

Structural change happens when dedicated, passionate people organize to confront pressing societal issues. With decades of experience successfully bringing global communities together to work toward common goals, Free & Open Source Software can be a powerful force for combatting the environmental impact of digitization. We know how to organize—now, it's a matter of turning plans into practice, goals into reality. Let's unite to combat software-driven environmental harm. Let's foster a culture of digital sustainability in our software communities. Let's build energy and resource efficient software, together!

A Note On Sources

Some material in this section is based directly on text from two Wikipedia articles: (i) "Waste Electrical and Electronic Equipment Directive" and (ii) "Electronic waste". Both texts are released under the Creative Commons Attribution-Share-Alike License 3.0.

PART II: ECO-CERTIFYING DESKTOP SOFTWARE

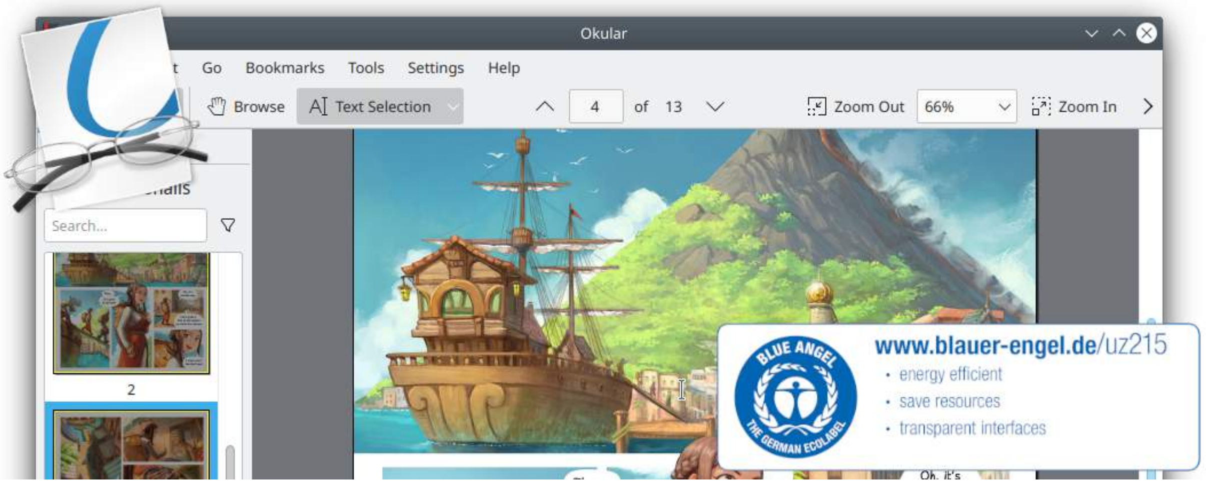


Figure 13: Okular, KDE's popular multi-platform PDF reader and universal document viewer, was awarded the Blue Angel ecolabel in March 2022. (Image from KDE published under a CC-BY-4.0 license.)

What do construction products, toilet paper, and software have in common?

Each of these can be eco-certified by the Blue Angel environmental label, the official environmental label of the German government!

The Blue Angel ecolabel is awarded to a range of products and services, from paper products and construction materials to printers, and it certifies that the product meets a list of stringent requirements for being environmentally friendly over a product's life cycle. In 2020, the German Environment Agency extended the award criteria to include software products, the first environmental certification to link transparency and user autonomy with sustainability.

Specifically, the eco-certification criteria requires transparency about the software's energy consumption when in use while ensuring that the software is capable of running on older hardware. Moreover, the criteria also include a list of requirements related to user autonomy which will reduce the environmental impact of software.

This section provides a broad overview of the Blue Angel and the ABCs of the award criteria for desktop software. It also demonstrates how meeting the award criteria can reduce environmental harm. Here we will zoom in on details about the user autonomy requirements of the Blue Angel award criteria, which we will return to in Part III. But first, a brief introduction to the Blue Angel and the KDE Eco initiative.

The Blue Angel

Introduced in 1978, the Blue Angel is the first ecolabel worldwide and the official environmental label awarded by the German government. The label is adminstered by Germany's Federal Ministry for the Environment, Nature Conservation, Nuclear Safety, and Consumer Protection (German: *Bundesministerium für Umwelt, Naturschutz, nukleare Sicherheit und Verbraucherschutz*, or BMUV). The Blue Angel ecolabel is also a member of the Global Ecolabelling Network (GEN), an international network of Type I ecolabels that at the time of writing has 37 members across nearly 60 countries.

Figure 14: Logo of the Blue Angel ecolabel. The logo is intentionally designed to correspond to the logo of the United Nations Environment Programme. This reflects the aim of the German government to embed the UNEP goals in Germany. (Image published under a CC-BY-SA-4.0 license.)

The Blue Angel was not the first Type I ecolabel for software—the Hong Kong Green Council, also a member of the Global Ecolabelling Network, released criteria in 2010 for Green IT software. But the Blue Angel ecolabel criteria are the first to identify a process for measuring software's energy consumption and the first to specify ways that user independence reduces environmental harm.

What is a Type I environmental label?, you may be wondering. For Type I labels, the entire life cycle of the product is taken into account, and compliance with the award criteria is assessed by a third-party (compliance with Type II environmental labels, on the other hand, is self-declared and do not require any third-party auditing).

The Blue Angel ecolabel has been awarded to around 100 product groups and services across a variety of sectors, including paper products, building products, furnishings, clothing, washing and cleaning agents, cleaning services, household chemicals, packaging, vehicles, energy and heating, and household electrical devices. As of 2022, with the eco-certification of KDE's popular PDF and universal document reader Okular, that list also includes desktop software.

The award criteria for certification are developed transparently by the German Environment Agency. The process includes the Environmental Label Jury, a body made up of suppliers as well as civil society organizations and research institutions. The independent auditor RAL GmbH assesses compliance with the award criteria and awards the seal. Importantly, the Blue Angel does not certify that a product is completely harmless. Instead, certified products represent a "lesser evil" with respect to environmental harm—this can be summed up with the motto '*as little as possible, as much as necessary*'. Rather than compare different products, the Blue Angel ecolabel indicates that a product fulfills a list of requirements for a specific category.

The ABCs Of The Award Criteria For Desktop Software

The Blue Angel's award criteria for "Resource and Energy-Efficient Software Products" were released in January 2020.

There are two primary objectives of the Blue Angel for software: (i) to award software with lower performance requirements such that "longer operating lives for [. . .] hardware are possible"; and (ii) to recognize products which "stand out due to their high level of transparency and give users greater freedom in their use of the software" (p. 6). To achieve this, there are three main categories, referred to here as the ABCs of the award criteria.

ABCs of Certification Criteria
Resource & Energy Efficiency
Potential Hardware Operating Life
User Autonomy

Figure 15: The ABCs of the award criteria. (Image from KDE published under a CC-BY-SA-4.0 license. Time icon by Adrien Coquet licensed under a CC-BY license. Design by Lana Lutz.)

- (A) Resource & Energy Efficiency
- (B) Potential Hardware Operating Life
- (C) User Autonomy

The criteria in category (A) requires that the energy consumption of a software product be measured and reported, and states that the energy consumption of the software cannot increase by more than 10% from the time of its certification. Energy consumption data is measured using an external power meter and accounts for other hardware performance information, such as CPU usage or network traffic, while running the software in a representative way.

The criteria in category (B) ensure that the software has low-enough performance requirements to run on older, less powerful hardware at least five years old. Compliance entails a declaration of backward compatibility, with details about the hardware on which the software runs and the required software stack.

Finally, the criteria in category (C) ensure that users have an influence on the energy consumption and resource-conserving use of their software. There are eight categories for the autonomy criteria.

1. Data Formats — *Interoperability To Give Users A Choice*

   Data formats should not be used by vendors to lock users into using a specific computer program, nor should they impose onerous switching costs. Interoperable data formats prevent users from being stuck using a program that consumes a high amount of energy, when a more efficient program can achieve the same results with fewer hardware demands. Users should be able to easily change programs and still access all of their data.

2. Transparency — *Removing User Dependencies For Long-Term Use*

   Transparency in software code and application interfaces means removing any dependencies on a particular company or organization. It also means removing restrictions on the short and long-term use of the software. When developers decide to end support for their software, either security updates should continue to be provided (see below) or the source code should be made publicly available so third parties can continue providing support for the software. Furthermore, enhancing the functionality of software must not be limited by restrictive or undocumented application interfaces (APIs).

3. Continuity Of Support — *Security Updates To Prevent E-Waste*

   Dependency on suppliers for essential updates should not leave users with abandonware, software which cannot be used without presenting serious disadvantages for the user such as vulnerabilities to malware. Security updates should be provided for up to five years after software development is discontinued. Moreover, security updates should be separable from feature updates so that users are not coerced into adopting unwanted functionalities, e.g., feature creep and other forms of software bloat.

4. Uninstallability — *Removing Unwanted Software To Increase Efficiency*

   Being able to completely uninstall software that is not needed has ecological benefits. Software bloat, feature creep, and unwanted software components can create inefficiencies by occupying memory, wasting processing time, adding disk usage, consuming storage, and causing delays at system startup and shutdown. When a user no longer wishes to continue using a software product, it must be possible to completely purge it from the system while keeping all user-generated data.

5. Offline Capability — *Removing Dependencies To Keep Software Usable Long-Term And Decrease Energy Consumption*

   Use of the software should be possible without an internet connection—unless, of course, a network connection is necessary for the software's intended functionality. License servers and other forms of access control restrict the use of an application in ways that are unnecessary to the software's intended functionality. When a server goes down or there is Internet outage, such access control locks people out from using their software, possibly permanently. Moreover, such dependencies depend on network traffic and thus consume energy beyond that which is needed for the intended purpose of the software.

6. Modularity — *To Decrease Memory And Energy Demands*

   Users should be able to install only what they need. Non-essential functions increase memory and energy demands, making the software less efficient and perhaps unable to run on older hardware. People should have the ability to limit the range of software functions to those that they either want or require.

7. Freedom From Advertising — *Opting-Out To Reduce Energy Consumption*

   Discussed in Part I, unwanted data use in the European Union alone is roughly equivalent to the annual energy consumption of a city like Lisbon or Turin. Allowing users to opt out of ads reduces resource and energy demands on end-user devices as well as on the servers pushing the ads. Opting out also decreases data volume transmitted and thus reduces energy consumption of network traffic.

8. Documentation — *To Support Resource-Conserving, Continuous Use Of Software … And Therefore Hardware*

   Documentation is a prerequisite for the long-term viability of a software product. Documentation must also demonstrate the software's capacity for conserving resources. By documenting the criteria listed above, users can continue using software (and thus, hardware) in a sustainable way, while developers can maintain the software without dependencies or restrictions imposed by vendors.

Software products that comply with the award criteria are less likely to suffer from various forms of inefficiencies. This in turn can help mitigate the problem of e-waste: with software that doesn't drive hardware obsolescence, fewer devices need to be produced and shipped, which means fewer valuable metals that need to be mined and processed, which in turn results in a reduction of water and soil pollution. By ensuring user autonomy, developers can ensure that their software reduces environmental harm in more ways than one—whether by keeping devices in use for longer, or by reducing the software's use of energy and resources while in use.

With its focus on transparency in resource and energy-efficiency, hardware operating life, and user autonomy, the Blue Angel award criteria for software provides a comprehensive framework to begin a discussion around software sustainability. In FOSS communities, we often take user autonomy and transparency and their benefits for granted. Although being Free & Open Source Software is not a requirement to obtain the Blue Angel ecolabel, it is in this category that FOSS really shines—in so many ways, we are already at the forefront of sustainable software design!

