Manual Action.NET → OpenDSS Application

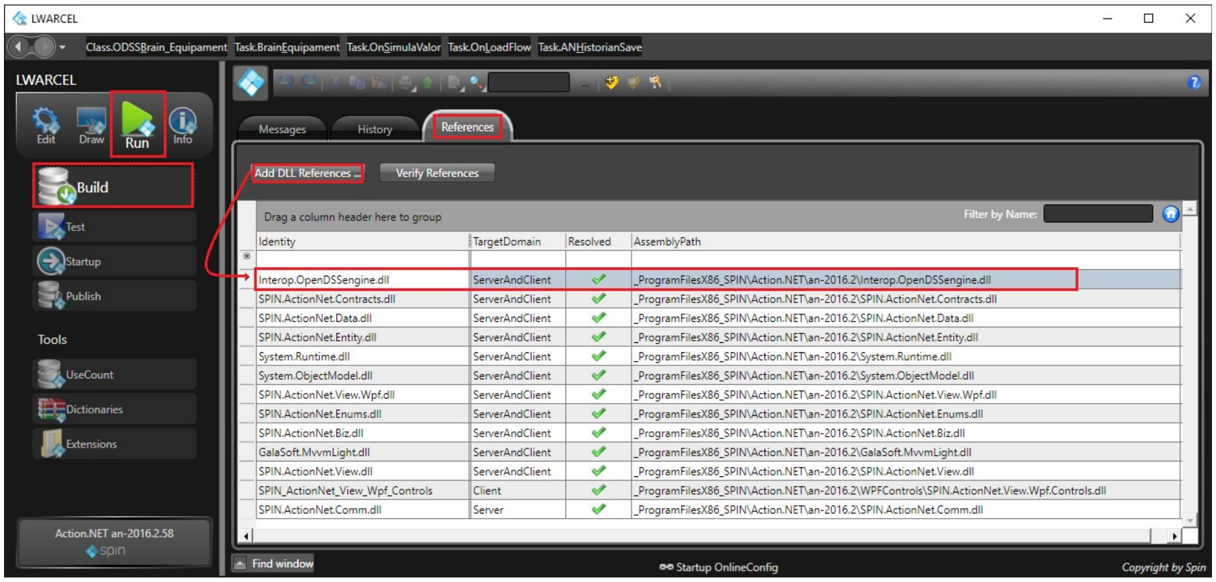
1. Integration with Action.NET

OpenDSS provides the possibility to develop external functionalities using the COM (Common Object Module) interface, through numerous programming languages, such as C#.

The integration is possible through OpenDSSengine.dll composed of classes and methods capable of controlling OpenDSS.

For integration with Action.NET the Interop.OpenDSSengine.dll was developed. It contains all classes and methods of the native OpenDSS dll, and we can implement the same example available in the OpenDSS manual directly in Action.NET.

To use the dll just go to Run → Build → References, click Add DLL References, and search for the "dll" in *C:\Program Files (x86)\SPIN\Action.NET\an-2016.2*.

