Successful Online Education — GeckoCIRCUITS as Open-Source Simulation Platform

A. Müsing,
J. W. Kolar

Successful Online Education – GeckoCIRCUITS as Open-Source Simulation Platform

Andreas Müsing
Gecko-Simulations AG
Physikstrasse 3, 8092 Zurich, Switzerland
www.gecko-simulations.com
andreas.muesing@gecko-simulations.com

Johann W. Kolar
Power Electronic Systems Laboratory, ETH Zurich
Physikstrasse 3, 8092 Zurich, Switzerland
www.pes.ee.ethz.ch
kolar@lem.ee.ethz.ch

***Abstract*—Modern lectures in power electronics should be complemented by interactive content, e.g. computer assisted learning via web-based applets or exercises employing a circuit simulator. The online e-learning system "iPES" was established in 2001; since then the iPES Java applets are frequently used in academia and industry to teach the basics of power electronics. This paper introduces new features of the open-source power electronics circuit simulator GeckoCIRCUITS, which enable a more flexible way to generate online content in comparison to iPES. In particular, the animation of current paths in a circuit model as well as a new language translation and internationalization scheme will be presented in detail. Finally, the popular iPES platform will be enhanced or replaced with Java applets based on GeckoCIRCUITS.**

***Keywords*—Power electronics circuit simulation, online education, open-source, current path animation.**

I. Introduction

Many typical engineering problems are quite complex, show dynamic behavior and a large number of different operating states. It is difficult for a lecturer to describe complex interrelationships and parameter dependencies solely via verbal description and static pictures. In power electronics, current conduction paths depend on diode and switch conduction states, and PWM control adds another level of complexity. The student often finds it difficult to understand how a system works if he is only passively following a lecture in the classroom.

To address this issue, the Power Electronic Systems Laboratory at ETH Zurich introduced the online education platform iPES in 2001 [1]-[4]. An example of a typical iPES Java applet is shown in Fig. 1 [5]. Later on, the circuit simulator GeckoCIRCUITS was developed and used as an extension to the iPES platform [6]. Since 2013, GeckoCIRCUITS is available as open-source software published under the free GNU Public License [7]. Hence the software can now be used for free. All exercises in the basic and advanced power electronics courses at ETH Zurich include a GeckoCIRCUITS simulation model [8]. Additionally, specific simulation exercises motivate the students to use a modern simulation tool to understand complex lecture contents.

This paper introduces two new features of GeckoCIRCUITS which are especially attractive for educational purposes: A dynamic current path visualization and a novel scheme for translating the graphical user interface into arbitrary languages. The main intention for implementing the new functionality was to reach a wider international audience in education for the simulation tool GeckoCIRCUITS.

This paper is structured in the following way: In the next section, the special requirements for a circuit simulation platform in education are discussed. The concept of the original iPES applets is reviewed in Section II. The new current path visualization and animation feature of GeckoCIRCUITS is then introduced in Section III, including some of the technical implementation aspects.

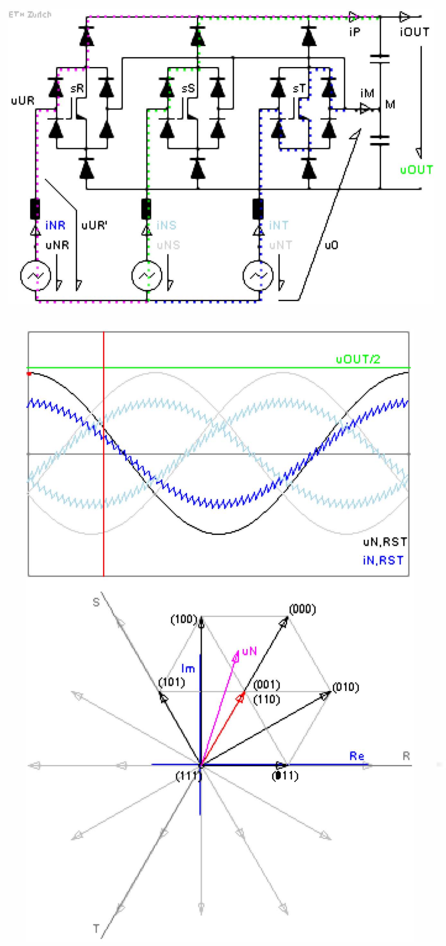


Fig. 1. A Vienna Rectifier iPES education applet including a current path animation and a space-vector visualization [5]. The red time slider in the waveform graph can be used to modify the animated circuit switching state and the space-vector display.

Section III describes the new GUI translation and internationalization mechanism of GeckoCIRCUITS. Finally, a conclusion and an outlook to the future development of the education platform is given.

A. Requirements for a Successful Education Tool

Considering circuit simulators for power electronics, there are several commercial and few free tools available. However, a proper choice of a circuit simulator for education in power electronics depends on many requirements, namely the following criteria:

1. Students of a power electronics lecture are problem oriented, i.e. their primary goal is to understand the teaching content. Thus, the introduction of a new simulation tool should not lead to any additional challenges. Most important, the software has to be intuitive and easy-to-use.

