Web application for analyzing power electronic converter data

Cite as: AIP Conference Proceedings **2172**, 110013 (2019); https://doi.org/10.1063/1.5133616
Published Online: 13 November 2019

Zoran Zlatev, and Nikolay Hinov

Web Application for Analyzing Power Electronic Converter Data

Zoran Zlatev[1, a)] and Nikolay Hinov[1, b)]

[1] *Department of Power Electronics, Faculty of Electronic Engineering and Technologies Technical University of Sofia 8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria*

[a)]Corresponding author: zokizlatev@gmail.com
[b)]hinov@tu-sofia.bg

Abstract. The goal is to provide fast and easy way for engineers to take a look at data oscillation, analyze the information gathered from every element of one convertor, filter the oscillation data by time to see voltage change and compare results. For the making of this web application we used several technologies for saving, collecting and representing data. All data is saved in a relational database from which we take all the information and represent it in our application. In the next part we will discuss about every separate technology.

INTRODUCTION

Every web application needs a server or a host with suitable environment in order to build it and later to work on it. For developing this application, we will use the very famous XAMPP software package which is very easy to install end configure. XAMPP is an open source software developed by Apache friends and contains Apache distributions for Apache server, MySQL, MariaDB, PHP, and Perl. This server is a local host or a local server which programmers usually use to develop and test applications before uploading it to the remote web server. With help of XAMPP software we can develop and test applications on our local computer.

- Apache server is a remote server (computer) which uses HTTP servers for sending and accepting requests.
- MySQL is a relational database management system which uses SQL for organizing, managing, retrieving and updating data whenever we wish to do.
- PHP is a server-side scripting language that helps developers to create dynamic websites. Its main purpose is for building web-based applications and it works very good with MySQL.

XAMPP has been designed to be easy installed and run as a development server on our computer. After downloading and installing XAMPP, we should run the control panel and start the services we need. For our application we are going to start Apache service and MySQL service by clicking START on the control panel.

DATABASE

We are working with relational database which means we have several tables connected with each other with specific keys and data. Every table in our database represents a different model and has its own attributes which describes it. Our database is made on MySQL platform and we use SQ language to save and select data from the database in order to show it on the application.

Our database tables are:

1. Users
2. Convertors
3. Elements
4. Oscillation Data

5. Passwords Resets
6. Migrations

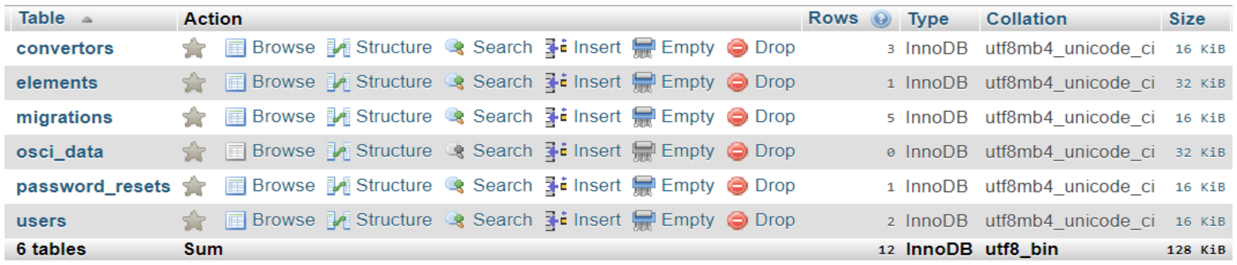


Table	Action	Rows	Type	Collation	Size
convertors	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16 KiB
elements	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	32 KiB
migrations	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_unicode_ci	16 KiB
osci_data	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32 KiB
password_resets	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	16 KiB
users	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_unicode_ci	16 KiB
6 tables	Sum	12	InnoDB	utf8_bin	128 KiB

FIGURE 1. Database structure

Users is a table in which we store our registered users. Every engineer who wants to have access to our web application should make a registration or use another's person account to log in. Every user is defined with its attributes which are: name, email, and password.

Convertors table is used for storing each convertor we want to analyze. Every convertor must be defined with its name, type and photo of the simulation circuit.

Elements is actually the table which refers to the elements of every convertor. This table is connected with the convertors table with the 1: n database relation which means every convertor can have one or many (more) elements. Every different element is described with its name, type and convertor_id. Convetor_id is the attribute which connects every element to their "parent" (convertor).

Oscillation data represents every different value that one element can have in certain moment of time. Every item is defined with time, value and element_id. This table is connected to the elements table with n: 1 database relation which means many oscillation data items can belong to one element. This table is very important because we will use it for drawing element's charts.

Password resets is a table for managing people who will require password reset. In this table every item is defined with email (of the person who asked for password reset) and token which will help authorizing the person who made that request.

Migrations is automatically created table from the framework we are using in the making of this application. This table saves every different migration we use to make changes in our database structure. We will discuss more about this feature in the next parts of the thesis.