First Eco-Certified Computer Program: KDE's Popular Document Reader Okular

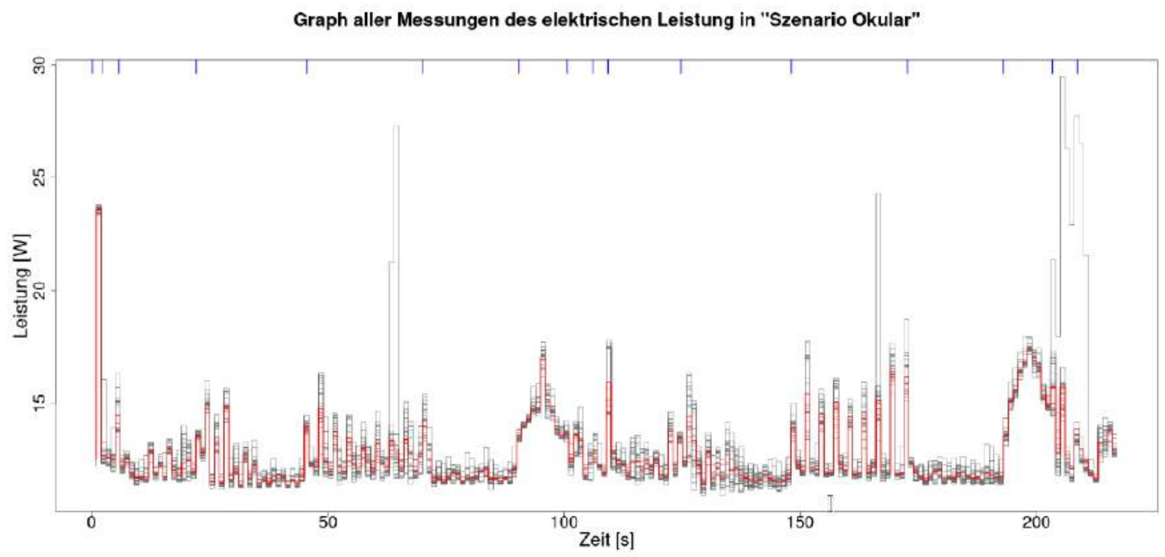


Figure 16: Okular's energy consumption report.

In February 2022 Okular, KDE's popular multi-platform PDF reader and universal document viewer, was the first software product to be officially recognized for sustainable software design as reflected in the Blue Angel award criteria. Okular is also the first eco-certified computer program within the Global Ecolabelling Network, representing nearly 60 countries.

Okular is just one software product maintained by KDE, a world-wide community of software engineers, artists, writers, translators, and creators who are committed to Free Software development. KDE maintains numerous FOSS products, including the Plasma desktop environment; the design app for painters and graphic artists, Krita; the GCompris suite of educational activities for children; Kdenlive, a professional video-editing software product; and of course Okular, a document viewer for PDFs, comics, scientific and academic papers, and technical drawings.

With KDE's long-standing mission and guiding vision since its founding in 1996, as well as the talent and capabilities of its community members, KDE is a pioneer in championing sustainable software. In 2021, KDE started KDE Eco, a project with the goal of putting KDE and Free Software at the forefront of sustainable software design. Sustainability is not new for Free & Open Source Software (FOSS)—the four freedoms have always made Free Software sustainable software. But now, the two pillars of FOSS—transparency and user autonomy—have wider recognition for their impacts on sustainability, and were incorporated into the sustainability criteria set by the German Environment Agency through the Blue Angel ecolabel.

Figure 17: Logo of the KDE Eco initiative. (Image from KDE published under a CC-BY-SA-4.0 license. Design by Lana Lutz.)

With the first ever eco-certified software product, the KDE community celebrates the achievement together with the wider Free Software community, as well as with the computer science department at Umwelt Campus Birkenfeld, where researchers measured the resource and energy-consumption of Okular and other KDE software.

The Blue Angel award criteria reflect KDE's values and those of the larger FOSS movement seamlessly. Free & Open Source Software guarantees transparency and hands control over to users, rather than obligating them to work with certain vendors or service providers. This allows users to decide what they want from the software they use and in turn, make decisions about the hardware they use as well. Users might be able to reduce the energy consumption of their programs with little or no loss in functionality, installing only what they need, no more and no less; they can also avoid invasive advertising or data-mining options which run processes in the background, further consuming resources on the device and in the network. As for FOSS developers, they typically continue to support hardware that the industry would be eager to make obsolete, providing users with up-to-date and secure software for devices that might otherwise be discarded as e-waste and end up polluting landfills.

Released under the GPLv2+ license, Okular is FOSS and therefore already fulfilling many of the user autonomy criteria necessary to obtain the Blue Angel seal of approval. Further work was carried out to make Okular fully compliant with the award criteria by documenting user autonomy features, providing transparency in energy and resource consumption, and supporting the potential extension of the hardware operating life of devices.

Figure 18: Icon for KDE's popular application Okular.

Okular lets you check digital signatures and sign documents yourself, as well as include annotated text and comments directly embedded into the document. Okular works on Linux, Windows, Android, and Plasma Mobile, and it is available to download for all GNU/Linux distributions, as a standalone package from Flathub and the Snap Store, through the KDE F-Droid release repository for Android, as well as from the Microsoft Store. The source code is also readily available at Okular's GitLab repository for all to use, study, share, improve, and most of all, enjoy.

KDE and the Free Software community would like to send a heartfelt thank you to the Okular developers for making environmentally-friendly software for all of us! In the next section, we will look at the steps you need to complete to join us in having your software project recognized for its sustainable software design.

Benefits Of Blue Angel

Steffi Lemke, Federal Minister for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection (German: *Bundesministerium für Umwelt, Naturschutz, nukleare Sicherheit und Verbraucher-schutz*, or BMUV), has said about the reputation of the Blue Angel:

> An increasing number of people focus on durability and environmental friendliness when purchasing products. This is precisely what the Blue Angel stands for. The ecolabel has been a guarantee of high standards for the protection for our environment and health for 40 years in an independent and credible way.

Indeed, in their 40th anniversary info-booklet "Blue Angel – 40 years. Good for me. Good for the environment", the German Environment Agency (UBA) explored the history, present, and future of the ecolabel. In the booklet, they identify some of the general criteria they consider when eco-certifying a product, such as:

- reduced emissions of harmful substances in the ground, air, water and indoors;
- sustainable production of resources;
- longevity, ability to repair and recycle the product; and
- efficient use, e.g. products which save energy.

As you reach the end of this section, we hope it is clear how eco-certification of desktop software promotes the above environmental benefits, among others.

Environmental labels can be an instrument to move markets in the direction of sustainable products. As the Blue Angel website states, "The aim of the environmental label is to provide private customers, large institutional consumers and public institutions with reliable guidance for environmentally conscious purchasing."

So what does the market say?

A survey from the info-booklet found that 92% of Germans recognize the ecolabel, and for 37% the label influences their purchasing choices. The ecolabel is recognizable outside of Germany, too! In conversations with representatives at the German Environment Agency, we were informed that up to 15% of Blue Angel recipients are outside of Germany. One reason for this is that unlike some other ecolabels, the Blue Angel puts no requirements on where a product can be marketed. Moreover, the Blue Angel seal is considered a mark of high quality internationally, and the award criteria are viewed as an indicator of direction of the EU market—and are even used as a guideline for optimizing products.

Receiving the Blue Angel seal can raise your product's profile not only among individuals, but also large organizations. Green Public Procurement (GPP) initiatives, which "seek to promote the public procurement of goods, services, and works with a reduced environmental impact throughout their life-cycle" (European Commission), influence purchasing choices both in the public and private sector. Eco-certifying your software product with the Blue Angel demonstrates a commitment to long-term digital sustainability, and it gives your product visibility both in Germany and abroad.

A Note On Sources

Some material in this section is based directly on the following three texts:

- The Wikipedia article "Blue Angel (certification)", which is released under the Creative Commons Attribution-ShareAlike License 3.0.
- The Wikipedia article "Software bloat", which is released under the Creative Commons Attribution-ShareAlike License 3.0.

- The KDE Eco blog post “First Ever Eco-Certified Computer Program: KDE’s Popular PDF Reader Okular”, which is released under the Creative Commons Attribution-ShareAlike 4.0 International License.

PART III: FULFILLING THE BLUE ANGEL AWARD CRITERIA

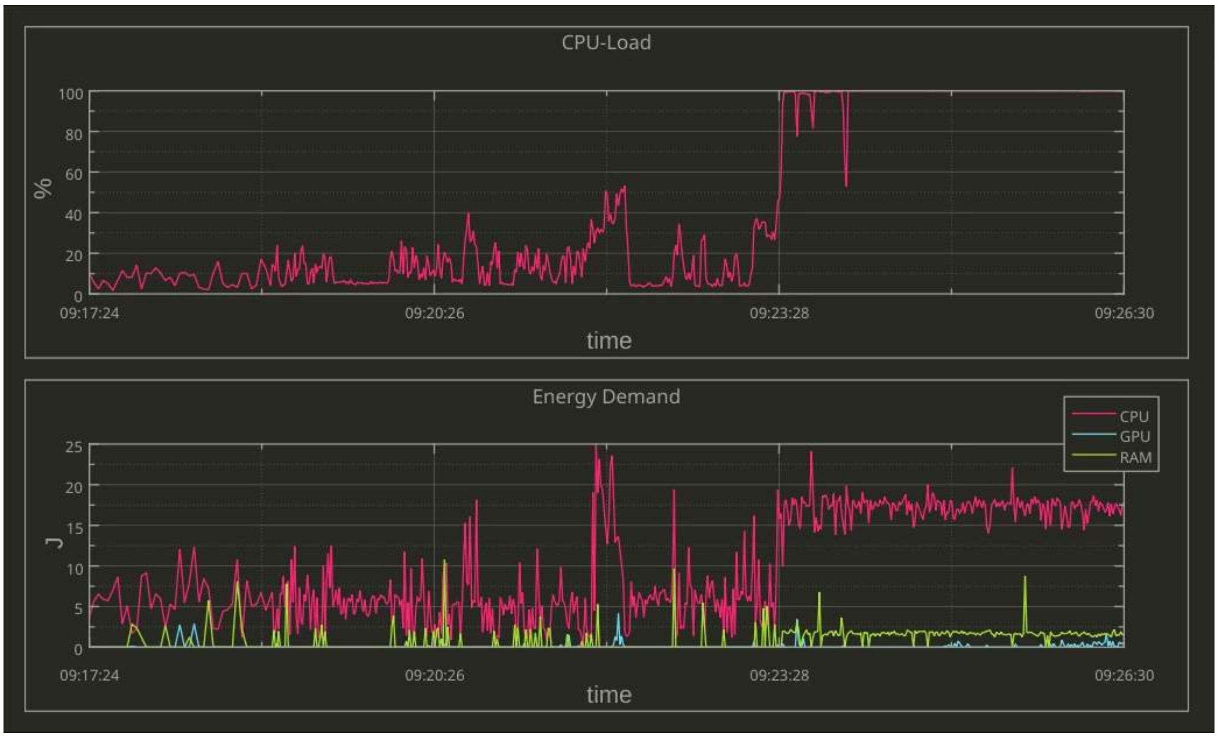


Figure 19: Monitoring energy and hardware consumption in real time with KDE’s LabPlot. (Image from Alexander Semke published under a CC-BY-NC-ND-4.0 license.)

The three main categories of the Blue Angel award criteria for desktop software are:

- (A) Resource & Energy Efficiency
- (B) Potential Hardware Operating Life
- (C) User Autonomy

In this section we’ll cover the three categories in more depth, providing a hands-on guide to fulfilling each set of criteria. There are numerous benefits of going through the certification process. By making the energy consumption of your software transparent and complying with the hardware operating life and user autonomy criteria, you get the benefits of:

- **Eco-Certification**: Apply for the Blue Angel ecolabel to demonstrate to users, companies, and governmental organizations that your software is designed sustainably.
- **Data-Driven Development**: Locate inefficiencies in terms of energy and hardware consumption, and make data-driven decisions for your software development.
- **Sustainable Design**: Take user autonomy criteria into consideration when planning your software design.
- **End-User Information**: Highlight to your users the ways your software is already sustainably designed by using the Blue Angel criteria as a benchmark.