2. For many academic institutions, the software licensing costs of commercial circuit solvers are too high, and it is often not possible to provide classroom licenses to groups of typically 10-50 individual students. Various software vendors offer academic institutions discounts on their software licenses. However, due to the large number of needed licenses, the annual cost still may remain too high. Another approach is that software vendors distribute a limited version of the software tools for free, e.g. the number of circuit components is constrained to a certain maximum. However, ideally the software used in combination with a course should be available for free without any limitations. A further benefit is that students may use the software at home or on their laptop computer outside the university classroom for solving exercises.

3. Circuit simulation in power electronics often suffers from stability and convergence issues. Therefore, a SPICE-based solver is not very well-suited for power electronics circuit simulation. Often, a piece-wise linearization of non-linear components such as switches and diodes is faster, more stable and is adequately accurate [15]. Additional desired features for a good power electronics circuit solver are thermal modeling and loss calculation, analog and digital control blocks, and a feature-rich toolbox for simulation result visualization.

4. The graphical user interface of the simulation software should be available in many different languages; proper internationalization is unfortunately not always available in commercial tools. It is often assumed that software users in industry and academia are sufficiently versed in the English language. For students not having English as their mother tongue, the missing translation is a great hurdle.

5. Other desired but not absolutely necessary features for educational purposes are platform independence, i.e. the tool has to run on the main operating systems Windows, Linux or OS X, as well as the capability to provide course content online.

B. GeckoCIRCUITS as Open-Source Simulation Platform

The power electronics simulator GeckoCIRCUITS was developed in the platform independent programming language Java, under consideration of all the above mentioned requirements. The software was published at the beginning of 2013 as free software under the GNU General Public License (GPL). This open-source software license does not only permit to use the software for free, but additionally the Java source code of GeckoCIRCUITS is available as download [7]. It can be inspected, modified and extended by anyone. The GPL ensures that the software is available also in future without any license cost. Hence, GeckoCIRCUITS evolved into a very powerful, easy-to-use, and easily accessible simulation tool for education in power electronics.

II. Review of the iPES Online Education Platform

A. The Basic Concept of iPES

One of the problems in studying and teaching power electronics is the large number of possible reactions of a system to parameter changes. Another challenge in understanding circuit topologies is that a converter system typically goes through a sequence of different states. The traditional way of presenting such content is by using a sequence of slides showing a single system state displaying the behavior of the circuit for a particular set of parameters.

A main idea of iPES [4] is to provide Java-based animations where a “time slider” can be dragged by the student via the mouse; the slider position directly shows the related system state. Figure 2 shows an iPES Java-applet of a single phase PWM converter [9]. The time slider is the vertical red line in the waveform diagrams. According to the position of the time slider, the current flow of the associated path in the topology is animated employing moving dots.

Another basic concept of iPES applets is the possible manipulation of some predefined system parameters via the graphical user interface and interaction points, which are always indicated in red color. The user interaction allows to immediately investigate the resulting behavior of the system. In order to draw focus of the applet user to the most important circuit properties, the number of possible interactions is kept thereby intentionally low. Thus, the user is not distracted by irrelevant parameters and the learning effect is maximized. The variable system parameters allow the animation of interrelations within a converter system, for example the influence of switching frequency or the inductance on a current ripple.

Besides iPES, other projects of web-based interactive animations have been developed for power electronics in Java or Adobe Flash, e.g. [10] - [14]. Some of these projects used the basic principle and design of iPES as foundation.