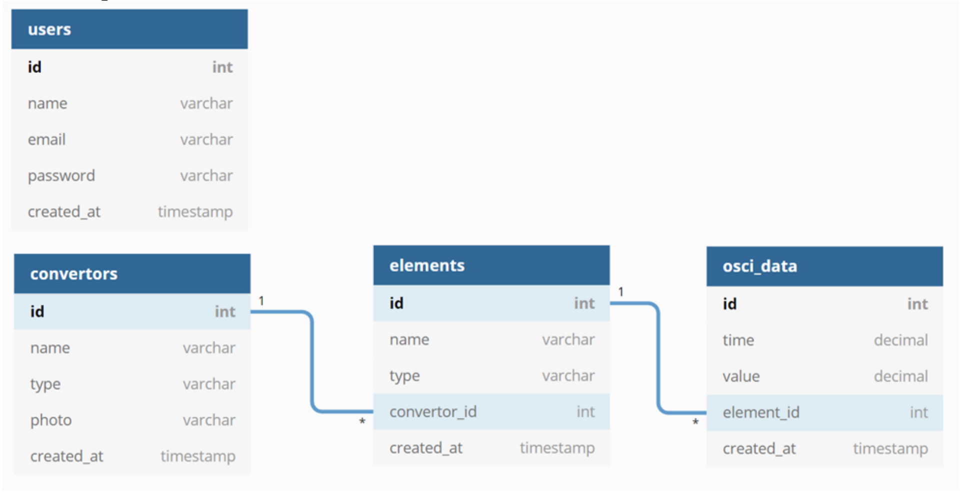


FIGURE 2. Database Diagram

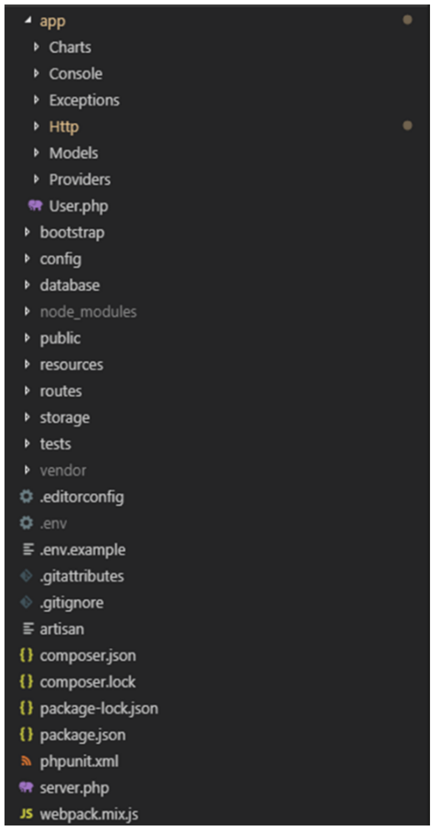
PROGRAM LANGUAGES AND FRAMEWORK

The web application was built using the Laravel framework version 5.8 which is the last stable version of this framework. Laravel is a modern PHP framework that aims at making PHP development easier, faster and more intuitive. This web application was built following the MVC architecture pattern which is the default pattern for working with this framework.

Thanks to the Laravel's ability to manage packages through Composer's Packages online repository, we can easily expand and add new features to our web application. Also, this framework makes the application secure, upgradable and very fast.

Every new Laravel project comes with full directory tree with previously defined structure which can be customizable but provides quick start to the actual development process. This configuration gives all Laravel web applications a similar code structure which is very characteristic and identifiable. This constraint makes it a lot easier to build web applications.

In the next parts, we will discuss on the most important parts of the framework which are crucial to understand for building our web application.



```
app
  Charts
  Console
  Exceptions
  Http
  Models
  Providers
  User.php
bootstrap
config
database
node_modules
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package-lock.json
package.json
phpunit.xml
server.php
webpack.mix.js
```

FIGURE 3. Laravel directory tree

MVC

These days, MVC became the industry's standard practice used in every modern development environment. MVC stands for MODEL-VIEW-CONTROLER pattern which separates the logic behind the application from the representation layer. As we mentioned briefly before, Laravel is a fully-fledged MVC framework.

- Model represents the layer residing between the data and the application. In our application every different model is made for every specific table in which we will save data, and then represent it on the web application. Models can have accessors and mutators to format every attribute value when attempting to retrieve the value or store it in the database. Also, our models will have different functions which are used to describe database relations and other functions for modifying data we need. Models we use are: User, Convertor, Element and Oscillation Data.
- View is actually the representational layer who separates our controllers from our presentation logic. Views are used to show data and they are made with HTML, CSS and JavaScript. Maybe you will wonder why we use these program languages in our PHP framework, but actually Laravel provides the views part where we create the views using the Blade template language and we are able to design them and make them more beautiful. Data that is shown on the views is passed to them as array or key values pairs. Inside the views we're going to access each value using

PHP syntax. We have several views like: convertors.blade.php (used to show all saved convertors), convertor.blade.php (shows one chosen convertor), element.blade.php (shows the oscillation chart of one chosen element from the chosen convertor) etc.