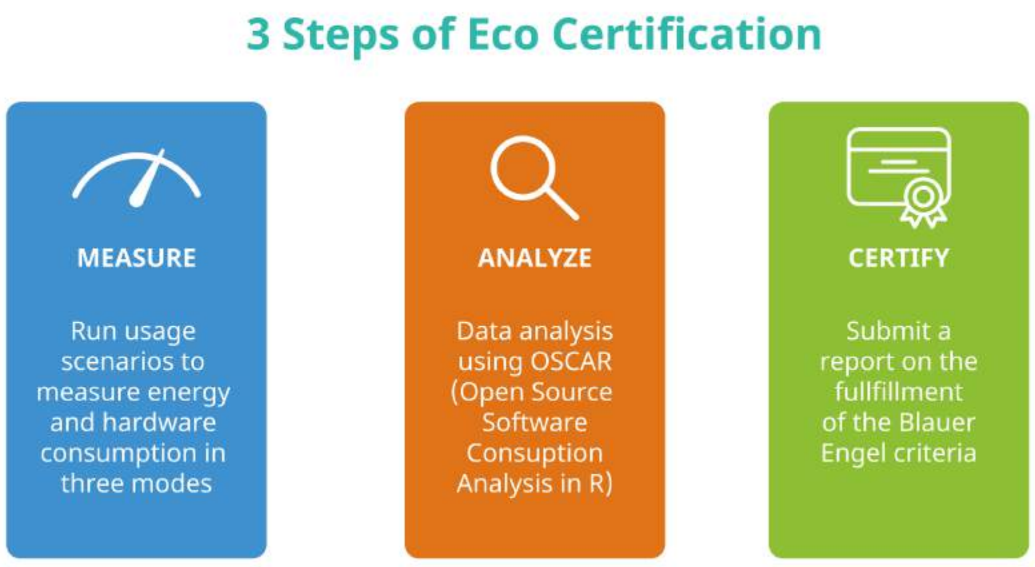

3 Steps of Eco Certification
MEASURE
Run usage scenarios to measure energy and hardware consumption in three modes
ANALYZE
Data analysis using OSCAR (Open Source Software Consuption Analysis in R)
CERTIFY
Submit a report on the fullfillment of the Blauer Engel criteria

Figure 20: The three steps to eco-certification: Measure, Analyze, Certify. (Image from KDE published under a CC BY-SA 4.0 International license. Certificate icon by Ongycon licensed under a CC-BY license.)

(A) How To Measure Your Software

The laboratory setup consists of a power meter, a computer to aggregate and evaluate the power meter output, and a desktop computer for the system under test where user behavior is emulated. The setup described here follows the specifications from the Blue Angel Basic Award Criteria for Resource and Energy-Efficient Software Products.

Terminology comes in part from Kern et al. (2018): "*Sustainable software products — Towards assessment criteria for resource and energy efficiency*".

See also the following resources from the Umwelt Campus Birkenfeld:

- Seiwert & Zaczyk (2021): "*Projektbericht: Ressourceneffiziente Softwaresysteme am Beispiel von KDE-Software*" (German only)
- Mai, Franziska (2021): *Vergleichende Analyse und Bewertung von Betriebssystemen hinsichtlich ihrer Energieeffizienz* (German only)
- OSCAR Manual (German only)

Overview Of Laboratory Setup

The laboratory setup requires 1 power meter and at least 2 computers:

- *Power Meter*

  One of the devices recommended by the Blue Angel is the Gude Expert Power Control 1202 Series (manual). It provides outlets for powering the computer and measures the current during operation. The device can be controlled and read via cabled Ethernet. There is a web-based user interface, a REST API, and the device supports various protocols such as SNMP or syslog.

- *Computer 1: Data Aggregator & Evaluator*

  The computer will be used for collecting and evaluating results from the power meter.

  A Python script to read out the data from the Gude Expert Power Control 1202 Series is available at the FEEP repository.

  It is recommended to monitor progress live with the second computer in order to ensure everything is proceeding smoothly. This can be done with KDE's Labplot, for instance; read more here.

  Other power meters may require non-Free software, e.g., Janitza's GridVis Power Grid Monitoring Software.

Figure 21: Screenshot of the Gude Power Meter.

- *Computer 2: System Under Test*

  The reference system is the hardware used to measure the energy consumption of the "system under test", or SUT. The SUT includes the operating system and software installed for (i) testing the software product, (ii) emulating the standard usage scenario[2] and (iii) collecting the hardware performance results.

A few notes:

For GNU/Linux systems, the Blue Angel criteria (Section 1.1) recommend Fujitsu computers as the reference system.

For emulating activity in the standard usage scenario, Free Software task automation tools such as `xdotool`, `KDE Eco Tester` (in progress), or `Actiona` (GPLv3) can be used.

For collecting hardware performance data (e.g., processor and RAM utilization, hard disk activity, network traffic), the Free Software tool `Collectl` (GPLv2/Artistic License) can be used.

It's also possible to repurpose cheap switchable power plugs as measurement devices; see Section "Alternative: Gosund SP111 Setup" for set up instructions.

[2] It is possible to have a setup using 3 computers, with the Standard Usage Scenario emulation generated on a computer independent of the SUT; see Kern et al. (2018). Details of a similar setup for an external workload generator can be found at the FEEP repository.

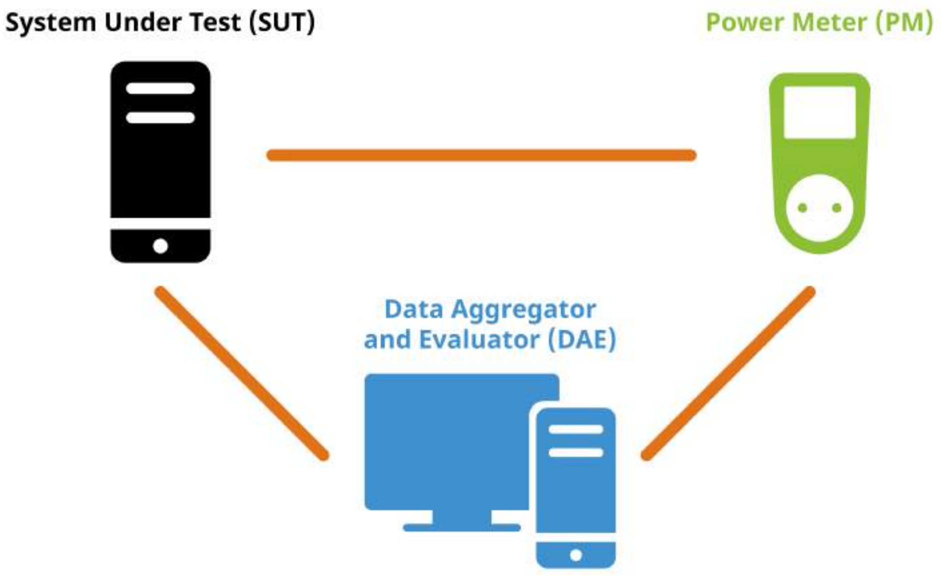


Figure 22: Overview of laboratory setup, including the System Under Test, the Power Meter, and the Data Aggregator and Evaluator. (Image from KDE published under a CC-BY-SA-4.0 license. Design by Lana Lutz.)

System Under Test (SUT) The Fujitsu Esprimo P920 Desktop-PC proGreen selection (Intel Core i5-4570 3,6GHz, 4GB RAM, 500GB HDD) is one of the recommended reference systems. See the award criteria for other recommended Fujitsu systems.

Set up the System Under Test (SUT) on the reference system. The SUT must reduce unrelated energy consumption and have a standardized configuration. This includes the following:

- Overwriting the entire hard drive of the machine with a standardized OS.
- Deactivating all possible background processes (automatic updates, backups, indexing, etc.).
- Installing the necessary software (i.e., the application to be measured as well as the emulation and data collection software).

Preparing The Standard Usage Scenario (SUS) Preparing the SUS requires the following:

- Identifying tasks users typically carry out when using the application under consideration.
- Identifying functionalities which require high energy demand or high resource utilization.

- Based on the above, scheduling a flow chart of individual actions and emulating these actions with a task automation tool.
- Note that when running the SUS, the cache should be cleared between runs and any new files deleted before starting the next measurement.

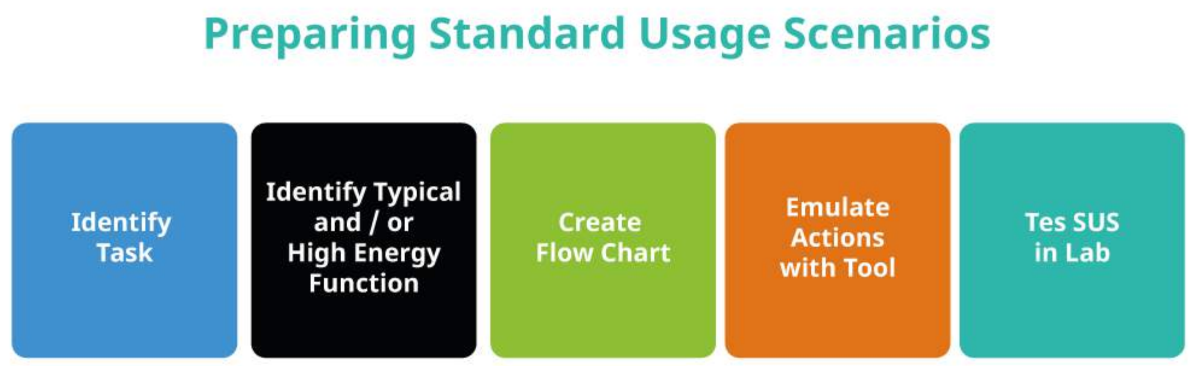

Preparing Standard Usage Scenarios
Identify Task
Identify Typical and / or High Energy Function
Create Flow Chart
Emulate Actions with Tool
Tes SUS in Lab

Figure 23: Steps for preparing Standard Usage Scenario (SUS) scripts to measure the energy consumption of software. (Image from KDE published under a CC-BY-SA-4.0 license. Design by Lana Lutz.)

An example of tasks in the standard usage scenario for KMail includes searching for an email, writing a reply or forwarding the email, saving an attachment, deleting a folder in the mail client, etc. See the Actiona scripts used to test KMail, Krita, and Okular for examples.

Note: If the task automation tool uses pixel coordinates to store the position of the automated clicks (e.g., `Actiona`) and, moreover, the screen resolution of the computer used in preparation differs from that of the laboratory computer, all pixel coordinates will eventually have to be set anew for the laboratory environment.

Emulation Tools For Usage Scenarios An automation tool is required to run the usage scenarios and does not need human intervention, so it can be run repeatedly in a well-defined way to provide accurate measurements.

Beyond `xdotool`, `KDE Eco Tester` (in progress), or `Actiona`, there are other candidates for tools which might meet the requirements. See a list from KDE Contributor David Hurka in the presentation “Visual Workflow Automation Tools”.

Most of those tools use X11-specific features, and thus do not work on Wayland systems. There are a few possible approaches here:

- Selenium Webdriver using AT-SPI (in testing durinmg Season of KDE 2023)
- The XDG RemoteDesktop portal
- Various Wayland protocols (support varies between compositors):
  - https://github.com/swaywm/wlr-protocols/blob/master/unstable/wlr-virtual-pointer-unstable-v1.xml
  - https://api.kde.org/frameworks/kwayland/html/classKWayland_1_1Client_1_1FakeInput.html
- libinput user devices

Measuring Software

The measurement process is defined in Appendix A of the Basic Award Criteria. It requires recording and logging energy data and performance indicators with a granularity of 1-second so that they can be processed and average values can be calculated.