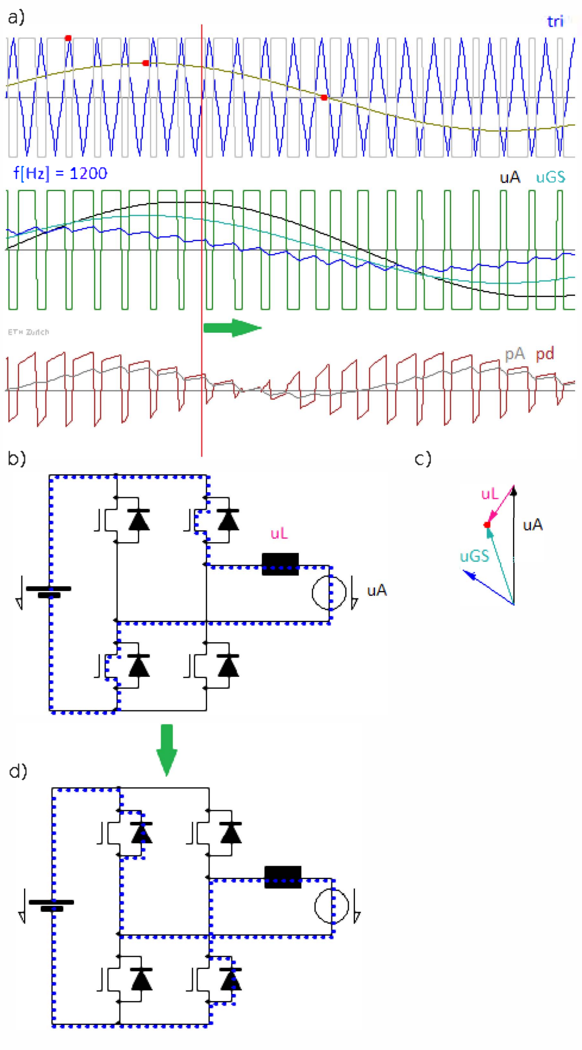


Fig. 2. A single-phase PWM converter as iPES online Java applet [9]. a) The model parameters indicated in red color (PWM frequency, load phase shift, and time-slider) can be modified interactively by the user. The voltage vector diagram c) is updated according to the load current phase shift and amplitude. By moving the time-slider in direction of the green arrow, the animated current path changes its state from b) to d).

B. Acceptance and Users of iPES

The education platform iPES went online in August 2001. Up to now, it has been translated into 12 languages (in alphabetical order: Arabic, Chinese, Dutch, English, Farsi, French, German, Italian, Japanese, Korean, Spanish and Vietnamese). Currently, iPES has up to 500 visitors per day and on average more than 10’000 visits per month. The worldwide user distribution is shown in Fig. 3. The good acceptance of iPES in Asia can be attributed to the translation into many Asian languages. This indicates that a GUI translation of the circuit simulator GeckoCIRCUITS is desired, too, for generating a sufficient user acceptance. Therefore, a new software translation approach will be presented in Section IV.

C. Shortcomings of iPES Applets

Applets in iPES allow only a small number of parameters to be changed by the user. A further restriction is

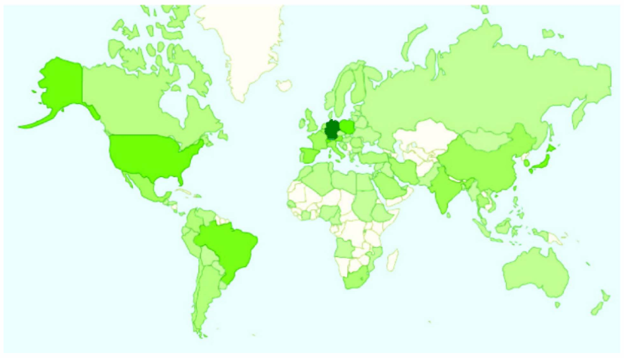


Fig. 3. The figure shows the global distribution of iPES-visitors based on 10’000 visits per month from more than 100 countries. The color intensity indicates the number of visitors per country. 30% of the visitors are from Germany (1st), 5% are from Japan (2nd).

that parameter changes have to be calculated numerically, but the user should experience an immediate effect. Accordingly, the underlying simulations, e.g. long-term transients in converter systems with low damping caused by a parameter change have to be calculated extremely fast.

The technical solution when implementing the iPES applets was to accept numerical inaccuracies by employing large numerical time steps and adding artificial damping resistors. The parameter variations then modify the shown waveforms. The changes are felt as immediate due to the fast but partly inaccurate simulation. The relationships and tendencies are all correct, but the values might show errors up to 10%. Numeric values are therefore not given in many iPES applets. After having gained experience and understanding in power electronics with the educational iPES-applets, many users have asked for real circuit simulation and for much more flexibility. This was the motivation to integrate online circuit simulation with the software GeckoCIRCUITS into the platform iPES [6].

Running Java applets requires the user to have a Java Runtime Environment (JRE) installed on the computer and enabled for the internet browser. At the moment, more than 90% of all computers have Java installed. However, running iPES or GeckoCIRCUITS on an Android based tablet computer is not yet possible. Recently, Java had some negative headlines in the media concerning security issues. Since then, many internet browsers allow to run Java applets only in case the most recent Java update is installed. This is an inconvenience which must be taken into account when using online education platforms based on Java. Nevertheless, computer security with Java is superior in comparison to online contents written in other internet technologies such as Adobe Flash.

The biggest shortcoming of iPES is the static property of its compiled Java binaries. The applet collection cannot cover all aspects of lectures at other courses outside of ETH Zurich and a modification or extension is difficult for electrical engineers without programming skills in Java. Another disadvantage of iPES is its large administration effort. Currently, more than 150 single Java applets are available in 12 different languages. Due to this large number, software maintenance and bug-fixing is quite challenging.

Hence, our goal is a replacement of most iPES applets with simulation models in GeckoCIRCUITS. The new features as introduced in this paper are a major step to achieve this effort. The main benefits of an easy-to-use but yet powerful circuit simulator can then be used by the power electronics engineer to generate educational applets tailored for a specific lecture.