- Controller is a file which contains all the logic behind storing, showing and analyzing data. They contain functions for retrieving views, storing, showing and updating objects. Every route in our application calls to a specific controller function and does a specific functionality. [1-4]

Database management in Laravel

Laravel framework provides an easy to use way for connecting and working with databases. Laravel uses Eloquent ORM which stands for object relational mapping and lets the developer issue database queries where instead of writing SQL code he can simply use predefined methods to interact with the database tables. The Laravel Schema builder is another feature that provides managing all database related work like creating or deleting tables, attributes, changing attributes type etc. Schema builder functions are written in specific files called migrations. They allow us to change the database schema, describe and record all those specific changes in a migration file. After running the migrations, their record is stored in our Migrations table in the database.

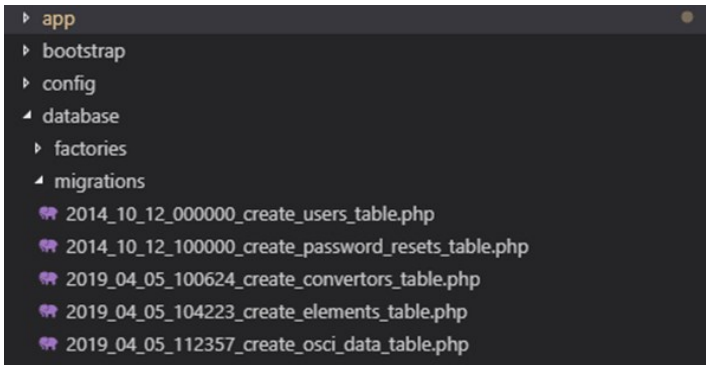
BUILDING THE APPLICATION

As we mentioned before, for building this application we are using the Laravel framework, so we started with creating a fresh new Laravel project. Every new project comes with predefined structure and a lot of files for different use, but in the next part we will discuss only about the parts of the framework we are going to use.

Creating the database

We will fill our database with help of Laravel schema builders, but first we must create the database and for that we start the MySQL service of XAMPP and we can manage our database on the localhost/phpmyadmin url. Our database name is convertors. After creating the database, we configure our .env file in the Laravel project to connect with our database.

After connecting the project with the created database, we can continue creating the tables and attributes only using Laravel functions. As mentioned before, database tables and attributes are created with help of Laravel migrations. Every migration uses Schema builder to create the table attributes and define their type, relation, indexes etc. After creating all wanted migrations we ran the function php artisan migrate in the project's terminal and with that we create all tables and attributes.



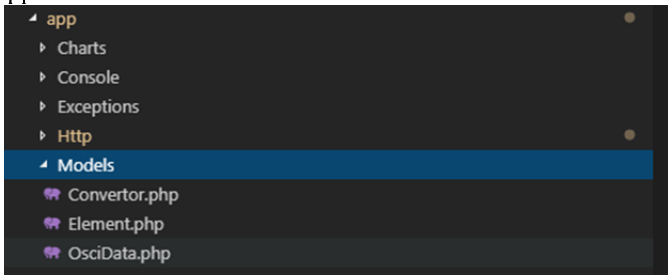
```
▸ app
▸ bootstrap
▸ config
◂ database
  ▸ factories
  ◂ migrations
    2014_10_12_000000_create_users_table.php
    2014_10_12_100000_create_password_resets_table.php
    2019_04_05_100624_create_convertors_table.php
    2019_04_05_104223_create_elements_table.php
    2019_04_05_112357_create_osci_data_table.php
```

FIGURE 4. Laravel migrations files

After creating all tables and their attributes, that doesn't mean that in the future we can't create new tables or change the ones that are already created. We can add new migration files and with help of Laravel we can completely manage our database without using the phpMyAdmin service.

Creating Models

We have already defined the Model property of Laravel framework, but we will use this part to explain how we can create models. [5] By running the command php artisan make:model ModelName we can create all our models. These are the models in our application.