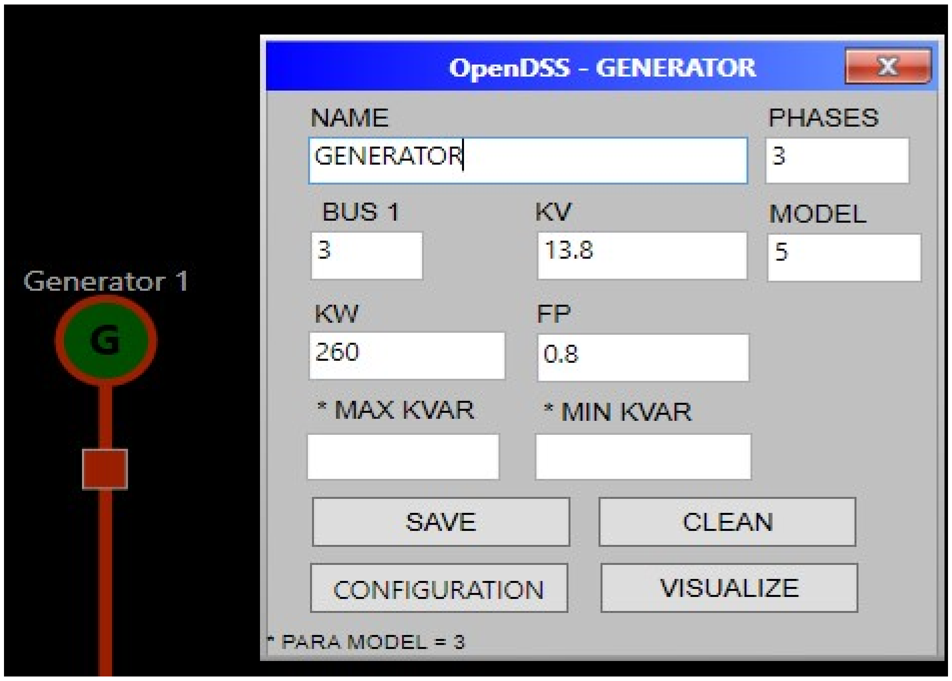


Identity	TargetDomain	Resolved	AssemblyPath
Interop.OpenDSSengine.dll	ServerAndClient	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\Interop.OpenDSSengine.dll
SPIN.ActionNet.Contracts.dll	ServerAndClient	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\SPIN.ActionNet.Contracts.dll
SPIN.ActionNet.Data.dll	ServerAndClient	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\SPIN.ActionNet.Data.dll
SPIN.ActionNet.Entity.dll	ServerAndClient	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\SPIN.ActionNet.Entity.dll
System.Runtime.dll	ServerAndClient	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\System.Runtime.dll
System.ObjectModel.dll	ServerAndClient	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\System.ObjectModel.dll
SPIN.ActionNet.View.Wpf.dll	ServerAndClient	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\SPIN.ActionNet.View.Wpf.dll
SPIN.ActionNet.Enums.dll	ServerAndClient	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\SPIN.ActionNet.Enums.dll
SPIN.ActionNet.Biz.dll	ServerAndClient	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\SPIN.ActionNet.Biz.dll
GalaSoft.MvvmLight.dll	ServerAndClient	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\GalaSoft.MvvmLight.dll
SPIN.ActionNet.View.dll	ServerAndClient	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\SPIN.ActionNet.View.dll
SPIN_ActionNet_View_Wpf_Controls	Client	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\WPFControls\SPIN.ActionNet.View.Wpf.Controls.dll
SPIN.ActionNet.Comm.dll	Server	✓	_ProgramFilesX86_SPIN\Action.NET\an-2016.2\SPIN.ActionNet.Comm.dll

Image 1: Adding the dll on Action.NET.

In order to facilitate the integration between OpenDSS and Action.NET, a basic application composed (DefaultNewProject) of Symbols, Displays and Scripts containing the integration logics was developed, all you have to do is simply model your electrical system without worrying about C# code.

Image 2 exemplifies the Symbol Generator and its configuration screen, where the user should enter the generator data.



NAME		PHASES
GENERATOR		3
BUS 1	KV	MODEL
3	13.8	5
KW	FP	
260	0.8	
* MAX KVAR	* MIN KVAR	

* PARA MODEL = 3

Image 2: Generator symbol and configuration screen.

This sample has five objects: **Load**, **Generator**, **Transformer**, **Line** and **Circuit** and for each object there are a display, like image 2, where the user can enter with the attributes of the object. For each Display, there are a symbol associated. All symbols are grouped in the Load_flow symbol Library.