III. Current Path Visualization and Animation in GeckoCIRCUITS

A main feature of many iPES applets is the dynamic visualization and animation of current paths within the circuit schematic, see Figures 1 and 2, and Section II-A. The idea of including similar current path animations in GeckoCIRCUITS is obvious and adds educational value to the circuit solver. Besides its use in education, this feature is also useful for debugging complex simulation models. Missing or wrong gate signals, short circuits, etc. are thereby detected easily in a circuit model.

The concept of circuit simulation with animated currents is not new. Besides iPES, there is a similar functionality in other software packages. However, the tools available at [18] and [19] are just simple circuit applets comparable to iPES, which do not permit the user to build more complex circuits for power electronics. The power electronics circuit solver Caspoc [16], [17] includes a current animation feature, as well. However, the proprietary software Caspoc is only available as commercial tool. Furthermore, its current visualization does not show the current flow direction, connectors are here displayed with varying colors depending on the current flow.

At first glance, the implementation of a current animation seems to be straight forward. However, during the development and application of the new feature for GeckoCIRCUITS, some implementation issues have become apparent which had to be solved for a full functionality within the circuit solver. The are discussed in detailed in the following paragraphs.

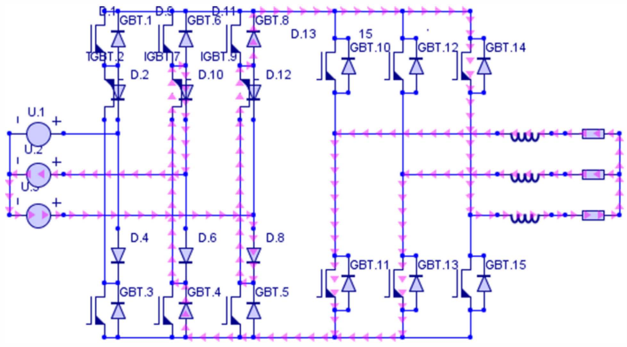


Fig. 4. Current path animation of a Sparse Matrix Converter in GeckoCIRCUITS. In the applet [20], the arrows in magenta move into current direction.

TABLE I. Memory consumption of Sparse Matrix Converter Current Animation from Figure 4).

Approach	Memory consumption
Float-storage of all current values	13 MB
Single-byte representation of data	3.2 MB
Commutation events are considered only	80 kB
Repeated states saved as object reference	30 kB

A. *Memory Consumption of the Current Path Animation*

The simplest approach for the new animation feature would be to store all component current values during a simulation run. Then, the software can generate the animation based on the data stored in memory. However, this naive approach would lead to an excessive RAM memory consumption. Java applets are limited in memory, and thus this simple approach would only be applicable to very small simulation models. Therefore, the software takes advantage of the following properties:

- For the current animation, it is not necessary to store component currents as floating point values. All required information can be stored within three states: Forward conduction, backward conduction, and no current (off-state). Hence, the smallest usable data type has 8 bits (1 byte) in contrast to 32 bits of a float data type.

- Data storage during all simulation steps is not necessary. The actual animation state only changes its value after switching events and/or a current commutation. Hence, the frequency of state changes is in the order of the converter switching frequency and not the simulation stepwidth Δt.

- In power electronics, identical or similar conduction states are repeating several times during a simulation. Therefore, the software has to save a specific animation state only once. An additional list contains those points in time where an animation state needs to be changed. The list entries are then linked to the corresponding animation state.

Applying all above mentioned optimizations, the memory requirement of the animation feature was greatly reduced. Table I summarizes the memory consumption for the Sparse Matrix Converter example from Fig. 4. The circuit model consists of 42 components, and the simulation duration and step width were set to $T = 20\,\text{msec}$ and $\Delta t = 250\,\text{nsec}$. During the simulation, 1550 commutation events with 117 unique circuits states were detected by the software. Table I indicates that the presented approach allows to run even large simulation applet models without memory problems.

B. *Current Animation of Complex Interconnections*

Within GeckoCIRCUITS, the user can set up arbitrary component placements with complex wire interconnections. Thereby, wire layout properties have to be considered in addition to the ordinary circuit simulation. This is illustrated in Fig. 5. The current animation direction is immediately obvious for the example in Fig. 5a): The marked connector flow directions ① and ② at star point

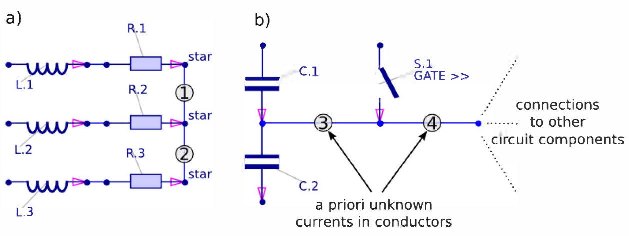


Fig. 5. Component configurations, where the current animation of wire segments has to be calculated as post-processing step based on Kirchhoff's current law.

