Integration of Software SchematicSolver into Mathematica® Environment

Miroslav Lutovac, *Senior Member, IEEE*, Vladimir Mladenović and
Maja Lutovac Banduka, *Member, IEEE*

***Abstract*—SchematicSolver is an application for symbolic-numeric simulation, analysis and implementation of the linear continuous-time systems and linear or nonlinear digital systems and algorithms. In order to provide for this application to be used in the same environment (the computer algebra system Mathematica), as other built-in functions, the strict architecture of folders and files was developed. Instead of using development platforms that are usually used in other software environments, the whole methodology is based on the Wolfram language. The proposed methodology is applicable for small and medium sized application.**

***Index Terms*—Wolfram language; computer algebra system; analog systems; digital systems.**

I. Introduction

PRODUCTION of application software should follow the general principles but the details may be unique in some specific environments. These principles are quite different than that for the fast prototyping and rapid exploration. The code should be divided into smaller packages and bounded into application. As a standalone product the application can be delivered to users. Computer Algebra System (CAS), such as Mathematica [1], may require additional principles, for example to provide a documentation system that works with other Mathematica applications and to integrate within the Documentation Center. Although application developers can use additional tools in Mathematica, for example Wolfram Workbench, this is not quite appropriate for developers that are using only Wolfram language and with lack of skills in traditional programing. This can be a significant barrier for developers that do not have experience with other development platforms, for example Eclipse, that are very popular in software engineering.

The third party products and specific platforms may work with some versions of targeted environment, but it may not be fully compatible with the most recent versions. It is not unusual to have a new software version each year, with many new additional features but also with changed functionality of built-in commands. This is especially true with the formal beta testing versions.

Miroslav Lutovac is with Singidunum University, Danijelova 32, 11000 Belgrade, Serbia (e-mail: mlutovac@singidunum.ac.rs).

Vladimir Mladenović is with Univ. of Kragujevac, Faculty of Technical Sciences, Čačak, Serbia (e-mail: vladimir.mladenovic@ftn.kg.ac.rs).

Maja Lutovac Banduka is with Lola Institute, Belgrade, Serbia, (e-mail: majalutovac@yahoo.com).

Powerful software systems such as Mathematica can do almost everything that is possible on standard developing platforms, and the development of application can be accomplished using Wolfram language only, for example using ApplicationMaker [2]. The main purpose of this paper is to present an original approach for developing applications in Mathematica environment using Wolfram language. ApplicationMaker is a Mathematica package that facilitates the integration of projects and applications with the Mathematica Documentation Center without need to use the Wolfram Workbench and Eclipse development platform.

The application code was formerly released for Mathematica version 8, and it was tested with all versions up to Mathematica 11. One can conclude that ApplicationMaker is a bit outdated but it still works properly and do not require fixes.

II. SchematicSolver

SchematicSolver is a powerful and easy-to-use schematic capture, symbolic processing, symbolic analysis, and implementation tool in Mathematica [3]. Using SchematicSolver's unique capabilities and mixed symbolic-numeric processing, you can perform fast and accurate simulations of linear continuous-time (analog) systems and linear or nonlinear discrete-time (digital) systems and algorithms. The former version was based on mouse click build-in hidden functions, while the newest version is using some dynamic and GUI (graphical user interface) capabilities [4].

Like many graphical programming languages, SchematicSolver is using graphical icons on several pallets for drawing schematic of any system or algorithm. This schematic description of the system is placed in a variable as a specification list with detailed description of any constitutive element or descriptive entity enabling further symbolic analysis. One possible representation of a system is a block diagram that visualized main functionality of the system or algorithm. Another possibility is to derive some properties of the system or algorithm so that any output can be represented as a function of all inputs, which is especially important for MIMO (multiple-input multiple-output) system. Combining descriptive properties of each element, the list can be used for deriving transformed variables and thus providing the solution of the system in the transformed domain (the continuous time domain, the discrete time domain, the complex variable z for digital systems, and the complex variable s for analog systems). All naturally generated signals are real-valued and are referred to as real signals. But, in some applications, it is desirable to generate signals that are complex-valued, also called complex signals that are more appropriate for mathematical analysis.

Regardless of the initial system description, the transformed system can be easily generated by transforming the description in the specification list. Even more, any of the constative elements can be replaced with one or more connected elements and thus simplified analysis of very complex systems.