Some general comments:

- Times between the power meter and Computer 1 (Data Aggregator & Evaluator) must be synchronized.
- On Computer 1, confirm that the desired power outlet is read out (e.g., via the live graph created when using `LabPlot`).
- When using `Collectl` to collect performance load, ensure it is running in the console of the SUT; also, check that the required CSV file is also correctly generated.
- Since each run of the usage scenarios results in changes to the standard operating system, clearing the cache between runs is recommended.
- All runs (Baseline, Idle Mode, Standard Usage Scenario) must be for the same length of time, based on the time needed to run the usage scenario script.

During the energy measurement, you also need to record a set of performance indicators: processor utilisation, RAM utilisation, hard disk activity and network traffic. Tool candidate:

- `Collectl`

The following command is used for obtaining hardware performance data with `Collectl` (Mai 2021: p. 15):

```
$ collectl -s cdmn -i1 -P --sep 59 -f ~/performanceMeasures.csv
```

The specified options are:

- `-s cdmn`

  collect CDU, Disk, memory, and network data
- `-i1`

  sampling interval of 1 second
- `-P`

  output in plot format (separated data which consists of a header with one line per sampling interval)
- `--sep 59`

  semicolon separator for -P option
- `-f </PATH/TO/FILE>.csv`

  save file at specified path

Measuring Baseline, Idle Mode, And Standard Usage Scenarios

- Baseline Scenario: *Operating System* (OS)

  To establish the baseline energy consumption and hardware performance data of the system under test, a scenario is measured in which the OS is running but no actions are taken.
- Idle Mode Scenario: *OS + Application While Idle*

  To establish the energy consumption and hardware performance data of the application while idle, a scenario is measured in which the application under consideration is opened but no action is taken.

Important: the baseline and idle mode must be run for the same time needed to carry out the standard usage scenario. Since the power consumption for the baseline and idle scenario is relatively uniform, 10 repetitions for each is considered sufficient to obtain a representative sample (Seiwert & Zaczyk 2021).

- Standard Usage Scenario: *OS + Application In Use*

  To measure the energy consumption and hardware performance data of the application in use, the standard usage scenario should be run; see SUS preparation notes above. The measurement of the standard usage scenario should be repeated 30 times, which will take several hours to complete. The higher number of repetitions is necessary to obtain a representative sample as the energy consumption and performance data may vary across measurements (Seiwert and Zaczyk 2021).

Monitoring Output With Labplot You can use KDE's LabPlot to monitor the output live as data is coming in. To do this:

- Redirect the power meter output to a CSV file.
- In LabPlot, import the CSV file by selecting `File > Add New > Live Data Source ...`.
- Under "Filter", select the Custom option. Under "Data Format", define the separator value used (e.g., comma, semi-colon, space).
- You can check that the output is correct under the "Preview" tab.
- If everything looks good, click OK.
- Finally, right-click on the data frame window and selecting `Plot Data > xy-Curve`.

Analysis Of The Results With OSCAR

The Umwelt Campus Birkenfeld provides a useful tool for generating reports from measurement data called OSCAR (Open Source Software Consumption Analysis):

- Source Code
- Running instance

At the project website, you can also find the Oscar Manual with detailed instructions, including screenshots on how to use OSCAR.

CSV Files Analysis with OSCAR requires uploading the following files to the OSCAR website: (i) a log file of actions taken, (ii) the energy consumption data, and (iii) the hardware performance data. All files are CSV files, examples of which are provided below. Some preprocessing of the raw data may be necessary (e.g., performance data measured by `Collectl`; see below).

Important: OSCAR is very particular about data frames, including column names and cell values. The tables here provide examples which are confirmed to work. If you are having issues generating a report from your CSV files, make sure CSV files are as similar as possible to those shown here.

To test OSCAR yourself, you can download the data for Okular for the baseline and SUS measurements in this ZIP file. The data has been used to generate a report successfully using OSCAR v0.190404. The report can be downloaded at the FEEP repository.

- Log File Of Actions

The log file of actions should have the following format. Note the columns are separated by a semicolon. Also, columns have no names (i.e., there is no header in the CSV file). Note that the start and end of each iteration must be labelled with 'startTestrun' and 'stopTestrun' in the second column, whereas the actions can be listed with any name.

YYYY-MM-DD HH:MM:SS ;	startTestrun ;	
YYYY-MM-DD HH:MM:SS ;	;	action1
YYYY-MM-DD HH:MM:SS ;	;	action2
YYYY-MM-DD HH:MM:SS ;	;	action3
YYYY-MM-DD HH:MM:SS ;	stopTestrun ;	

An example log file of actions for measuring KDE's text editor (and more) `Kate`. The (i) date and time as well as (ii) start and stop times and (iii) actions are listed in three columns.

2022-05-21 18:54:36 ;	startTestrun ;	
2022-05-21 18:55:41 ;	;	go to line 100
2022-05-21 18:55:46 ;	;	toggle comment
2022-05-21 18:55:50 ;	;	find kconfig
2022-05-21 18:55:55 ;	;	move between searches 6 times
2022-05-21 18:56:05 ;	;	close find bar
2022-05-21 18:56:05 ;	;	standby 30 sec
2022-05-21 18:56:35 ;	;	go to line 200
2022-05-21 18:56:40 ;	;	select 10 lines
2022-05-21 18:56:43 ;	;	delete selected text
[. . .] ;	;	[. . .]
2022-05-21 18:59:13 ;	stopTestrun ;	

- Energy Consumption Data

The energy consumption data has the following format: the first column is the row number, the second column is the date and time in one-second increments, and the third column is the measurement output in Watts. Note that the following is confirmed to work with OSCAR: (i) the second and third column names as written below (i.e., “Zeit” and “Wert 1-avg[W]”), (ii) the date-time as a character string with the date and time separated by a comma, and (iii) no string delimiter used in the CSV file.

;	Zeit ;	Wert 1-avg[W]
1 ;	DD.MM.YY, HH:MM:SS ;	value1
2 ;	DD.MM.YY, HH:MM:SS ;	value2
3 ;	DD.MM.YY, HH:MM:SS ;	value3
4 ;	DD.MM.YY, HH:MM:SS ;	value4

When using the Gude Power Meter with the Python script available at the FEEP repository, the timestamp will be recorded in nanoseconds in Epoch time. For example, below is an example of the raw output for 7 rows from the Gude Power Meter output from the script. The first column shows the timestamp. The second column is the readout from the power meter in Watts.

1661611923019071 ;	43
1661611923142924 ;	43
1661611924293989 ;	29
1661611924417017 ;	28
1661611924744885 ;	28
1661611924869051 ;	28
1661611924992392 ;	28

The raw data can be preprocessed in R. Nanoseconds in Epoch time are converted to date-time with the command `as.POSIXct(<NANOSECONDS>/1000000, origin = '1970-01-01', tz = 'Europe/Berlin')`. For example, the nanoseconds in row 1 from the raw output is, after conversion, “2022-08-27 16:52:03 CEST”. For OSCAR, this date-time can then be converted to a character string with the date as DD.MM.YY followed by a comma. All of this can be achieved with one command (this operation can be vectorized over the entire column in the data frame):

```
stringr::str_replace(as.character(as.POSIXct(1661611923019071/1000000,
 origin = '1970-01-01', tz = 'Europe/Berlin')), '2022-08-27', '
27.08.22,')
```

The output in Watts should be averaged per second. The same data above is shown below after processing with R; note the 7 values above are averaged per second, resulting in two rows.

To save the CSV file with a semi-colon separator, the first column with row names starting at the number 1, and no string delimiter, use the following R command: `write.csv2(<DATAFRAME>, file = <PATH/TO/FILE.csv>, row.names = TRUE, quote = FALSE)`.

;	Zeit ;	Wert 1-avg[W]
1 ;	27.08.22, 16:52:03 ;	43.00000
2 ;	27.08.22, 16:52:04 ;	28.20000

- Performance Data (Raw)

When using `Collectl` for collecting hardware performance data, it is necessary to do the following before uploading the data to OSCAR.[3]

- Remove all information above the header row.
- Remove all # characters from the file.
- In the first column, no separator value should come between the data-time (otherwise, the date and time will be interpreted as two separate columns).
- The date should also have a character inserted between YYYYMMDD, e.g., MM.DD.YYYY as above. Whatever character is used must be specified in OSCAR.
- Column names can be anything you want as they will be specified within OSCAR.
- The file must be saved in CSV format.

Moreover, the hardware performance output from `Collectl` includes many columns that are not necessary for analyzing the data with OSCAR. The only measurements that need to be specified are the following columns: [CPU]Totl = Processor; [MEM]Used = Main memory - used kilobytes; [NET]RxKBTot = Network - Kilobytes received/s; [NET]TxKBTot = Network - Kilobytes transmitted/s; [DSK]ReadKBTot = Disk - Kilobytes read/s; and [DSK]WriteKBTot = Disk - kilobytes written/s. In the OSCAR interface, you can select NA for unused columns, e.g., "Auslastung Auslagerungsdatei".

Below is an example of the preprocessed results from `Collectl` measuring the performance data for Kate. The timestamp again increases in one-second increments.

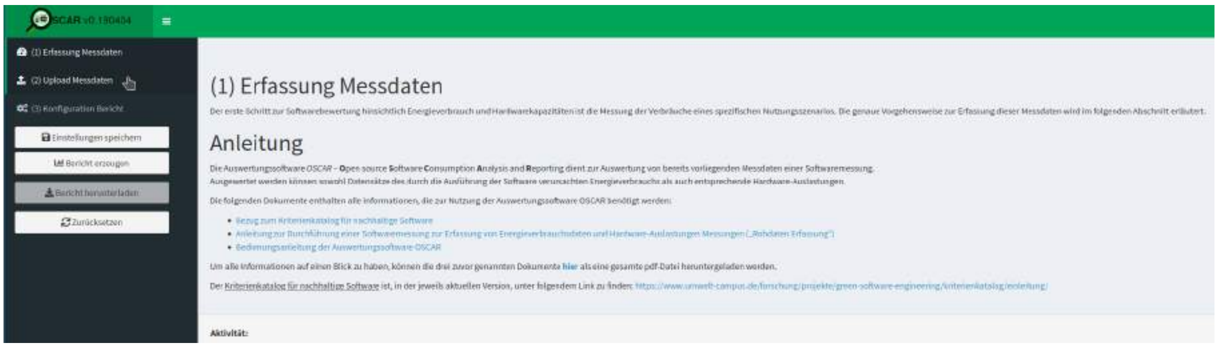
[3] See Seiwert & Zaczyk 2021: p. 13 for details; see also Appendix A 2 on p. 46 for a Python script to automate some of these tasks.

Date-Time ;	cpu ;	mem ;	net_rec ;	net_trn ;	dsc_rd ;	dsc_wr
27.08.2022 16:47:10 ;	1 ;	7131968 ;	0 ;	0 ;	0 ;	0
27.08.2022 16:47:11 ;	4 ;	7131968 ;	0 ;	0 ;	0 ;	0
27.08.2022 16:47:12 ;	1 ;	7131968 ;	0 ;	0 ;	0 ;	0
27.08.2022 16:47:13 ;	1 ;	7131968 ;	0 ;	0 ;	0 ;	120
27.08.2022 16:47:14 ;	1 ;	7131968 ;	0 ;	0 ;	0 ;	0
27.08.2022 16:47:15 ;	1 ;	7131968 ;	0 ;	0 ;	0 ;	56
27.08.2022 16:47:16 ;	1 ;	7131968 ;	0 ;	0 ;	0 ;	48
27.08.2022 16:47:17 ;	1 ;	7131968 ;	0 ;	0 ;	0 ;	0
27.08.2022 16:47:18 ;	1 ;	7131968 ;	0 ;	0 ;	0 ;	0
27.08.2022 16:47:19 ;	4 ;	7131968 ;	0 ;	0 ;	0 ;	132

Uploading Data Once the above CSV files are ready, you can run the analysis using OSCAR. `OSCAR` will generate a summary report you can use for eco-certification or for your own purposes. In the `OSCAR` interface, note the following:

- The interface language is currently German; see below for translations.
- The duration of the measurements in seconds must be specified.
- A semicolon separator is used.
- The correct formatting of the time stamp must be specified for each of the uploaded files, e.g.,`%Y-%m-%d %H:%M:%OS`.