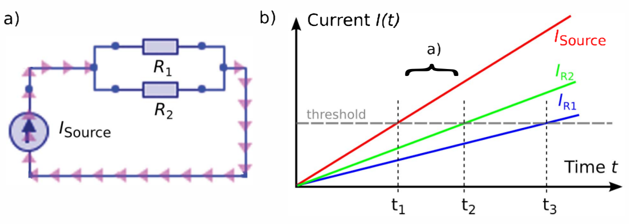


Fig. 6. Current path visualization without error correction: a) shows a loophole in the current animation. The problem can be recognized in b): The total current is larger than the threshold, whereas both resistor currents are below the threshold value.

potential are identical to the current values of resistors R.1 and R.3, respectively. The situation is more complicated for Fig. 5b). Here, the current flow direction is not immediately available at the wire segments ③ and ④. A circuit solver considers wire connections just as identical potentials of the connected circuit component terminals. Therefore, GeckoCIRCUITS had to be extended to take the connection topology into account. Basically, all simple wire connections without branching should be considered here with a new current measurement component. Finally, the missing current measurement is achieved by applying Kirchhoff's nodal rule, i.e. the sum of all incoming node currents has to be equal to zero. If several basic wire sections are connected to each other, Kirchhoff's current law has to be applied recursively until the system of equations is exclusively based on known circuit component currents. Once the network graph is analyzed, the additional effort for calculating all current directions in a postprocessing step is negligible during the circuit simulation. In the example of the Sparse Matrix Converter in Fig. 4, the calculation of the DC link currents is done by summation of 6 components at the converter output stage, respectively.

C. *Current Commutation and Minimum Threshold Value*

Based on a provided threshold value, the circuit solver determines if the animation of a conductor or component is in on- or off-state at a specific point in time. However, this is leading to the problem as depicted in Fig. 6. For $t = 0$, no current is visible at all, all components and conductors are in off-state. Then, the erroneous current animation state for the time interval between t_1 and t_2 is shown in the figure. The total current is larger than the threshold value. However, it is split into two resistor currents that are below the threshold value. This problem is inevitable when a fixed current threshold is used in combination with branching into more than two components. Hence, this issue will appear in most power electronic circuits.

The following approach helps to avoid the above mentioned visualization problem: The time intervals with wrong animations are typically quite short. Therefore it is possible to remove this time interval from the set of animation states. The problem can be detected quite easily: As soon as an activated current animation enters a node, at least one additional animation path with opposite sign has to exist. If this is not the case, the corresponding state should be removed by merging nearby current states at the center of the flawed current state time interval.

IV. A Novel Approach for Language Translation and Internationalization of GeckoCIRCUITS applets

A very important requirement for a successful online simulation and education tool is the software's capability for internationalization, i.e. the graphical user interface and the applet contents should be available in several or even all possible languages. Many commercial tools unfortunately do not offer any internationalization capability or only a translation into a few common languages. English is the main language in the scientific community. However, the amount of students having difficulties to work with education tools that do not allow internationalization should not be underrated. Therefore, the open-source software GeckoCIRCUITS was prepared to be available in all possible languages, e.g. Japanese, Chinese, Portugese, etc.

Often, commercial software manufacturers build language databases for all GUI component strings. This language database is then given to translation agencies for doing the internationalization as paid service. This approach has several difficulties: Professional translations are expensive, and software that evolves over time has to be updated repeatedly. The more languages that are involved, the more expensive it will be to keep all translations up-to-date. Furthermore, if the software contains highly technical terms, even professional translators will have difficulties in translating those terms correctly.

Hence, a new internationalization approach for GeckoCIRCUITS, which reflects both the open-source character and the networking capability of the software, is presented in the following paragraph.

A. *User Feedback for Language Translations*

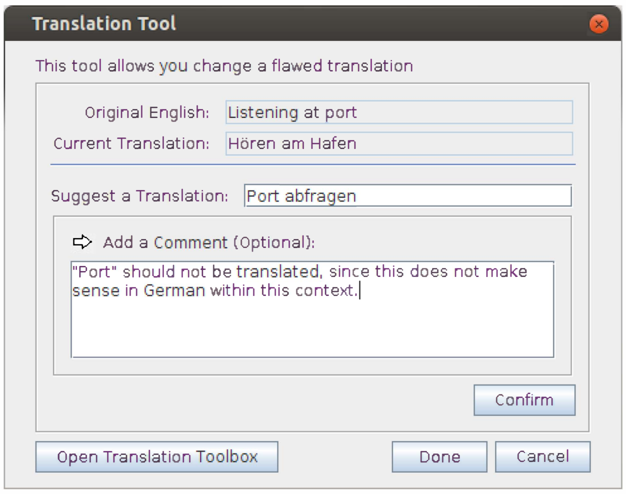
Allowing multilingual users to contribute directly to GeckoCIRCUITS's internationalization resolves the above mentioned problems when dealing with highly technical terms. Users who are well-versed in the technical jargon in their native language can provide the translations and upload their improved suggestion vocabulary to a centrally maintained server. In analogy to the various social/professional networks, GeckoCIRCUITS provides a virtual network of users who contribute to the translation database in a specific language.