The newest version of SchematicSolver has many new elements, such as resistors, capacitors, inductors, current and voltage sources, or operating amplifiers and semiconductor devices.

III. ApplicationMaker

It is likely that developed code for ourselves can end up being forgotten somewhere in hard disk, cloud server, or CD. After some time it is almost impossible to remember the basic idea of the purpose of the code and what is the meaning of the variables. Even more it is more difficult to expect that someone else can use the code without complete description of the purpose and the usage of the developed application.

With completing the usage documentation at the moment of the former implementation, the later usage can be simple for any user. For this reason, integrating documentation for specialized functions with Mathematica documentation center implies creating documentation for a function (as template), creating guides and tutorials, linking the notebooks to the documentation center, creating "usage" and error messages that display correctly in different environments.

ApplicationMaker is such an application that was successfully used for simple application as Vickrey Auction Game in Mathematica [2].

IV. Development of application

In this section we are presenting the complete procedure for the development of the application SchematicSolver.

The initial parts consist of discovering where the default folders are and reading the required knowledge:

```
directory = NotebookDirectory[];
SetDirectory[directory]
folderUserM = $UserBaseDirectory;
folderMathApp=FileNameJoin[{folderUserM,
  "Applications"}];
newApplication = "SchematicSolver";
folderMNewApplicationTest =
    FileNameJoin[{folderMathApp, newApplication}];
FileNames[folderMNewApplicationTest]
<< ApplicationMaker`
```

The first two lines read the folder name where the directory with the current evaluation notebook is (for example **generisanjeSS2DC.nb**) and sets the current working directory to **directory**.

The next line finds the base directory in which user-specific files have to be conventionally loaded by the Wolfram System. To name of this directory is added subdirectory named **Applications**. All application packages will be in that directory. As example, we have names of two folders (**$HomeDirectory** and **$UserBaseDirectory**)

```
C:\Users\MIROSLAV
C:\Users\MIROSLAV\AppData\Roaming\Mathematica
```

Now is a right moment to import knowledge for creating documentation, available from (**<< ApplicationMaker`**).

The next step is to define subdirectories of applications, such as:

```
FileNames[{
 FileNameJoin[{folderMNewApplicationTest,"Kernel"}],
  FileNameJoin[{…,"Documentation"}],
  FileNameJoin[{…,"Documentation","English"}],
    …, "Documentation","English","ReferencePages"}],
    …, "Documentation","English","Tutorials"}],
    …, "Documentation","English","Guides"}],
… "Documentation","English","ReferencePages","Guides"}],
}];
```

The subdirectories should contain some specific files, as is illustrated in Table 1. The SchematicSolver application package has the tree structure presented in Fig. 1.

TABLE I
Required files in subfolders

Folder	File
"Kernel"	"Init.m"
"Stylesheets"	"UserReference.nb"
"SchematicSolver"	"PacletInfo.m"
"Symbols"	"DrawElement.nb",…
"Guides"	"SchematicSolver.nb",…
"Tutorials"	"GettingStarted.nb",…

The PacletInfo.m is a descriptor file for Mathematica application; it is used by the documentation system so that Mathematica can locate documentation based on its settings. The PacletInfo.m contains a set of rules that describe the application and what it contains.

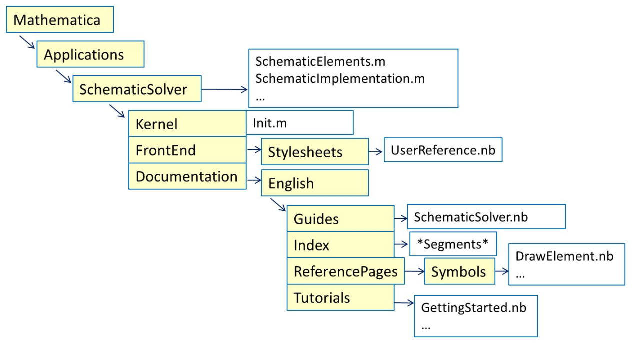


Fig. 1. Tree structure of SchematicSolver application.

The basic step is to identify symbols for the index file. It is necessary to import another knowledge using **Needs["DocumentationSearch`"]**. **$Package** gives a list of the contexts corresponding to all packages which have been loaded in the current Wolfram System session. Using **`$Package`** we can get the list of the contexts corresponding to loaded packages, such as **`DocumentationSearch`**`.