Step 1: Obtain Measurement Data The landing page of the website (see below) states that the first step is obtaining measurement data (German: *Erfassung Messdaten*). If you are at this point in the process, you should have already measured your software and prepared the CSV files.



(1) Erfassung Messdaten

Der erste Schritt zur Softwarebewertung hinsichtlich Energieverbrauch und Hardwarekapazitäten ist die Messung der Verbräuche eines spezifischen Nutzungsszenarios. Die genaue Vorgehensweise zur Erfassung dieser Messdaten wird im folgenden Abschnitt erläutert.

Anleitung

Die Auswertungssoftware *OSCAR* – **O**pen source **S**oftware **C**onsumption **A**nalysis and **R**eporting dient zur Auswertung von bereits vorliegenden Messdaten einer Softwaremessung.
Ausgewertet werden können sowohl Datensätze des durch die Ausführung der Software verursachten Energieverbrauchs als auch entsprechende Hardware-Auslastungen.

Die folgenden Dokumente enthalten alle Informationen, die zur Nutzung der Auswertungssoftware OSCAR benötigt werden:

- Bezug zum Kriterienkatalog für nachhaltige Software
- Anleitung zur Durchführung einer Softwaremessung zur Erfassung von Energieverbrauchsdaten und Hardware-Auslastungen Messungen („Rohdaten Erfassung")
- Bedienungsanleitung der Auswertungssoftware OSCAR

Um alle Informationen auf einen Blick zu haben, können die drei zuvor genannten Dokumente **hier** als eine gesamte pdf-Datei heruntergeladen werden.

Der Kriterienkatalog für nachhaltige Software ist, in der jeweils aktuellen Version, unter folgendem Link zu finden: https://www.umwelt-campus.de/forschung/projekte/green-software-engineering/kriterienkatalog/einleitung/

Aktivität:

Figure 24: Oscar screenshot with the first step of obtaining measurement data (German: "Erfassung Messdaten").

Step 2: Upload Measurement Data Once you have the CSV files for the baseline, idle mode, and usage scenario measurements, click on *(2) Upload Messdaten*, or "upload measurement data".

Under *Messungen*, upload the idle mode or standard usage scenario measurements.

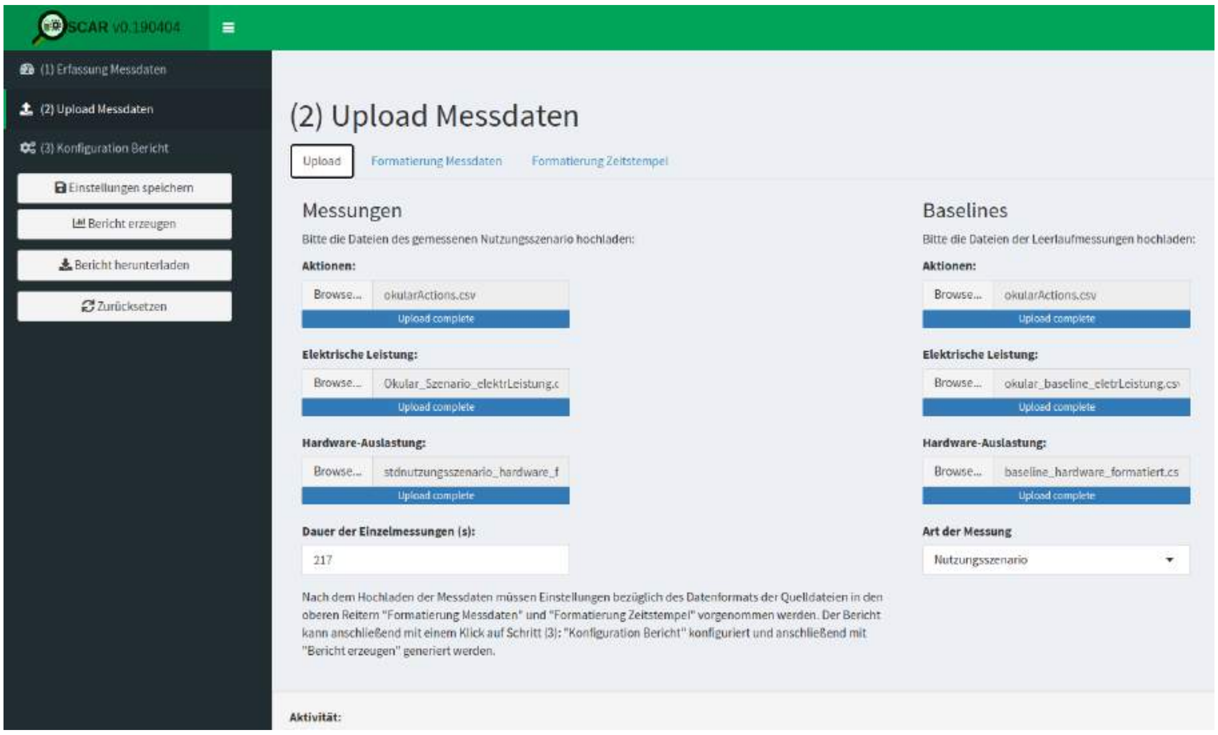
Under *Baselines* upload the baseline measurements.

You should also upload other relevant data, including the log file of actions taken (German: *Aktionen*), energy consumption (German: *Elektrische Leistung*), and hardware performance data (German: *Hardware-Auslastung*).

For *Art der Messung* ('Type of Measurement') in the lower right of the following screenshot, select *Leerlauf* ('Idle Mode') or *Nutzungsszenario* ('Usage Scenario') depending on which report you wish to generate (here, *Nutzungsszenario* was selected).

Note that the baseline measurements are always uploaded along with the idle mode or standard usage scenario measurements. It is also important to indicate the duration of the measurement scenarios in seconds (German: *Dauer der Einzelmessungen (s)*).

See below for what a completed upload for the *Nutzungsszenario* with a duration of 217 seconds looks like:



(2) Upload Messdaten

Messungen

Bitte die Dateien des gemessenen Nutzungsszenario hochladen:

Aktionen: Browse... okularActions.csv — Upload complete

Elektrische Leistung: Browse... Okular_Szenario_elektrLeistung.c — Upload complete

Hardware-Auslastung: Browse... stdnutzungsszenario_hardware_f — Upload complete

Dauer der Einzelmessungen (s): 217

Baselines

Bitte die Dateien der Leerlaufmessungen hochladen:

Aktionen: Browse... okularActions.csv — Upload complete

Elektrische Leistung: Browse... okular_baseline_eletrLeistung.cs — Upload complete

Hardware-Auslastung: Browse... baseline_hardware_formatiert.cs — Upload complete

Art der Messung: Nutzungsszenario

Nach dem Hochladen der Messdaten müssen Einstellungen bezüglich des Datenformats der Quelldateien in den oberen Reitern "Formatierung Messdaten" und "Formatierung Zeitstempel" vorgenommen werden. Der Bericht kann anschließend mit einem Klick auf Schritt (3): "Konfiguration Bericht" konfiguriert und anschließend mit "Bericht erzeugen" generiert werden.

Aktivität:

Figure 25: Oscar screenshot with the second step of uploading measurement data (German: "Upload Messdaten").

Timestamps

Once the data has been uploaded, you will need to tell OSCAR how to read the data. Let's start with the timestamp, since this is one aspect of the process which can cause problems if not done correctly. As an example, consider the Okular data:

- For the log file of actions (German: *Aktionen*), the datetime is encoded as *YYYY-MM-DD HH:MM:SS*, e.g., "2022-10-04 12:32:43.656" in "okularActions.csv" for baseline measurements. Note the hyphen in the date; OSCAR will take care of the fractional seconds.

  In OSCAR, this is specified as "%Y-%m-%d %H:%M:%OS" (see screenshot below).

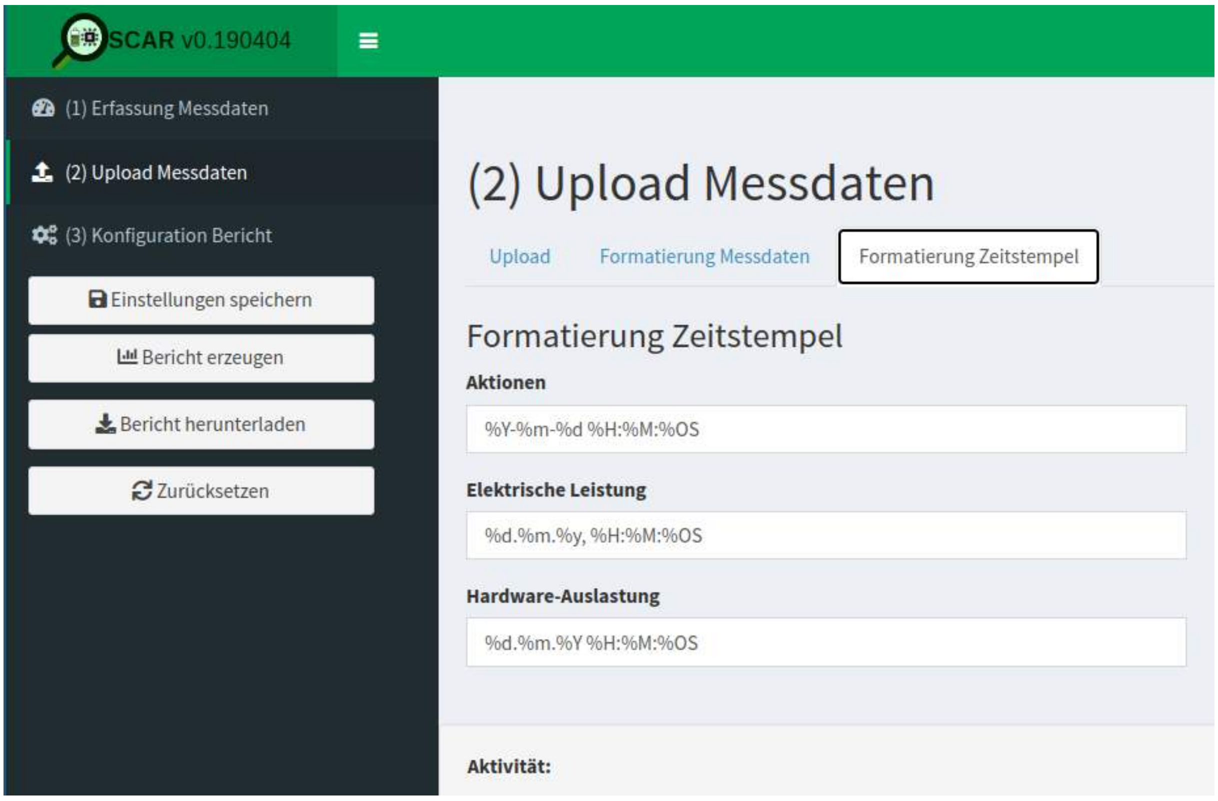
- For the energy consumption data (German: *Elektrische Leistung*), the datetime is encoded as *DD.MM.YY, HH:MM:SS*, e.g., "04.10.22, 12:32:43" in "okular_baseline_eletrLeistung.csv" for the baseline measurements. Note the period in the date and the comma seperating date from time, as well as only having two digits for the year.

  In OSCAR this is specified as "%d.%m.%y, %H:%M:%OS" (see screenshot below), in which the lowercase "%y" indicates a year with two digits.