Initially, the software provides a database of translations that was generated automatically, i.e. applying the translation API's of Google, Yandex and Microsoft. The initial database contains already more than 50 different languages, which covers most of the users' native languages. Obviously, the quality of the auto-generated database is rather bad.

Users who do not accept poor translations have the option to use English as fall-back language instead of database contents that were auto-generated. As second option, in case the user encounters a translation into his native language which is flawed, he or she has the opportunity to immediately improve the translation and contribute to the vocabulary database.

The following example clarifies the new translation procedure: Let's assume a software user whose native language is German. Most parts of the GUI are translated correctly. However, the string "Listening at Port:", was auto-translated to German as "Hören am Hafen", which does not make sense within its context, since a re-interpretation into English results in "hearing at the harbor". Then, the user can open an internationalization wizard by shift-clicking onto the component that contains the flawed string. A screenshot of the corresponding dialog window is shown in Fig. 7. Here, the user has the possibility to modify any GUI related string. Additionally, an optional comment can be inserted, preferably in English, which clarfies the modification. Comments are later useful for a database administrator without knowledge in German. Possibly, the administrator has to select from several different user suggestions. After pressing the "Confirm" button, the suggestion for improvement is sent via network to a database server and is, in principle, from now on available for the large user community of GeckoCIRCUITS.

Besides the input of a single text selection, the toolbox has a batch translation functionality accessible via the button "Open Translation Toolbox" in Fig. 7. The batch translation can be used for a series of user data input, where all available GUI strings are presented sequentially. Currently, there are about 500 single entries in the language database. In batch mode, the strings are sorted by priority. Since many of the used strings are rarely used, they probably do not require an immediate translation. Details on the actual client-server architecture and the procedure of approving user contributions are given in the next paragraph.


Translation Tool

This tool allows you change a flawed translation

Original English: Listening at port

Current Translation: Hören am Hafen

Suggest a Translation: Port abfragen

Add a Comment (Optional):

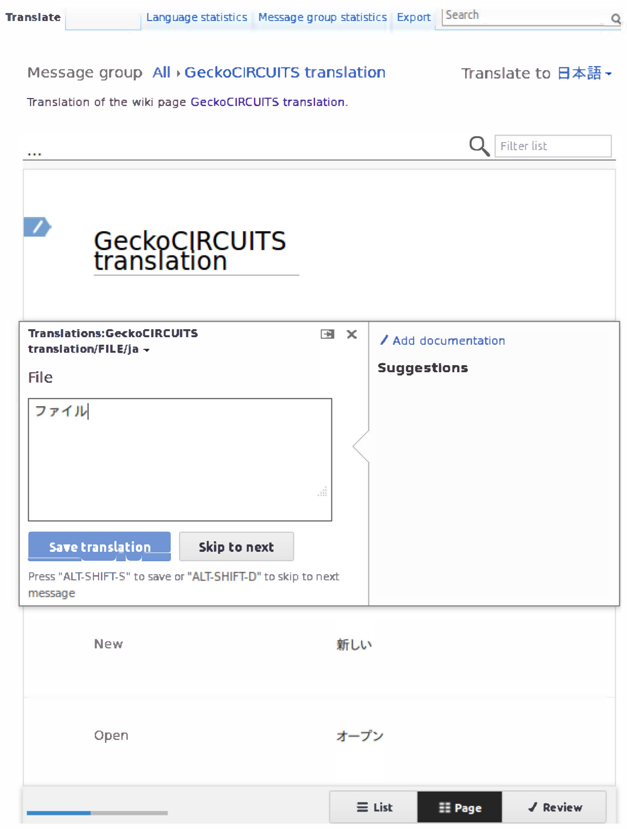
"Port" should not be translated, since this does not make sense in German within this context.

Fig. 7. Dialog window which pops up when the GeckoCIRCUITS user shift-clicks on a GUI component that contains an incorrect native language translation.

B. *Client-Server Architecture of the Translation Toolbox*

The architecture of the new translation approach is built on a client-server model: The Java software runs as stand-alone tool on a personal computer or alternatively within an internet browser and has a network connection to the server that provides the translation database. If a translation is missing or incorrect, the user can directly insert or edit the string and send the new content to the server.

In order to prevent abuse, i.e. to filter undesired content, the contributors of translations are grouped into different categories: Anonymous online users without identification, trusted translators with login and password, and administrators. Translations that are sent to the server have to be peer-reviewed by trusted translators. Finally after a review-process, translations are integrated into the



Message group All › GeckoCIRCUITS translation

Translate to 日本語

Translation of the wiki page GeckoCIRCUITS translation.

GeckoCIRCUITS translation

Translations:GeckoCIRCUITS translation/FILE/ja

File

ファイル

Press "ALT-SHIFT-S" to save or "ALT-SHIFT-D" to skip to next message

New	新しい
Open	オープン

Fig. 8. Browser screenshot of the MediaWiki translation feature which is very useful for database administration. Language administrators can be notified when new suggestions for improvements are available. Furthermore, convenient features as e.g. language statistics and the history are available within MediaWiki.

master database which is actually used by the software all around the world. Without such a review process, it may be possible that undesirable discriminatory text or misleading contributions could degrade the translation database.