Using the presented code, we can create the folder for all symbol pages (that describe how are working each function in a new package) and the folder for index files:

```
C:\Users\MIROSLAV\AppData\Roaming\Mathematica\
Applications\SchematicSolver\Documentation\English\
ReferencePages\Symbols
C:\Users\MIROSLAV\AppData\Roaming\Mathematica\
Applications\SchematicSolver\Documentation\English\
Index
```

The opening and closing sessions are initialized for documentation search and are preparing index files.

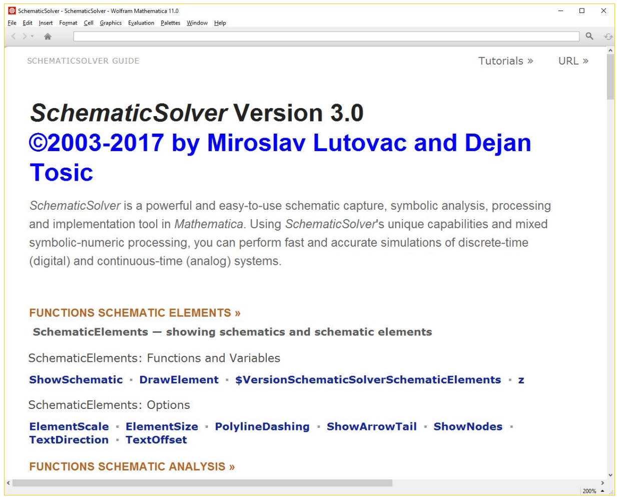
A. Guide, Tutorial, and Function notebooks

To create guide pages, we can use the function **`NewGuide`**. **`NewGuide`** creates a guide template notebook, say **`guideName.nb`**, inside the documentation for the application. If we try to create a guide that has already been created we will obtain an error message with a link to open the existing guide:

```
NewGuide::guideerr: The guide you are trying to create in the application SchematicSolver already exists. Click here to edit its contents. $Failed
```

Each guide notebook is a separate file that can be modified independently from preparing documentation, but the final version should be used for completing the whole procedure of making the application.

The Mathematica help system works with a highly structured set of documents. Each document can be a guide notebook, reference notebook, and tutorial notebook. Each of these pages has a structure and specific style defined in UserReference.nb.


SchematicSolver - SchematicSolver - Wolfram Mathematica 11.0
File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
SCHEMATICSOLVER GUIDE
Tutorials »
URL »

SchematicSolver Version 3.0
©2003-2017 by Miroslav Lutovac and Dejan Tosic

SchematicSolver is a powerful and easy-to-use schematic capture, symbolic analysis, processing and implementation tool in *Mathematica*. Using *SchematicSolver*'s unique capabilities and mixed symbolic-numeric processing, you can perform fast and accurate simulations of discrete-time (digital) and continuous-time (analog) systems.

FUNCTIONS SCHEMATIC ELEMENTS »

SchematicElements — showing schematics and schematic elements

SchematicElements: Functions and Variables

ShowSchematic · **DrawElement** · **$VersionSchematicSolverSchematicElements** · **z**

SchematicElements: Options

ElementScale · **ElementSize** · **PolylineDashing** · **ShowArrowTail** · **ShowNodes** · **TextDirection** · **TextOffset**

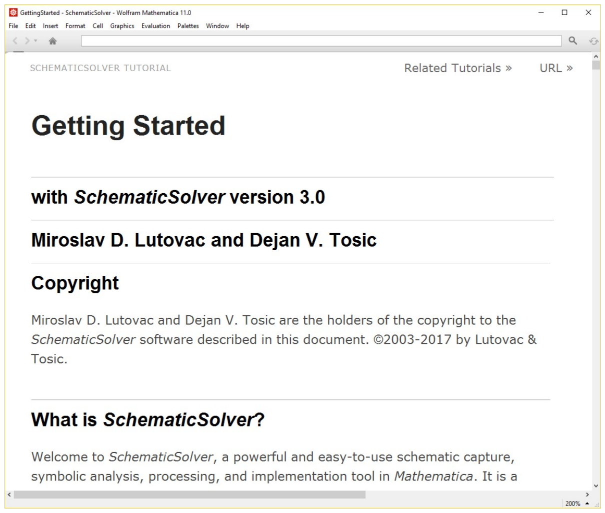
FUNCTIONS SCHEMATIC ANALYSIS »

Fig. 2. Sample guide notebook.