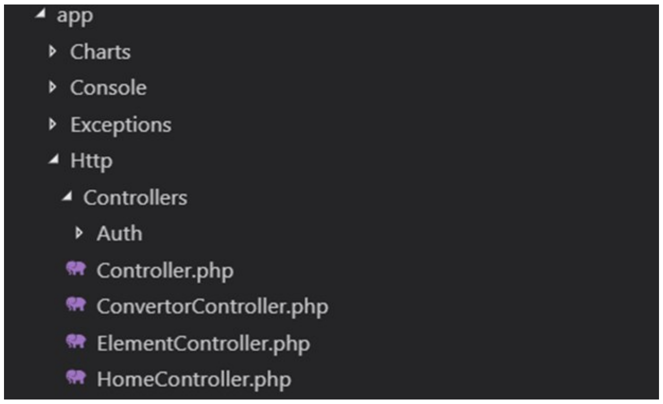
```
app
  Charts
  Console
  Exceptions
  Http
  Models
    Convertor.php
    Element.php
    OsciData.php
```

FIGURE 5. Laravel models files

In these files we will write our model functions and model relationships. Just like migrations, we can add new models while developing and upgrading the web application. [6-9]

Creating Controllers

Controllers are the brain of the application and we use them to write all the most important functions for storing, updating, representing or deleting data. The functions in the controller can return a view, or they can simply do something, call another function anything just like we choose to organize our web application code. Controllers are made with the simple command php artisan make:controller ControllerName and can be done in every step of our development or further. These are the controllers we have in our code:



```
app
  Charts
  Console
  Exceptions
  Http
    Controllers
      Auth
      Controller.php
      ConvertorController.php
      ElementController.php
      HomeController.php
```

FIGURE 6. Laravel controllers files

Creating charts

The main goal of this software was to show how voltage of convertor's elements change in time and draw a chart which will show the data. For drawing these charts, we used a library called Laravel Charts which is specially developed for creating and drawing different charts in Laravel application. This library has a variety of charts based on JavaScript program language and they can be easily implemented without any previously knowledge of JavaScript. This library also offers functions for changing charts type, look, or color. For our use, we chose Highcharts library because it's specially designed to support large datasets, which we have in our case.

Other than drawing charts, we will present our chart's maximum and minimum value, and also in this view we can filter our chart to show values only between chosen time.

Application map

Application map is actually the URL structure of the web application. We must carefully define our URL's because having expressive and clean URLs can really affect how fast, user friendly and easy to navigate will our application be. The best thing about Laravel URLs is that there are already predefined URLs which correspond to the CRUD functions of controllers. CRUD stands for Create, Read, Update and Delete methods. Laravel also defines the methods for this URLs either is POST, GET, PATCH or DELETE method. We are going to utilize the following routes in our web application to fulfill the initial set of requirements.

Short review

As speaking about the process of building the application we only mentioned the basic structure and programming process, every further coding depends on the personal programming style and wanted functionalities. That is why we won't discuss about the details in our code, but in the next few sections we will show some pictures of our code and will explain how all that works.

Exploring the application

This is the part where we will discuss about every view, page and functionalities that our application provides. We tried to make a simple and user-friendly design because it will be mostly used by engineers, so we focused more on the fast execution and correctness of our data analyzes.

Our landing page is simple and symbolic containing the name of the application and buttons for login or register. People who don't have an account, cannot access to the data provides in our web application. For now, there aren't any restrictions about which people can make an account, but that can simply be added in the future development.

If we click on the register button there is a form to fill with our name, email and password for further logging into the application. After making an account it directly takes you to the Dashboard. On the dashboard there is information about the total number of users, convertors and elements.

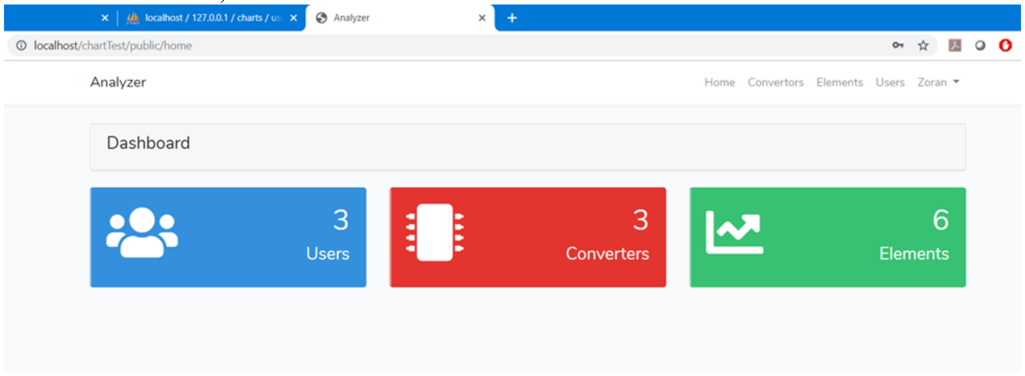
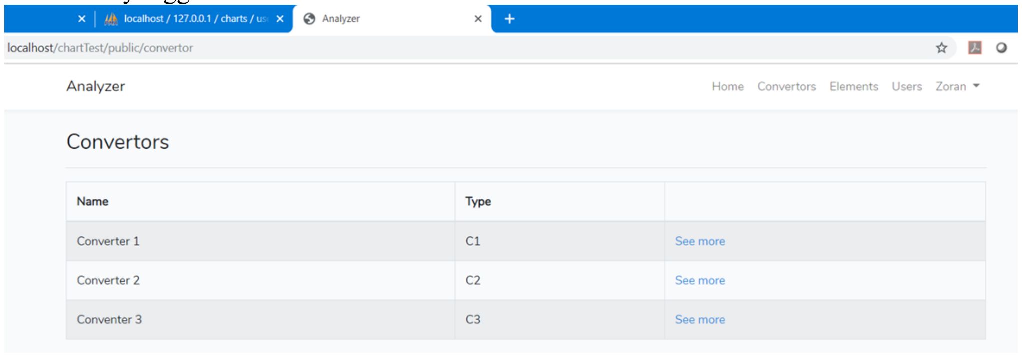