A network connection is mandatory for Java applets which run inside an internet browser. Due to security and privacy reasons, Java applets are executed in a so-called "sandbox" within the web browser, preventing them from accessing local data and the file system. The available Java applet memory is limited to 30-120 Megabytes. In order not to waste the limited RAM memory, the applet exclusively loads the language database which was actually requested by the user. For the stand-alone version of GeckoCIRCUITS, a copy of all translation data is stored locally on the hard drive. Since an internet connection is not always available or a firewall might prevent the network communication, the locally stored files are necessary as a backup.

On the server side, quite a special and unique configuration was chosen as database solution. Instead of creating a database system from scratch, the MediaWiki engine runs on the server. This is the same software on which Wikipedia is based. MediaWiki is very easy to use and maintain, all data gets stored in Wiki pages. Tools for administering user accounts together with e-mail notifications and various other aspects of the system are included. All changes made by the user community and administrators are automatically tracked and the page history is stored. These are very convenient features to have in a database. Additionally, there is an extension available for MediaWiki which provides a user interface for translating text within the Wiki. The "Translate Extension" allows an easy administration of the numerous languages and progress statistics. A screenshot of the online MediaWiki interface is shown in Fig. 8. Hence, the effort for setting up the server system of the translation toolbox was negligible.

A schematic overview of the new translation toolbox is given in Fig. 9, and the GUI translation of GeckoCIRCUITS into Japanese is shown in Fig. 10. Since the new software internationalization feature was published just recently, most languages are currently initialized with

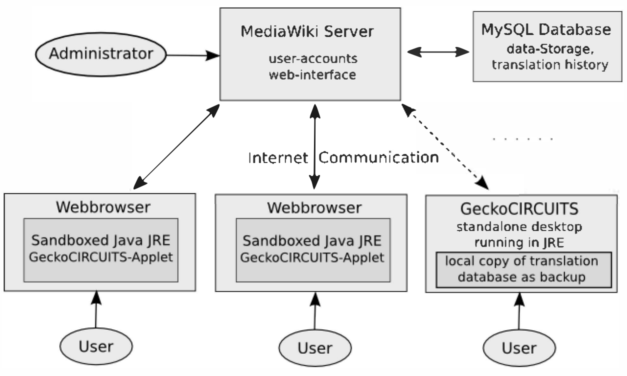


Fig. 9. Schematic overview of the GeckoCIRCUITS translation toolbox client-server configuration.

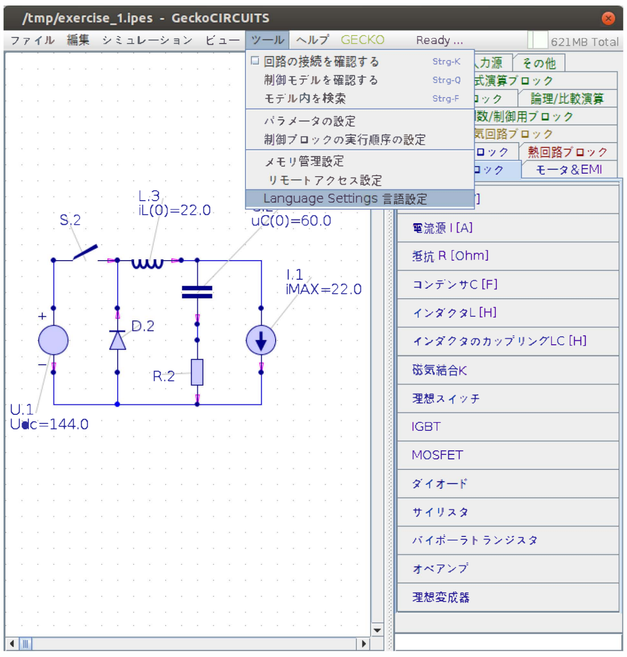


Fig. 10. Screenshot of the GeckoCIRCUITS graphical user interface translated into Japanese.

auto-generated contents which provides not the best data quality. However, with an active user community in the background, the database contents should evolve and improve over time. Every reader of this paper is invited to contribute to the project, for instance online using the Java applet in [20], or alternatively via the stand-alone open-source version of GeckoCIRCUITS, which can be downloaded from [7] for free. The only requirement for a user contribution is the installation of a Java runtime environment, good language skills and some patience.

V. CONCLUSIONS

In the first part of this paper, some requirements for interactive education platforms were briefly discussed. Our conclusions were that such a platform should be easy-to-use and available for free, preferably as open-source software. Then, the well-known iPES applet collection was reviewed, including a discussion of its advantages and shortcomings. iPES is very useful in demonstrating the behavior of power electronic circuits, but does not perform numerically accurate simulations. In order to allow unlimited simulations, the circuit simulation program GeckoCIRCUITS has been integrated into the iPES-platform as an online simulator.