A guide notebook focuses on a single topic, providing links to functions that share functionality. Sections are provided to link to other related documents. A sample guide page is shown in Fig. 2.

Tutorial notebooks focus on the functionality and include short descriptions of useful features, in a form of textbook-like information. Sections are provided to link to other related documents. A sample tutorial notebook is presented in Fig. 3.

All documents generated as guide, tutorial or function description (presented in Fig. 4) are available in a same manner as any built-in Mathematica function.


GettingStarted - SchematicSolver - Wolfram Mathematica 11.0
File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
SCHEMATICSOLVER TUTORIAL
Related Tutorials »
URL »

Getting Started

with *SchematicSolver* version 3.0

Miroslav D. Lutovac and Dejan V. Tosic

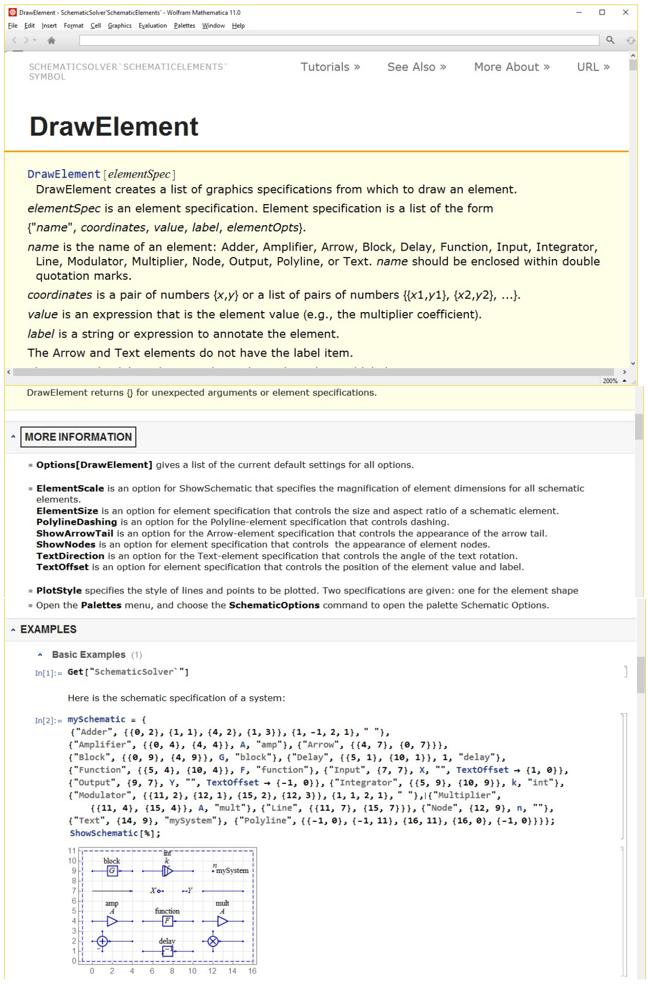
Copyright

Miroslav D. Lutovac and Dejan V. Tosic are the holders of the copyright to the *SchematicSolver* software described in this document. ©2003-2017 by Lutovac & Tosic.

What is *SchematicSolver*?

Welcome to *SchematicSolver*, a powerful and easy-to-use schematic capture, symbolic analysis, processing, and implementation tool in *Mathematica*. It is a

Fig. 3. Sample tutorial notebook.


DrawElement - SchematicSolver`SchematicElements` - Wolfram Mathematica 11.0
File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
SCHEMATICSOLVER`SCHEMATICELEMENTS` SYMBOL
Tutorials »
See Also »
More About »
URL »
block
int
k
n
mySystem
X
Y
amp
function
mult
delay

DrawElement

`DrawElement[`*elementSpec*`]`
DrawElement creates a list of graphics specifications from which to draw an element.

elementSpec is an element specification. Element specification is a list of the form {"*name*", *coordinates*, *value*, *label*, *elementOpts*}.

name is the name of an element: Adder, Amplifier, Arrow, Block, Delay, Function, Input, Integrator, Line, Modulator, Multiplier, Node, Output, Polyline, or Text. *name* should be enclosed within double quotation marks.

coordinates is a pair of numbers {*x*,*y*} or a list of pairs of numbers {{*x*1,*y*1}, {*x*2,*y*2}, ...}.

value is an expression that is the element value (e.g., the multiplier coefficient).

label is a string or expression to annotate the element.