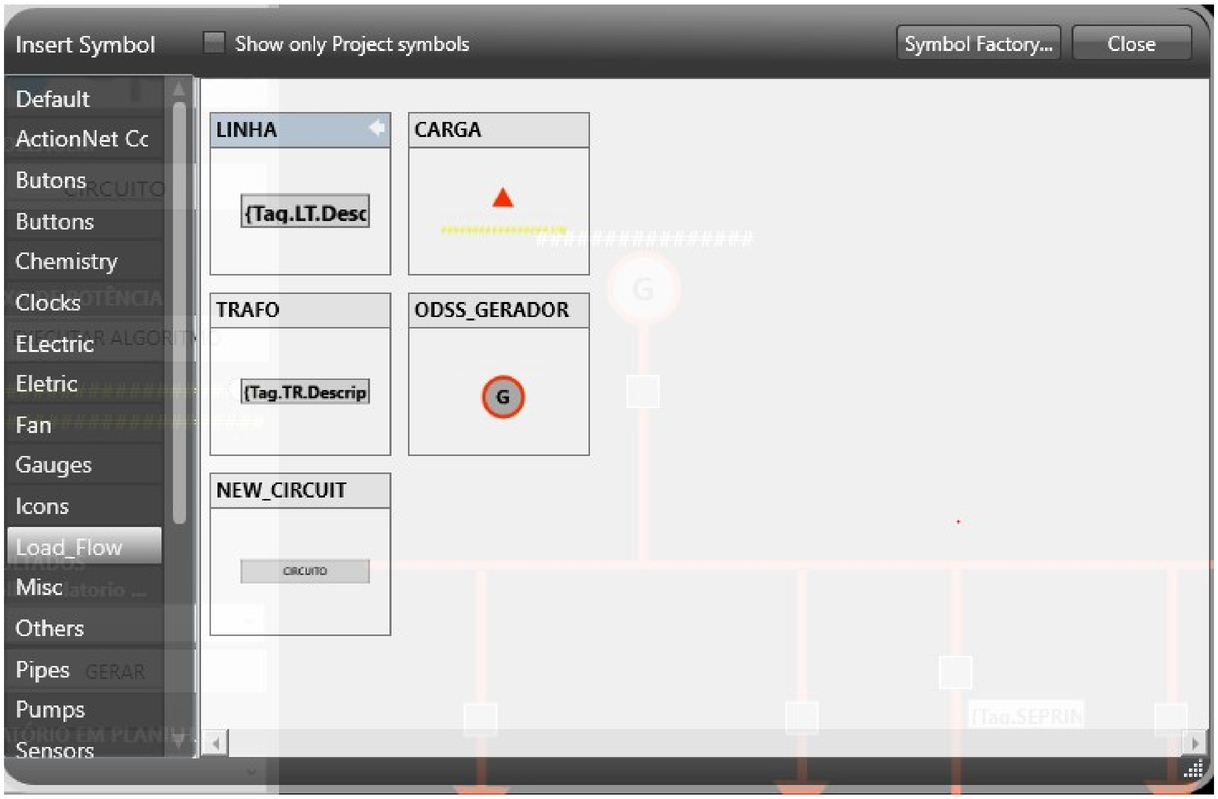
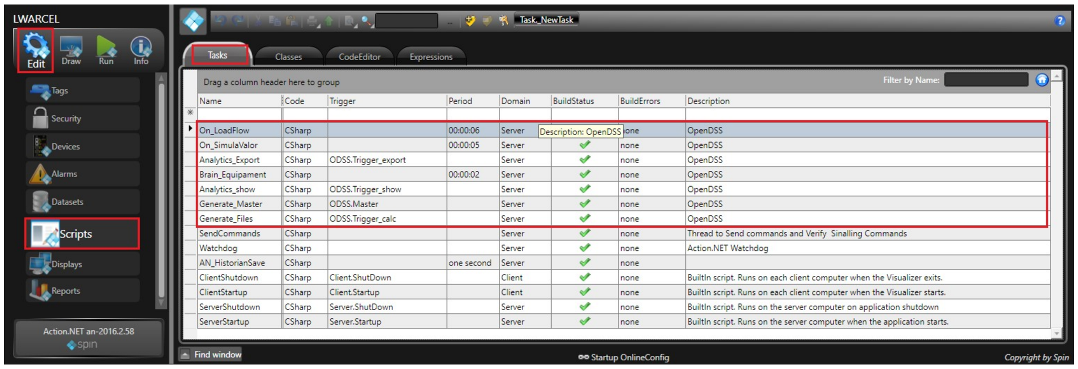


Image 3: Load Flow symbol library.