Furthermore, two new features of the open-source circuit simulator GeckoCIRCUITS were introduced which improve the user experience and the educational value: A unique dynamic current path visualization now complements GeckoCIRCUITS as well as a novel language translation toolbox for its graphical user interface. The technical implementation of both features was explained in some detail.

VI. Outlook

The possibilities of online education via Java applets are not yet fully exploited by iPES and GeckoCIRCUITS. Our intention on the long term is a replacement of traditional iPES applets with models implemented in GeckoCIRCUITS. The necessary software adjustments and new features will allow users without in-depth programming skills to generate lecture-specific applets. The following list of features is yet missing in the circuit simulator to finally achieve this goal:

GeckoCIRCUITS has a built-in scripting capability, i.e. all component values, parameters and the simulation itself can be controlled via a simple source code. In order to include all iPES functionalities in the circuit solver, component values of resistances, inductances, etc. should be modifiable via mouse gestures.

The graph visualization of GeckoCIRCUITS should be more closely integrated into the circuit model. Then, for instance, it will possible to drag simulation curves for changing the amplitude of an sinusoidal input voltage or the duty ratio of a gate signal.

As an option, all irrelevant GUI components for the final education applet should be removed when the program is run in its applet mode. This would help the student to focus on the actual education content instead of the GUI of the program.

Finally, the software should provide compatibility with a vector graphics file format, for example the svg format (scalable vector graphics). The contents of many iPES applets is composed by static images and dynamic content. Then it is would be possible to easily draw the static images with an external program, e.g. Adobe Illustrator or Inkscape, whereas the dynamic contents would be implemented within GeckoCIRCUITS.

References

[1] U. Drofenik, J. W. Kolar, P. J. Van Duijsen and P. Bauer, "New web-based interactive e-learning in power electronics and electrical machines," Proc. IEEE Ind. Appl. Conf. 36th IAS Annual Meeting, Chicago, vol. 3, pp. 1858-1865, October 2001.

[2] U. Drofenik and J. W. Kolar, Survey of modern approaches of education in Power Electronics, Proc. of the 17th Applied Power Electronics Conference (APEC) , vol. 2, pp. 749-755, Dallas, March 2002.

[3] U. Drofenik and J. W. Kolar, "Interactive power electronics seminar (iPES) - a web-based introductory power electronics course employing Java-applets," Proc. of the 33rd IEEE Power Electronics Spec. Conf. (PESC), Cairns, vol. 2, pp. 443-448, June 2002.

[4] Online: www.ipes.ethz.ch

[5] Online: www.ipes.ethz.ch/ipes/2002Vienna1/vr1.html

[6] U. Drofenik, A. Müsing and J. W. Kolar, "Novel online simulator for education of power electronics and electrical engineering," Proc. of the Int. Power Electronics Conf. (IPEC-ECCE Asia), Sapporo, Japan, June 2010.

[7] Online: www.gecko-simulations.com/geckocircuits.html

[8] Online: www.pes.ee.ethz.ch/en/our-range/education/courses.html

[9] Online: www.ipes.ethz.ch/ipes/Inverter/e_H_Bruecke.html

[10] V. Ramaswamy, "Interactive power electronics online text," online: http://services.eng.uts.edu.au/$\tilde{v}$enkat/pe_html/peintro.htm

[11] S. Harb, K. Kalaldeh, A. Harb, I. Batarseh, "Interactive java applets for power electronics e-learning," Proc. of the IEEE Workshop on Power Electronics Education, pp. 26 -33, 2005.

[12] J. Hamar , H. Funato , S. Ogasawara , O. Dranga and C. K. Tse, "Multimedia based e-learning tools for dynamic modeling of DC-DC converters," Proc. of IEEE Int. Conf. on Industrial Technology (ICIT), Hong-Kong, December 2005.

[13] J. Hamar, I. Nagy, H. Funato, S. Ogasawara, 0. Dranga and Y. Nishida, "Virtual power electronics: novel software tools for design, modeling and education", Proc. of the Power Conversion Conference (PCC), pp. 502 - 509, Nagoya, April 2007.

[14] U. Probst, "Object-oriented Toolbox for simple generation of Java-Applets for web-based teaching of electrical engineering applications," Interactive Conference on Computer Aided Learning (ICL), Villach, September 2007.

[15] J. H. Allmeling and W. P. Hammer, "PLECS – piece-wise linear electrical circuit simulation for Simulink," Proc. of the IEEE Int. Conf. on Power Electronics and Drive Systems (PEDS), vol.1, pp.355-360, 1999.

[16] P. J. van Duijsen, P. Bauer and B. Davat, "Simulation and animation of power electronics and drives, requirements for education," Taiwan Power Electronics Conf. and Exh., pp. 1048-1053, 2005.

[17] Online: www.caspoc.com

[18] Online: www.icircuitapp.com

[19] Online: www.falstad.com/circuit/e-index.html

[20] Online: www.gecko-simulations.com/IPEC/sparsematrix.html