The Arrow and Text elements do not have the label item.

DrawElement returns {} for unexpected arguments or element specifications.

MORE INFORMATION

- **Options[DrawElement]** gives a list of the current default settings for all options.
- **ElementScale** is an option for ShowSchematic that specifies the magnification of element dimensions for all schematic elements.
  ElementSize is an option for element specification that controls the size and aspect ratio of a schematic element.
  PolylineDashing is an option for the Polyline-element specification that controls dashing.
  ShowArrowTail is an option for the Arrow-element specification that controls the appearance of the arrow tail.
  ShowNodes is an option for element specification that controls the appearance of element nodes.
  TextDirection is an option for the Text-element specification that controls the angle of the text rotation.
  TextOffset is an option for element specification that controls the position of the element value and label.
- **PlotStyle** specifies the style of lines and points to be plotted. Two specifications are given: one for the element shape
- Open the **Palettes** menu, and choose the **SchematicOptions** command to open the palette Schematic Options.

EXAMPLES

Basic Examples (1)

```
In[1]:= Get["SchematicSolver`"]
```

Here is the schematic specification of a system:

```
In[2]:= mySchematic = {
 {"Adder", {{0, 2}, {1, 1}, {4, 2}, {1, 3}}, {1, -1, 2, 1}, " "},
 {"Amplifier", {{0, 4}, {4, 4}}, A, "amp"}, {"Arrow", {{4, 7}, {0, 7}}},
 {"Block", {{0, 9}, {4, 9}}, G, "block"}, {"Delay", {{5, 1}, {10, 1}}, 1, "delay"},
 {"Function", {{5, 4}, {10, 4}}, F, "function"}, {"Input", {7, 7}, X, "", TextOffset → {1, 0}},
 {"Output", {9, 7}, Y, "", TextOffset → {-1, 0}}, {"Integrator", {{5, 9}, {10, 9}}, k, "int"},
 {"Modulator", {{11, 2}, {12, 1}, {15, 2}, {12, 3}}, {1, 1, 2, 1}, " "},{"Multiplier",
    {{11, 4}, {15, 4}}, A, "mult"}, {"Line", {{11, 7}, {15, 7}}}, {"Node", {12, 9}, n, ""},
 {"Text", {14, 9}, "mySystem"}, {"Polyline", {{-1, 0}, {-1, 11}, {16, 11}, {16, 0}, {-1, 0}}}};
 ShowSchematic[%];
```

Fig. 4. Sample reference notebook.

New template for guide file is generated using the function **NewGuide[.]**, while new template for tutorial notebook is a result of the function **NewTutorial[.]**. All files that are generated (guides, tutorials, and functions) are editable, but after building the applications with **BuildApplication[.]**, all documentation files (guides, tutorials, and functions) are non-editable.

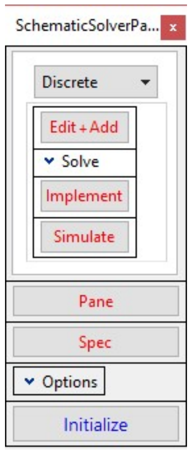
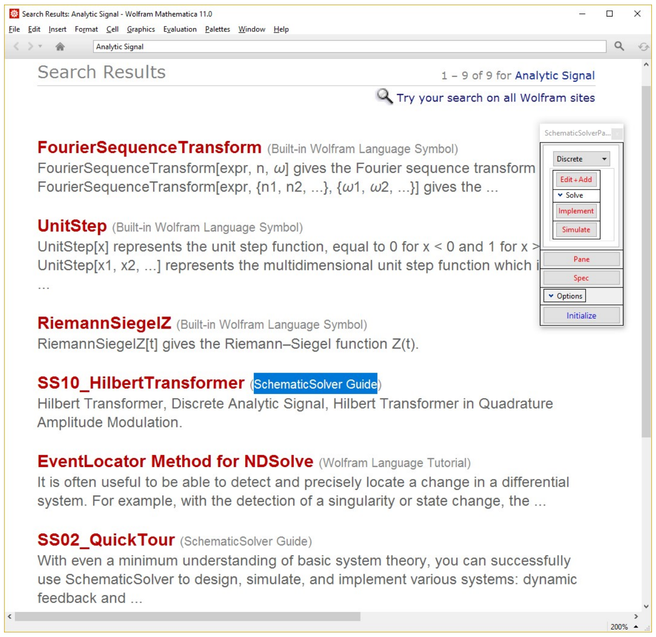


Fig. 5. SchematicSolver palette.