FIGURE 7. Home page of the web application

On the right upper corner of the page we have the menu containing links for the home page, converters page and the name of the currently logged in user.

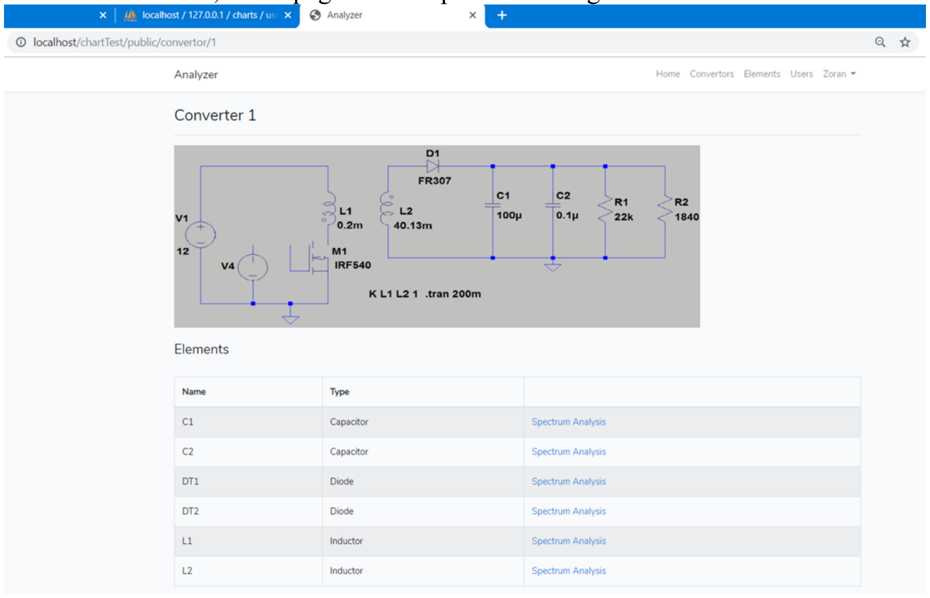

localhost / 127.0.0.1 / charts / us
Analyzer
localhost/chartTest/public/convertor
Analyzer
Home
Convertors
Elements
Users
Zoran
Convertors

Name	Type	
Converter 1	C1	See more
Converter 2	C2	See more
Converter 3	C3	See more

FIGURE 8. Converter's page

From the first page we can navigate to the Converters page where we can see a list from all converters we have in our database. Right now, we have saved only three different types of converters. The table has three columns for converter's name, type and See more link if you want to analyze any of the converters. The table can be further upgraded to show more data. [9-13]

If we click on the See more link, a new page will be opened showing more information about the chosen converter.

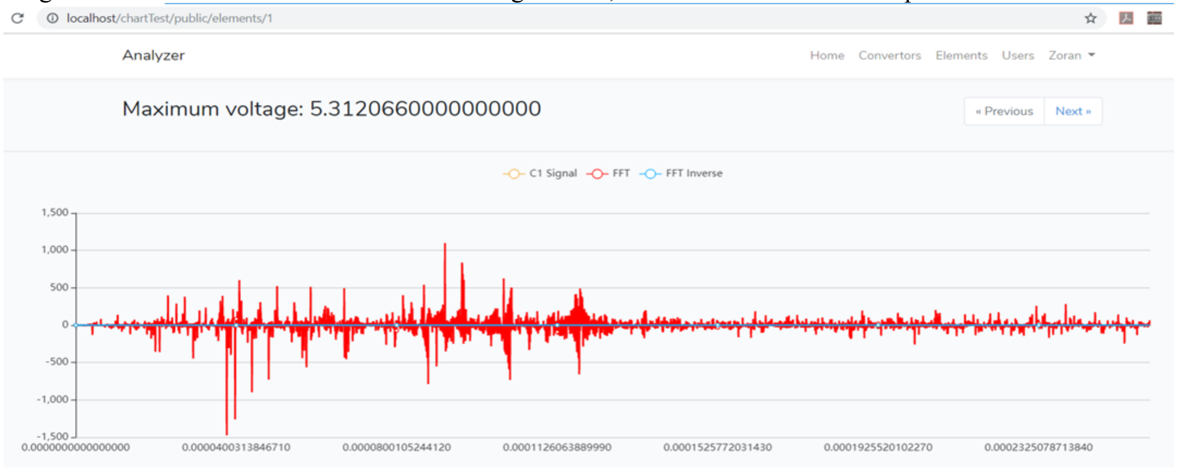


Name	Type	
C1	Capacitor	Spectrum Analysis
C2	Capacitor	Spectrum Analysis
DT1	Diode	Spectrum Analysis
DT2	Diode	Spectrum Analysis
L1	Inductor	Spectrum Analysis
L2	Inductor	Spectrum Analysis