- For the hardware performance data (German: *Hardware-Auslastung*), the datetime is encoded as

DD.MM.YYYY HH:MM:SS, e.g., “04.10.2022 12:31:43” in “baseline_hardware_formatiert.csv” for the baseline measurements. Note the period in the date and the four-digit year.

In OSCAR, this is specified as: “%d.%m.%Y %H:%M:%OS” (see screenshot below), in which the uppercase “%Y” indicates a year with four digits.



(2) Upload Messdaten

Formatierung Zeitstempel

Aktionen

%Y-%m-%d %H:%M:%OS

Elektrische Leistung

%d.%m.%y, %H:%M:%OS

Hardware-Auslastung

%d.%m.%Y %H:%M:%OS

Aktivität:

Figure 26: Oscar screenshot for uploading the measurement data where one specifies the format of the timestamp (German: “Formatierung Zeitstempel”).

Format Of The Measurement Data

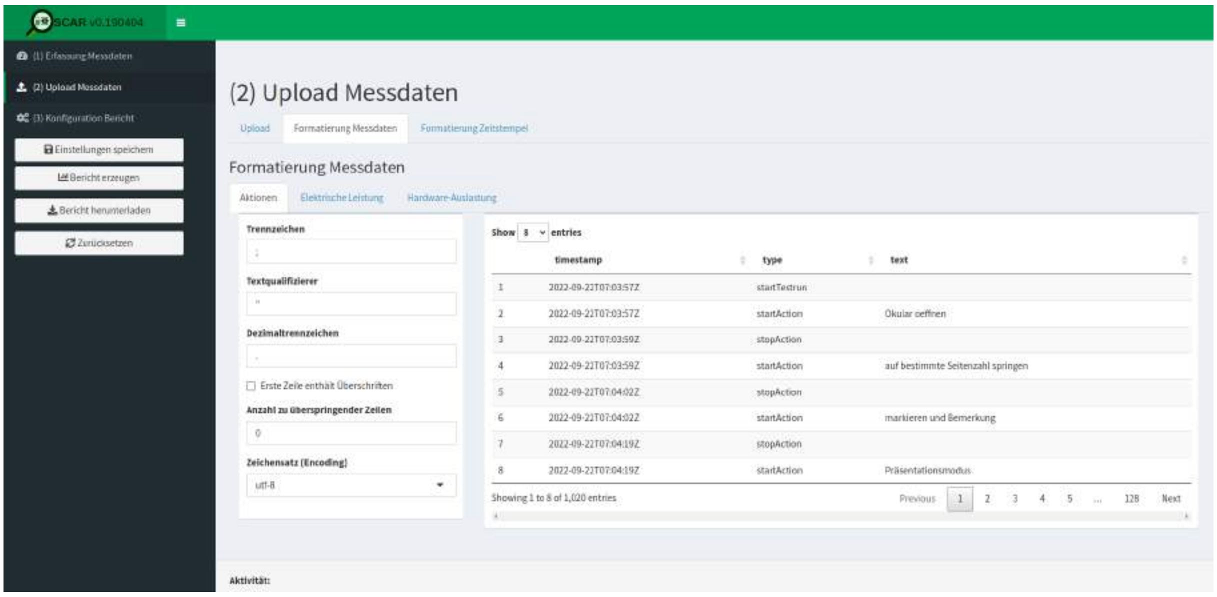
After the timestamps have been correctly specified, let’s explore the format of the measurement data (German: *Formatierung Messdaten*) in OSCAR.

First, take a look at the log file of actions (German: *Aktionen*). Here you need to indicate for the uploaded CSV file the separator (German: *Trennzeichen*), the string delimiter (German: *Textqualifizierer*), and the decimal separator (German: *Dezimaltrennzeichen*). For the Okular data, this is defined as a semi-colon separator, double quotation string delimiter, and a period or full-stop decimal separator (see following screenshot).

Additionally, you will need to indicate whether the first line contains headings (German: *Erste Zeile enthält Überschriften*); if necessary, the number of lines to skip (German: *Anzahl zu überspringender*

Zeilen); and the character encoding (German: *Zeichensatz (Encoding)*). For the Okular data, this is defined as follows in the following screenshot: "first line contains headings" is unchecked, 0 lines are skipped, and character encoding is utf-8.

Since everything is specified correctly, a preview of the spreadsheet is shown in OSCAR.



(2) Upload Messdaten

Formatierung Messdaten

Trennzeichen: ;

Textqualifizierer: "

Dezimaltrennzeichen: .

☐ Erste Zeile enthält Überschriften

Anzahl zu überspringender Zeilen: 0

Zeichensatz (Encoding): utf-8

Show 8 entries

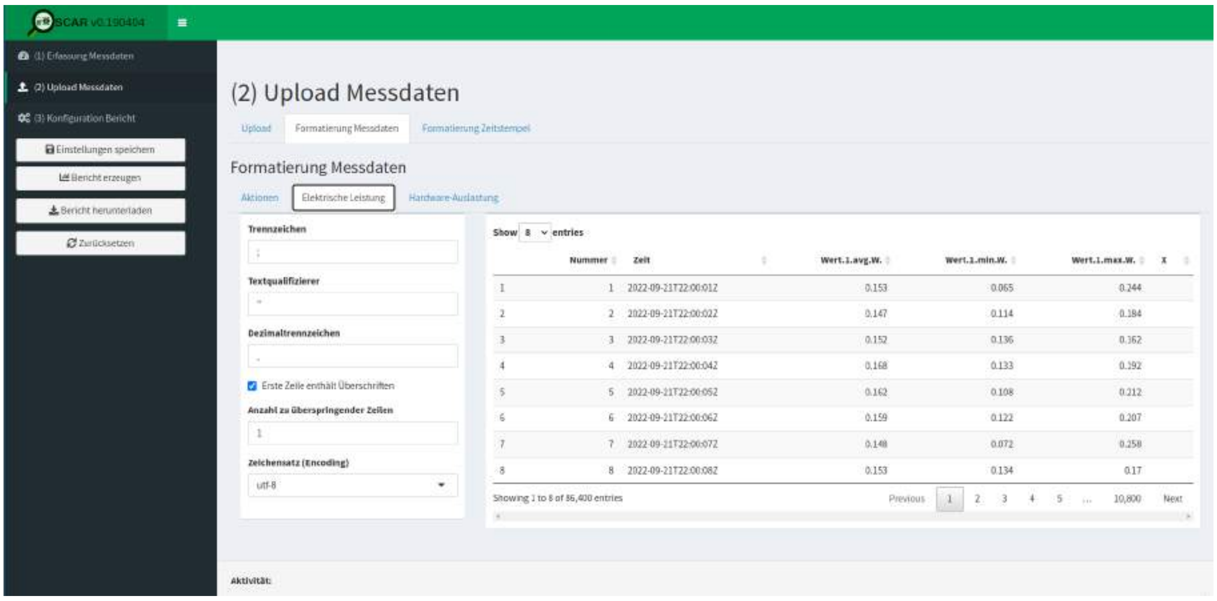
	timestamp	type	text
1	2022-09-22T07:03:57Z	startTestrun	
2	2022-09-22T07:03:57Z	startAction	Okular oeffnen
3	2022-09-22T07:03:59Z	stopAction	
4	2022-09-22T07:03:59Z	startAction	auf bestimmte Seitenzahl springen
5	2022-09-22T07:04:02Z	stopAction	
6	2022-09-22T07:04:02Z	startAction	markieren und Bemerkung
7	2022-09-22T07:04:19Z	stopAction	
8	2022-09-22T07:04:19Z	startAction	Präsentationsmodus

Showing 1 to 8 of 1,020 entries

Figure 27: Oscar screenshot for the log file of actions (German: "Aktionen") where one specifies the format of the measurement data (German: "Formatierung Messdaten").

For the energy consumption measurements (German: *Elektrische Leistung*), the required input is the same as for the log file of actions, seen in the following screenshot. For the Okular data, this is a semi-colon separator, double quotation string delimiter, and character encoding utf-8. However, now a comma is specified for the decimal separator and a checkmark indicates that the first line contains headings. Finally, that 1 line is skipped is specified.

Because we have defined everything correctly, a preview of the spreadsheet is shown in the OSCAR interface.



(2) Upload Messdaten

Formatierung Messdaten

Trennzeichen: ;

Textqualifizierer: "

Dezimaltrennzeichen: .

☑ Erste Zeile enthält Überschriften

Anzahl zu überspringender Zeilen: 1

Zeichensatz (Encoding): utf-8

Show 8 entries

	Nummer	Zeit	Wert.1.avg.W.	Wert.1.min.W.	Wert.1.max.W.	X
1	1	2022-09-21T22:00:01Z	0.153	0.065	0.244	
2	2	2022-09-21T22:00:02Z	0.147	0.114	0.184	
3	3	2022-09-21T22:00:03Z	0.152	0.136	0.362	
4	4	2022-09-21T22:00:04Z	0.168	0.133	0.192	
5	5	2022-09-21T22:00:05Z	0.162	0.108	0.212	
6	6	2022-09-21T22:00:06Z	0.159	0.122	0.207	
7	7	2022-09-21T22:00:07Z	0.148	0.072	0.258	
8	8	2022-09-21T22:00:08Z	0.153	0.134	0.17	

Showing 1 to 8 of 86,400 entries

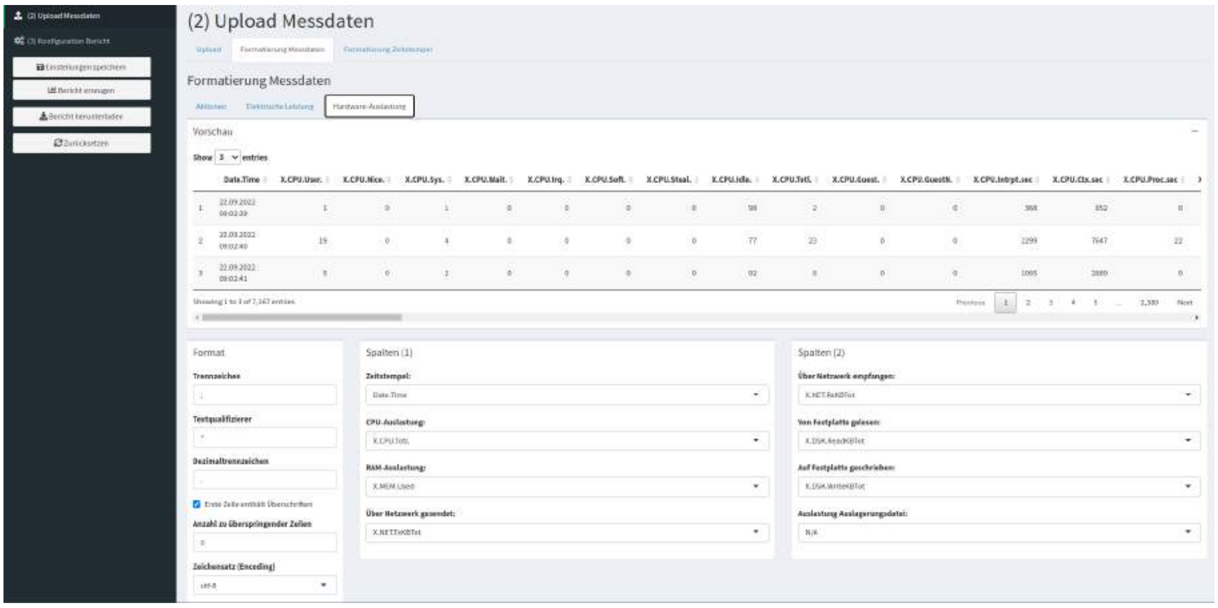
1 2 3 4 5 … 10,800

Aktivität:

Figure 28: Oscar screenshot for the energy consumption measurements (German: "Elektrische Leistung") where one specifies the format of the measurement data (German: "Formatierung Messdaten").