Search Results: Analytic Signal - Wolfram Mathematica 11.0
File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
Analytic Signal
SchematicSolverPa...
Discrete
Edit+Add
Solve
Implement
Simulate
Pane
Spec
Options
Initialize
200%

Search Results

1 – 9 of 9 for Analytic Signal

Try your search on all Wolfram sites

FourierSequenceTransform (Built-in Wolfram Language Symbol)
FourierSequenceTransform[expr, n, ω] gives the Fourier sequence transform
FourierSequenceTransform[expr, {n1, n2, ...}, {ω1, ω2, ...}] gives the ...

UnitStep (Built-in Wolfram Language Symbol)
UnitStep[x] represents the unit step function, equal to 0 for x < 0 and 1 for x >
UnitStep[x1, x2, ...] represents the multidimensional unit step function which i
...

RiemannSiegelZ (Built-in Wolfram Language Symbol)
RiemannSiegelZ[t] gives the Riemann–Siegel function Z(t).

SS10_HilbertTransformer (SchematicSolver Guide)
Hilbert Transformer, Discrete Analytic Signal, Hilbert Transformer in Quadrature Amplitude Modulation.

EventLocator Method for NDSolve (Wolfram Language Tutorial)
It is often useful to be able to detect and precisely locate a change in a differential system. For example, with the detection of a singularity or state change, the ...

SS02_QuickTour (SchematicSolver Guide)
With even a minimum understanding of basic system theory, you can successfully use SchematicSolver to design, simulate, and implement various systems: dynamic feedback and ...

Fig. 6. Search results for *analytic signal*.

At final stage, some textual description should be changed in newest version of Mathematica in the file **PacletInfo.*m***. The **Context** and **MainPage** should be changed according to what additional knowledge we would like to add into the current version of SchematicSolver.

B. *Palette Notebook*

Palettes can be used to perform many actions in the Wolfram System front end, including drawing systems, solving systems, and preparing reports. After saving palette into appropriate folder we can use it as a keyboard, as it is illustrated in Fig. 5 for Discrete-time systems. Palettes for the design electric circuits were presented in [4].

C. *Documentation center*

The purpose of this paper is to demonstrate that a new application developed to work within Mathematica is integrated in the Documentation center of this Computer Algebra System (CAS). When Mathematica is started, we can search for some terms, for example *analytic signal*. Search results are presented in Fig. 6. The searched term is in several built-in Wolfram Language symbols, Wolfram language tutorials, but also in SchematicSolver guides. This proves that the application SchematicSolver is integrated into Wolfram documentation center and all functions and usages are visible in a same manner as built-in commands.

The whole procedure was tested on desktop computers, but also on Raspberry Pi [5].

V. Conclusion

Application SchematicSolver is integrated into Computer Algebra System (CAS) as third-party product. The proposed methodology is based only on Wolfram language without usage of specific development environment. This paper can help everyone who is using Wolfram language to integrate his work into Mathematica using basic functions, in such a way that developed new functions are available as built-in functions.

Acknowledgment

This work was supported by the Ministry of Education and Science of Serbia under Grant TR 32023.

References

[1] S. Wolfram, *An elementary introduction to the Wolfram Language*, Champaign, IL, USA: Wolfram Media, 2015.

[2] M. D. Lutovac and A. M. Lutovac, "Vickrey Auction Game as Mathematica Application," Proc. 22nd Telecommunications forum TELFOR 2014, Belgrade, Serbia, pp. 1023-1026, Nov. 25-27, 2014.

[3] M. Lutovac, D. Tošic, *SchematicSolver Version 2.3,* 2014, http://www.wolfram.com/products/applications/schematicsolver/.

[4] M. D. Lutovac, V. Mladenović, M. Lutovac-Banduka, "Graphical user interface for electrical engineering systems using Wolfram Language," Telecommunications Forum (TELFOR), Belgrade, Serbia, November 2016, pp. 9.39.1–4.

[5] C. Severance, "Eben Upton: Raspberry Pi," *IEEE Computer*, vol. 46*,* no. 10*,* pp. 14–16, 2013.