FIGURE 9. Page of one converter

On this page we can see an image of the converter showing how its electrical network is designed and a table showing all converter's elements. Table has columns for element's name, type and link to see the data chart of the element.

By clicking on the link Spectrum Analysis, a new page will be opened showing the data chart of the chosen element with every value of voltage by time, FFT analysis and inverse FFT. This chart is specially designed with library for drawing Charts in Laravel framework. After loading all data, the chart looks like on the picture bellow:



Maximum voltage: 5.3120660000000000

FIGURE 10. Data chart for one element showing voltage change, FFT and FFT Inverse

The element contains more than 140000 samples which are divided into 4096 samples per page. Maximum voltage is showing the highest voltage for this part of the signal. The button next will show us to the next group of 4096 samples and with going previous or next we can analyze the signal in detail. With clicking on the upper labels, we can switch the chart to show only the chosen samples. This is how C1 signal looks like:

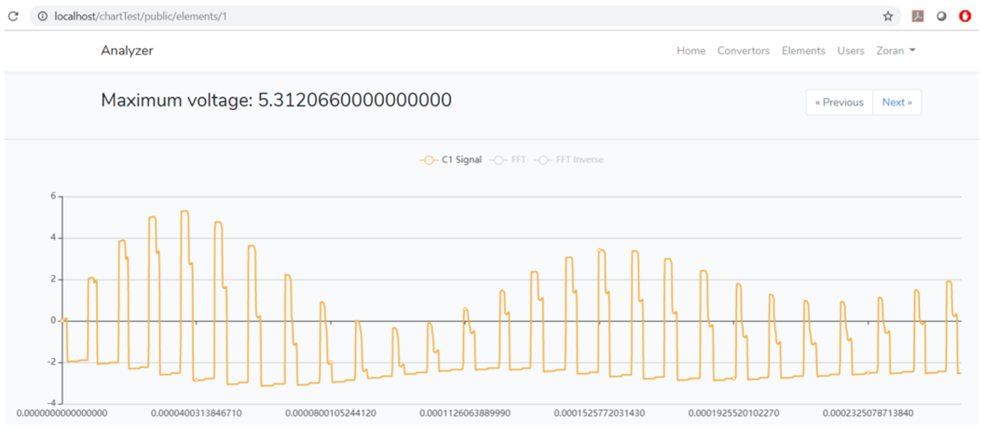


FIGURE 11. C1 signal

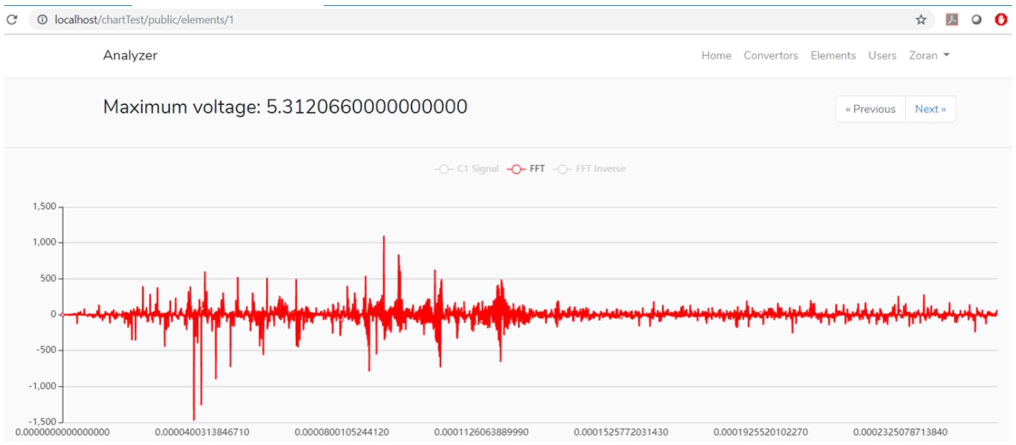


FIGURE 12. Fast Fourier Transformation (FFT) of the C1 signal

The Fast Fourier Transformation is made with help of FFT Calculator library called webit/fft for Laravel and PHP applications. In order to use this library, we have installed it via Composer Packagist. The process of calculating FFT is very simple.