For the hardware performance data (German: *Hardware-Auslastung*), the required input is the same, but now with the additional requirement of specifying the columns. The input is a semi-colon separator, a double quotation string delimiter, a period (a.k.a. full stop) decimal separator, there is a checkmark that the first line contains headings, 0 lines are skipped, and the character encoding is utf-8. For the columns (German: *Spalten*) specification, the following are defined:

- *Zeitstempel*: Datetime (i.e., 'Date-Time')
- *CPU-Auslastung*: CPU (i.e., 'X.CPU.Totl')
- *RAM-Auslastung*: RAM (i.e., 'X.MEM.Used')
- *Über Netzwerk gesendet*: Network transmitted (i.e., 'X.NET.TxKBTot')
- *Über Netzwerk empfangen*: Network received (i.e., 'X.NET.RxKBTot')
- *Von Festplatte gelesen*: Disk read (i.e., 'X.DSK.ReadKBTot')
- *Auf Festplatte geschrieben*: Disk written (i.e., 'X.DSK.WriteKBTot')
- *Auslastung Auslagerungsdatei*: Swap (here, 'N/A')



(2) Upload Messdaten

Formatierung Messdaten

Vorschau

Show 3 entries

	Date.Time	X.CPU.User.	X.CPU.Nice.	X.CPU.Sys.	X.CPU.Wait.	X.CPU.Irq.	X.CPU.Soft.	X.CPU.Steal.	X.CPU.Idle.	X.CPU.Totl.	X.CPU.Guest.	X.CPU.GuestN.	X.CPU.Intrpt.sec	X.CPU.Ctx.sec	X.CPU.Proc.sec
1	22.09.2022 09:02:39	1	0	1	0	0	0	0	98	2	0	0	368	352	0
2	22.09.2022 09:02:40	19	0	4	0	0	0	0	77	23	0	0	2299	7647	22
3	22.09.2022 09:02:41	8	0	2	0	0	0	0	92	8	0	0	1065	2880	0

Showing 1 to 3 of 7,167 entries

Previous 1 2 3 4 5 … 2,389 Next

Format

Trennzeichen: ;

Textqualifizierer: "

Dezimaltrennzeichen: ,

☑ Erste Zeile enthält Überschriften

Anzahl zu überspringender Zeilen: 0

Zeichensatz (Encoding): utf-8

Spalten (1)

Zeitstempel: Date.Time

CPU-Auslastung: X.CPU.Totl.

RAM-Auslastung: X.MEM.Used

Über Netzwerk gesendet: X.NET.TxKBTot

Spalten (2)

Über Netzwerk empfangen: X.NET.RxKBTot

Von Festplatte gelesen: X.DSK.ReadKBTot

Auf Festplatte geschrieben: X.DSK.WriteKBTot

Auslastung Auslagerungsdatei: N/A

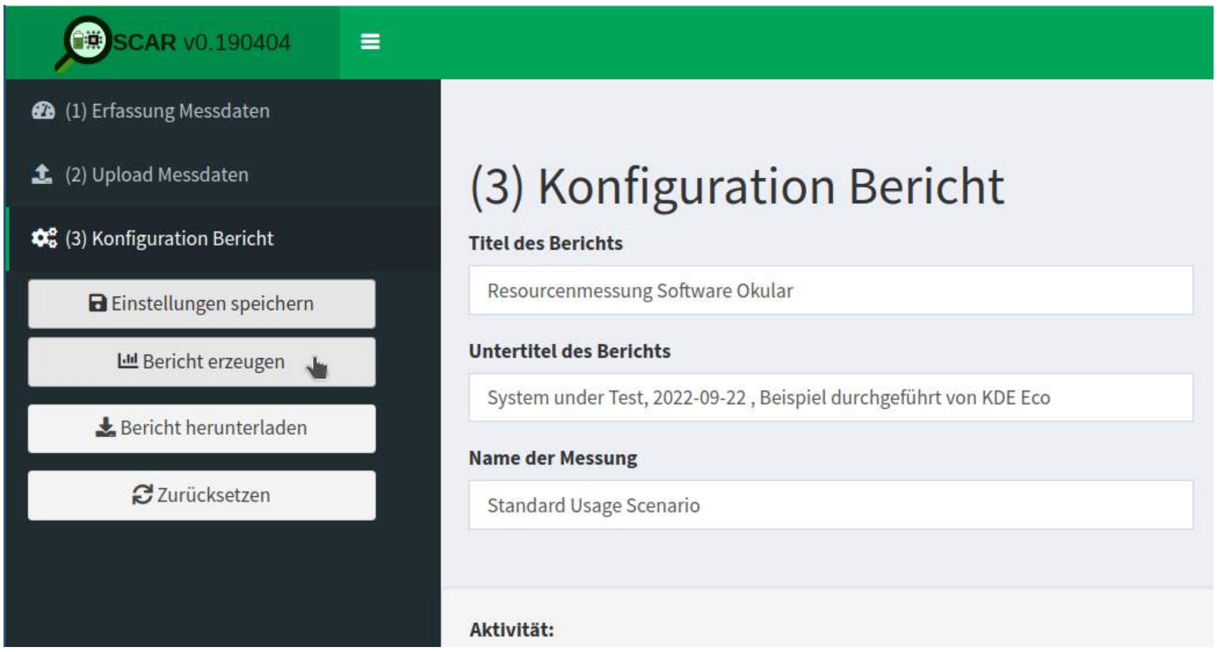
Figure 29: Oscar screenshot for the energy consumption measurements (German: "Hardware-Auslastung") where one specifies the format of the measurement data (German: "Formatierung Messdaten").

Translations Here is an overview of some of the terminology for OSCAR in German and their English translations:

- *Messungen*: Measurements (e.g., Idle Mode or SUS)
- *Aktionen*: Actions (i.e., log file of actions taken)
- *Elektrische Leistung*: 'Electrical power' (i.e., energy consumption measurements)
- *Hardware-Auslastung*: 'Hardware load' (i.e., hardware performance measurements)
- *Dauer der Einzelmessungen (s)*: 'Duration of the individual measurements (s)' (i.e., specify how long each iteration was in seconds)
- *Art der Messung*: 'Type of measurement'
  - *Leerlauf*: 'Idle' (i.e., Idle mode)
  - *Nutzungsszenario*: 'Usage scenario' (i.e., SUS)
- *Formatierung Messdaten*: 'Formatting measurement data'
- *Formatierung Zeitstempel*: 'Formatting timestamp'
- *Trennzeichen* 'Separator'

- *Textqualifizierer*: 'String delimiter'
- *Dezimaltrennzeichen*: 'Decimal separator'
- *Erste Zeile enthält Überschriften*: 'First line contains headings'
- *Anzahl zu überspringender Zeilen*: 'Number of lines to skip'
- *Zeichensatz (Encoding)*: 'Character set (encoding)'
- *Spalten*: Columns
  - *Zeitstempel*: 'Datetime'
  - *CPU-Auslastung*: 'CPU utilization'
  - *RAM-Auslastung*: 'RAM utilization'
  - *Über Netzwerk gesendet*: 'Sent via network'
  - *Über Netzwerk empfangen*: 'Received via network'
  - *Von Festplatte gelesen*: 'Read from disk'
  - *Auf Festplatte geschrieben*: 'Written to disk'
  - *Auslastung Auslagerungsdatei*: 'Swap file utlization'

Downloading Reports After completing the above, the report can be generated and downloaded. You will need to do this process twice for (i) the idle mode and (ii) standard usage scenario measurements, resulting in two documents.



(3) Konfiguration Bericht

Titel des Berichts: Resourcenmessung Software Okular

Untertitel des Berichts: System under Test, 2022-09-22 , Beispiel durchgeführt von KDE Eco

Name der Messung: Standard Usage Scenario

Aktivität:

Figure 30: Oscar screenshot for generating the report (German: "Bericht erzeugen").

For Blue Angel eco-certification, the two reports need to be submitted for evaluation. For examples of the above for Okular, see KDE's Blue Angel Applications repository:

- OSCAR Report For Idle Mode Measurement
- OSCAR Report For Standard Usage Scenario Measurement

Submitting Measurement Documentation For Blue Angel

For Blue Angel eco-certification, alongside the two reports from `OSCAR` it is necessary to complete several spreadsheets.

Information that needs to be included is as follows:

- Details about the software (name, version, etc) and measurement process (when and where measurements were made, etc.)
- Technical details about the power meter (instrument, sampling frequency, length of scenario, sample size)
- Technical details about the reference system (year, model, processor, cores, etc.)
- Software stack used for measurements (`xdotool`, `Collectl`, etc.)
- Minimum system requirements (processor architecture, local working memory, etc.)
- Energy consumption results found in the `OSCAR` reports or equivalent.
- Hardware utilization results, which includes the following (for IDLE use Idle Mode measurements, and for SUS use SUS measurements):
  - *Full Load*: "For processing power, the full load is 100%, for working memory the sum of the installed RAM capacities, for network bandwidth the maximum transmission speed, etc." (Blue Angel award criteria: p. 23).
  - *Base Load*: Average load for the reference system in Baseline measurements.
  - *Idle/SUS Load*: Average load for the reference system for IDLE/SUS measurements.

    From the above measurements, the following calculations are made for hardware utilization (for IDLE use Idle Mode measurements, and for SUS use SUS measurements):

    * *Net Load*: IDLE/SUS Load - Base Load
    * *Allocation Factor*: Net Load/(Full Load - Base Load)
    * *Effective Load*: Net Load + Allocation Factor * Base Load
    * *Hardware Utilization* (SUS only): Effective Load * Time (seconds)

For Blue Angel eco-certification, the above data needs to be added to 2 documents called “Annex 1” and “Annex 2”. For examples of the above for Okular, see KDE’s Blue Angel Applications repository:

- Annex 1
- Annex 2

Alternative: Gosund SP111 Setup

Want to get started with the process, but short on cash or gear? Want to give the process a try without setting up a dedicated lab? Try this hack converting an inexpensive power plug to a power meter, courtesy of Volker Krause, who also documented the process provided here in detail. You can read more at the following blog posts from Volker’s blog:

- “Cheap Electric Power Measurement”
- “KDE Eco Sprint July 2022”

Below is a guide to setting up a Gosund SP111 power plug already flashed with Tasmota firmware in 10 steps.

Although the data from this inexpensive power meter will likely not be accepted by the Blue Angel for eco-certification, it is nonetheless possible to obtain preliminary data using this tool.

- (0) *Prerequisite*

  It is necessary to have the power plug already flashed with a sufficiently new Tasmota version.

- (1) *Firmware Reset*

  If the device had previously been connected to another Wifi network, it might need a full reset before being able to connect to a new one.

  If the device did open a WiFi access point named “tasmota-XXXXX” this is not needed, continue directly to (2).

  Press the button for 40 seconds.

  The device will restart and you should be able to continue at (2).

- (2) *WiFi Setup*

  The device opens a WiFi access point named “tasmota-XXXXX”—connect to that.

  Open http://192.168.4.1 in a browser.

  The device will ask you for the WiFi name and password to connect to after entering those. The device will reconnect to that WiFi and disable its access point.

While doing that it should show you its new address in the browser—make a note of it.

In case that did not happen, check your WiFi router for the address of the device.

- (3) *Tasmota Setup*

Open the address from step (2) in a browser.

You should see the Tasmota web UI (a big "ON/OFF" text and a bunch of blue and one red button).

Click "Configuration".

Click "Configure Other".

Copy

```
{"NAME":"Gosund SP111 2","GPIO":
[56,0,57,0,132,134,0,0,131,17,0,21,0],"FLAG":0,
"BASE":18}
```

into the template input field.

Tick the "Activate" checkbox.

Click "Save".

The device will restart; connect to it again.

The UI should now also contain text fields showing electrical properties, and the "Toggle" button should now actually work.

- (4) *Calibration*

Open the address from step (2) in a browser.

Connect a purely resistive load with a known wattage, such as a conventional light bulb (not a LED or energy-saving bulb).