First, we initiate an object from the FftCalculatorRadix2 class. Then, using the ComplexArray class we create complex array of our data values, not including time, and then we use calculateFft function of the FftCalculatorRadix2 class using our complex array and Dimension object measured from our data values. The result is an array of complex numbers, with help of map function we iterate through each value of the array and return the value using the getImaginary function. [14-17]

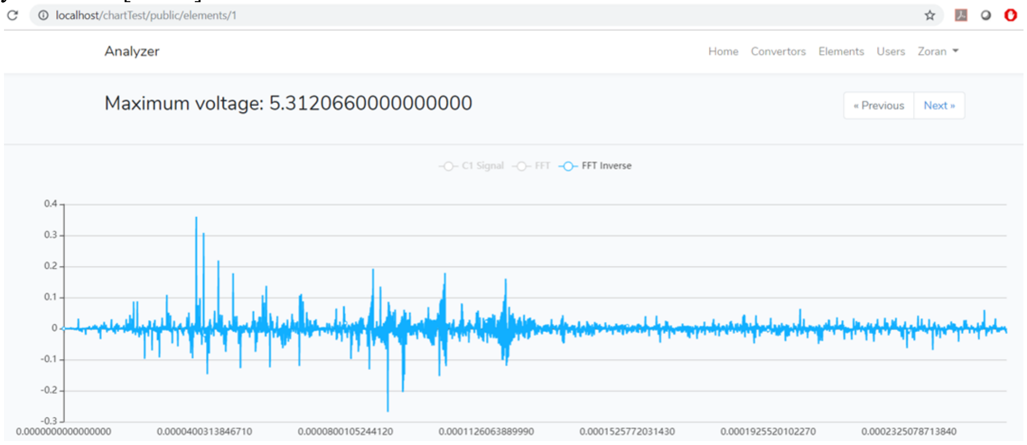


FIGURE 13. Inverse Fast Fourier Transformation (IFFT) of the C1 signal

The InverseFFT calculation uses IFftCalculator class and then uses the complex array for calling the calculateIFtt function.

After getting the array result, we use our chart library to draw the charts. Here is a picture of our FFT Inverse signal chart.

On the same page we also generate data chart showing the results from our Power Spectrum analysis. This chart is not divided, and it shows the power spectrum of the whole signal data values through time.

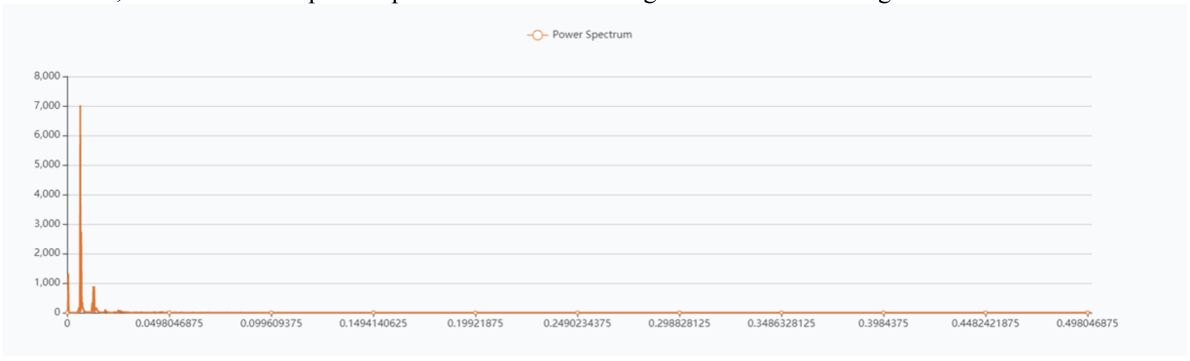


FIGURE 14. Power Spectrum of the C1 signal

For calculating the power spectrum chart, we use Python script which is better for calculating and measuring this kind of calculations. In the script we will import scipy.signal library for Python which is specially designed for handling complex math signal problems.

In the Python script we start HTTP server using BaseHTTPRequestHandler and HTTPServer. This server is used for receiving requests from our Laravel application. Within our Laravel application we sent HTTP request to a specific route within the server started with python, and we sent the data values of which we want to get the power spectrum analysis. Our python script gets the input data and calls a function periodogram from the class signal. The function accepts x which is array like time series of measurement values and returns f (array of sample frequencies) and Pxx (power spectral density of power spectrum of x). After the request is finished, our python server returns the array f and Pxx as json data to our Laravel application, and we use the response inside Laravel to draw the Power Spectrum chart.

CONCLUSION

The paper deals with an interactive software environment for the study of power electronic devices. A description of its main blocks is made, as well as the concept and technologies used to create it. An example of a DC-DC converter is considered.