Switch on power by clicking "Toggle" if needed.

Verify that the "Power Factor" value is shown as 1 (or very close to 1); if it is lower, the current load is not suited for calibration.

Click "Console".

Enter the following commands one at a time and press enter:

```
AmpRes 3
VoltRes 3
EnergyRes 3
WattRes 3
FreqRes 3
SetOption21 1
```

```
    VoltageSet 230
```

Enter the command PowerSet XXX with XXX replaced by the wattage specified for the test load (e.g., "40" for a 40W light bulb).

Click "Main Menu".

The main page now should show correct power readings with several decimals precision.

- (5) *MQTT Broker Setup*

  At present, the only known way to achieve high-frequency automatic readouts is by polling over MQTT. This is not ideal and needs additional setup, unfortunately.

  If you happen to have a MQTT Broker around already, skip to step (6); otherwise, you need to set one up. The below scenario assumes Mosquitto is packaged for your GNU/Linux distribution (and therefore does not configure any security), so only do this in your own trusted network and switch it off when not needed.

  - install the `mosquitto` package
  - add a file `/etc/mosquitto/conf.d/listen.conf` with the following content:

    ```
    listener 1883
    allow_anonymous true
    ```

  - start Mosquitto using `systemctl start mosquitto.service`

- (6) *MQTT Tasmota Setup*

  Connect to the Tasmota device using a web browser, and open the MQTT configuration page via Configuration > Configure MQTT.

  Enter the IP address of the MQTT broker into the "Host" field.

  Note down the value shown right of the "Topic" label in parentheses (typically something like "tasmota_xxxxxx"). This will be needed later on to address the device via MQTT. You can also change the default value to something easier to remember, but this has to be unique if you have multiple devices.

  Click "Save".

  The device will restart and once it is back you should see output in its Console prefixed with "MQT".

- (7) *Verifying MQTT Communication*

  This assumes you have the Mosquitto client tools installed, which are usually available as distribution packages.

You need two terminals to verify that MQTT communication works as intended.

```
- In terminal 1, run `mosquitto_sub -t 'stat/<topic>/STATUS10'`
- In terminal 2, run `mosquitto_pub -t 'cmnd/<topic>/STATUS' -m '
  10'`
```

Replace <`topic`> with the value noted down in step (6).

Everytime you run the second command, you should see a set of values printed in the first terminal.

- (8) *Continuous Power Measurements*

See these scripts.

- (9) *Switching WiFi Networks*

For security reasons, once connected to a WiFi network, Tasmota will not let you get back to step (2) by default without hard resetting the device (40-second button press). However, a hard reset also removes all settings and the calibration. If you need to move to a different network, there are less drastic options available, but these changes can only be made inside the network you originally connected to:

Under Configuration > Configure WiFi, you can add details for a second WiFi access point. Those will be tried alternatingly with the first configuration by default. This does not compromise security, but requires you to know the details for the network you want to connect to.

You can configure Tasmota to open an access point as in step (2) by default for a minute or so after boot, and then try to connect to the known configurations. This makes booting slower in known networks, and opens the potential for hijacking the device, but it can be convenient when switching to unknown networks. This mode can be enabled in the Console by the command `WifiConfig 2`, and disabled by the command `WifiConfig 4`.

For Tasmota version 11 the 40-second button press reset can leave the device in a non-booting state, whereas resetting from the Console using `Reset 1` doesn't have that problem, but has to be done before disconnecting from the known WiFi as well.

- (10) *Recovering Non-Booting Devices*

First and foremost: **DO NOT CONNECT THE DEVICE TO MAIN POWER**! That would be life-threatening. The entire flashing process is solely powered from 3.3V supplied by the serial adapter. Do not do any of this without having read this getting started guide.

With Tasmota 11, you can end up in a non-booting state by merely resetting the device using the 40-second button press. This does not permanently damage the device, and it can be fixed with reflashing via a serial adapter.

The basic process is described in the above guide. The PCB layout of the Gosund SP 111 can be seen here.

In order for this to work, you need to connect GPIO0 (second pin on bottom left in the above image) to GND **before** powering up (i.e., before connecting with USB). The device LEDs (red and blue) are a useful indicator of whether you ended up in the right boot mode: the red LED should be on, and not flashing quickly, and the blue and red LED should not be on together. Once in that state, the connection can be removed (e.g., if you just hold a jumper cable to the pin) and it will remain in the right mode until a reboot.

Again: **DO NOT CONNECT THE DEVICE TO MAIN POWER** as this is life-threatening.

Other Initiatives Working On Sustainable Software Tooling

There are many initiatives working on tooling for measuring the energy consumption of software. We would like to mention four in particular who have been working collaborating with the KDE Eco initiative:

- The Green Software Engineering work group at the Environmental Campus Birkenfeld (German: *Umwelt Campus Birkenfeld*)

  Since 2008, the Green Software Engineering work group have been working on research projects with a focus on sustainable software. Their team developed tools such as OSCAR and have measured various KDE applications, including Okular.

- Green Coding Berlin

  Green Coding Berlin is focused on research into the energy consumption of software and its infrastructure, creating open source measurement tools, and building a community and ecosystem around green software.

- The SoftAWERE project from the Sustainable Digital Infrastructure Alliance

  The SoftAWERE steering group oversees and sets the direction for the development of tools and labels for energy-efficient software applications.

- Green Web Foundation

  The Green Web Foundation tracks and accelerates the transition to a fossil-free internet.

(B) Hardware Operating Life

The criteria in category (B) ensure that the software has low-enough performance requirements to run on older, less powerful hardware at least five years old. Compliance entails a declaration of backward compatibility, with details about the hardware on which the software runs and the required software stack.

Many FOSS applications run on hardware much older than 5 years. In fact, members of the KDE community have noted that KDE's desktop environment `Plasma` runs on hardware from even 2005!

This category is relatively easily to fulfill for the Blue Angel application. To demonstrate compliance for Blue Angel certification, one needs to confirm the software runs on older hardware and document the following information:

- *Reference System Year* — e.g., 2015
- *Model* — e.g., Fujitsu Esprimo 920
- *Processor* — e.g., Intel Core i5-4570
- *Cores* — e.g., 4
- *Clock Speed* — e.g., 3,6 GHz
- *RAM* — e.g., 4 GB
- *Hard Disk (SSD/HDD)* — e.g., 500 GB
- *Graphics Card* — e.g., Intel Ivybridge Desktop
- *Network* — e.g., Realtek Ethernet
- *Cache* — e.g., 6144 KB
- *Mainboard* — e.g., Fujitsu D3171-A1
- *Operating System* — e.g., Ubuntu 18.04

For Blue Angel eco-certification, the above data needs to be added to 2 documents called "Annex 1" and "Annex 2". Again, you can find examples for Okular at the following links:

- Annex 1
- Annex 2

(C) User Autonomy

As discussed in Part II, the Blue Angel user autonomy criteria cover eight general areas:

1. Data Formats
2. Transparency
3. Continuity Of Support

4. Uninstallability
5. Offline Capability
6. Modularity
7. Freedom From Advertising
8. Documentation

Many FOSS projects may take for granted that Free Software respects user autonomy, and in some cases information from the above list is missing from websites, manuals, wikis, etc—these might include support for open standards in data formats, uninstallability, or continuity of support.

Documenting this information is important, both for fulfilling the Blue Angel award criteria and for giving users information about long-term sustainable use of their software and hardware.

This is not an exhaustive presentation for each of the above categories of the Blue Angel criteria. Rather, this guide focuses on aspects of the criteria which KDE/FOSS projects can easily document and provide (which is already most of the work). For the full criteria, see Section 3.1.3 in the award criteria.

2.1 Data Formats

The main information to include in documentation:

- Which (open) data formats are supported—with links to specifications, e.g., PDF?
- Also of interest: Are there examples of other software products that process these data formats?

For an example of the online documentation of supported data formats for Okular, see more here..

An example of documentation in a Blue Angel application can be found here.

2.2 Transparency Of The Software Product

When missing, provide links to documentation of the API, source code, and the license.

For the sake of example, for KMail:

- KDE PIM API documentation
- Source code
- License

An example of documentation in a Blue Angel application can be found here.

2.3 Continuity Of Support

Examples of details about continuity of support to add to documentation include:

- Information about how long the software has been supported for (with links to release announcements);
- Release schedule and details (e.g., who maintains the software);
- Statement that updates are free of charge;
- Declaration on how the free and open source software license enables continuous support indefinitely; and
- Information about whether, and how, functional and security updates may be installed separately.

An example of Okular's continuity of support documentation for Blue Angel certification can be found in Section 3.1.3.3 of the application.

2.4 Uninstallability

How are users able to completely uninstall the software? Relevant details might include:

- Uninstallation instructions that depend on how the software was installed (source code or binary).
- Examples of uninstallation instructions (source code or package managers, with relevant links to documentation).
- Information about whether user-generated data is also removed when uninstalling a program.

An example of Okular's uninstallability documentation for Blue Angel certification can be found in Section 3.1.3.4 of the application.

2.5 Offline Capability

This could be one of the easiest areas for most FOSS projects to document (alongside freedom from advertising; see below): Does the software require external connections such as a license server in order to run? If not, and no network connection is needed since the software can be used offline, this should be documented.

An example of Okular's offline capability documentation for Blue Angel certification can be found in Section 3.1.3.5 of the application.

2.6 Modularity

Information to document includes:

- What aspects of the software are modular and can be deactivated during installation?
- Can the software manuals or translations be installed separately?
- Are any modules unrelated to the core functionality of the software included with installation, such as tracking modules or cloud integration? If not, document it!

An example of Okular's modularity documentation for Blue Angel certification can be found in Section 3.1.3.6 of the application.

2.7 Freedom From Advertising

If the software does not display advertising, make this explicit in manuals and wikis and declare it in the Blue Angel application document.

2.8 Documentation

This includes the following:

- General process for installing/uninstalling the software? This may include generic instructions or tutorials for a specific desktop environment or package manager.
- Data import/export process?
- What can users do to reduce the use of resources (e.g., configuration options for improving performance)?
- Does the software have any resource-intensive functionality not necessary for the core functionality? If not, great. Let's tell the users!
- Licensing terms related to further development of the software products, with links to source code and license?
- Who supports the development of the software?
- Does the software collect any personal data? Is is compliant with existing data protection laws? If yes, document it!
- What is the privacy policy? Is there telemetry, and if yes, how does the software handle data security, data collection, and data transmission? Also, are there ads or tracking embedded in the software? If not, excellent—now make sure to spread the word!

An example of Okular's product documentation for Blue Angel certification can be found in Section 3.1.3.8 of the application.

Examples Below are examples of Blue Angel documentation for Okular.

- Annex 4: Data Formats (Section 2.1)
- Annex 5: Open Standards (Transparency Section 2.2)
- Annex 6: Product Information (Sections 2.3–2.8)
- See also Annex 7: "Data format for passing on the product information about resource and energy efficiency"

Submitting Your Application For The Blue Angel

For examples of all of the above documentation, see KDE's Blue Angel Applications repository.

Once you have all of the documentation prepared, you need to submit it for review to RAL gGmbH (if you recall, RAL is the authorized body that assesses compliance with the award criteria). The portal for submitting Blue Angel applications can be found here (https://portal.ral-umwelt.de/).

If you need help with the online interface, RAL provides documentation.

Funding Notice

The *Blauer Engel Für FOSS* project was funded by the Federal German Environment Agency (UBA) and the Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection (BMUV). The funds are made available by resolution of the German Bundestag.

Umwelt Bundesamt

Figure 31: Logo of the Federal German Environment Agency.

Federal Ministry
for the Environment, Nature Conservation,
Nuclear Safety and Consumer Protection

Figure 32: Logo of the Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection.

The publisher is responsible for the content of this publication.

PDF Generation

This PDF was generated using the Evisogel pandoc LaTeX template by Pascal Wagler.