Tools have been used to obtain different output characteristics of the tested devices. The purpose is to have them freely accessible so that the user does not need commercial versions of the software. Another important aspect of the study is the relationship between the output characteristics described in the paper and the dynamics of the power circuits. The idea is to obtain indirect information about the transients and the controller's stability stock from these characteristics.

Such an approach is particularly useful in power electronics training, where a thorough study of individual devices requires knowledge of a wide range of subjects: electrical engineering, mathematics, physics, theory of automatic control, digital and analog circuitry. In this way, this difficult area of modern engineering becomes attractive to young people, and its knowledge is important for the development of society and industry.

The use of information and communication technologies in the process of modeling, analysis, design and prototyping of power electronic devices is raising these new qualitatively new processes and making them inherently flexible and intelligent. The presented environment is expandable and upgradeable so models created with different open source or commercialized software can be used as input. Thus, the use of information models yields a complete research complex from the idea in the head of the constructor through the model to the finished prototype.

ACKNOWLEDGMENTS

The carried out research is realized in the frames of the project "Model based design of power electronic devices with guaranteed parameters", ДН07/06/15.12.2016, Bulgarian National Scientific Fund.

REFERENCES

1. A. V. Oppenheim and G. C. Verghese, “Introduction to Communication, Control, and Signal Processing” spring 2010.
2. R. J. Dirkman, "The simulation of general circuits containing ideal switches", IEEE Power Electronics Specialists Conf. (PESC), pp. 185-194, 1987.
3. V. Rajagopalan, Computer-Aided Analysis of Power Electronic Systems, New York:Marcel Dekker, 1987.
4. S. V. Vaseghi, “Advanced Digital Signal Processing and Noise Reduction, Second Edition”, 2000.
5. W. H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling,”Numerical Recipes”, *Cambridge University Press, 2nd Edition* p. 542.
6. P. Fishwick, Simulation Model Design and Execution: Building Digital Worlds, Prentice Hall, Englewood Cliffs, 1995.
7. R. M. Fujimoto, Parallel and Distributed Simulation Systems, John Wiley & Sons, New York, 2000.
8. M. Curphey, D. Endler, W. Hau, S. Taylor, T. Smith, A. Russell, G. McKenna, R. Parke, K. McLaughlin, N. Tranter et al., "A guide to building secure web applications", The Open Web Application Security Project, vol. 1, pp. 1, 2002. KELTON, W.D and AVERILL, M., Simulation Modeling and Analysis, McGraw Hill, Boston, 2000.
9. T. Ristenpart, E. Tromer, H. Shacham, S. Savage, "Hey you get off of my cloud: exploring information leakage in third-party compute clouds", Proceedings of the 16th ACM conference on Computerand communications security, pp. 199-212, 2009.
10. E. Kreyszig, Advanced Engineering Mathematics, John Wiley & Sons, New York, ISBN-13: 978-0470458365, ISBN-10: 9780470458365, 2011.
11. J. O. Malley, Theory and Problems of Basic Circuit Analysis, Schaum’s Outline Series, McGraw-Hill, Inc., New York, 1982.
12. L. K. Shar, H. B. K. Tan, L. C. Briand, "Mining sql injection and cross site scripting vulnerabilities using hybrid program analysis", Proceedings of the 2013 International Conference on Software Engineering, pp. 642-651, 2013.
13. J. Bau, E. Bursztein, D. Gupta, J. Mitchell, "State of the art: Automated black-box web application vulnerability testing", Security and Privacy (SP) 2010 IEEE Symposium on, pp. 332-345, 2010.
14. S. Subashini, V. Kavitha, "A survey on security issues in service delivery models of cloud computing", Journal of network and computerapplications, vol. 34, no. 1, pp. 1-11, 2011. LANG, S., Introduction to Linear Algebra, Addison-Wesley Publishing Co., Massachusetts, ISBN-13: 978-0387964126, ISBN-10: 0387964126, 1966.
15. B. Nagpal, N. Singh, N. Chauhan, A. Panesar, "Tool based implementation of sql injection for penetration testing", Computing Communication & Automation (ICCCA) 2015 International Conference on, pp. 746-749, 2015.
16. P. Bisht, T. Hinrichs, N. Skrupsky, R. Bobrowicz, V. Venkatakrish-nan, "Notamper: automatic blackbox detection of parameter tampering opportunities in web applications", Proceedings of the 17th ACM conference on Computer and communications security, pp. 607-618, 2010.
17. P. Bisht, T. Hinrichs, N. Skrupsky, V. Venkatakrishnan, "Waptec: whitebox analysis of web applications for parameter tampering exploit construction", Proceedings of the 18th ACM conference on Computer and communications security, pp. 575-586, 2011.
18. R. Vallée-Rai, P. Co, E. Gagnon, L. Hendren, P. Lam, V. Sundaresan, "Soot-a java bytecode optimization framework", Proceedings of the 1999 conference of the Centre for Advanced Studies on Collaborative research, pp. 13, 1999.