The scripts used for assembling the topology and creating the necessary files are:

Scripts written as Tasks:

- On_LoadFlow;
- On_Simulator;
- Analytics_Export.
- Brain_Equipament.
- Analytics_show.
- Generate_Master.
- Generate_Files.

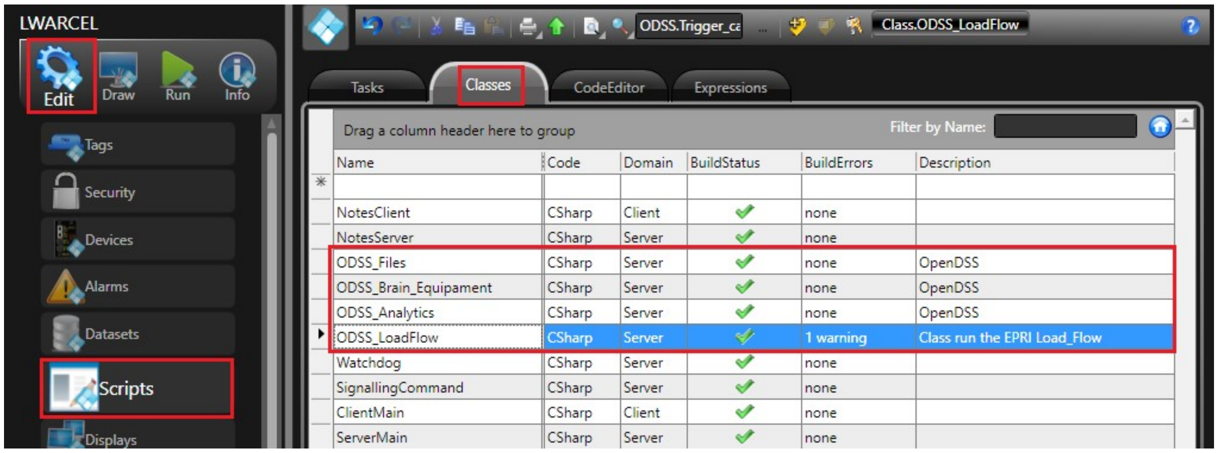


Name	Code	Trigger	Period	Domain	BuildStatus	BuildErrors	Description
On_LoadFlow	CSharp		00:00:06	Server	Description: OpenDSS	none	OpenDSS
On_SimulaValor	CSharp		00:00:05	Server	✓	none	OpenDSS
Analytics_Export	CSharp	ODSS.Trigger_export		Server	✓	none	OpenDSS
Brain_Equipament	CSharp		00:00:02	Server	✓	none	OpenDSS
Analytics_show	CSharp	ODSS.Trigger_show		Server	✓	none	OpenDSS
Generate_Master	CSharp	ODSS.Master		Server	✓	none	OpenDSS
Generate_Files	CSharp	ODSS.Trigger_calc		Server	✓	none	OpenDSS
SendCommands	CSharp			Server	✓	none	Thread to Send commands and Verify Sinalling Commands
Watchdog	CSharp			Server	✓	none	Action.NET Watchdog
AN_HistorianSave	CSharp		one second	Server	✓	none	
ClientShutdown	CSharp	Client.ShutDown		Client	✓	none	BuiltIn script. Runs on each client computer when the Visualizer exits.
ClientStartup	CSharp	Client.Startup		Client	✓	none	BuiltIn script. Runs on each client computer when the Visualizer starts.
ServerShutdown	CSharp	Server.ShutDown		Server	✓	none	BuiltIn script. Runs on the server computer on application shutdown
ServerStartup	CSharp	Server.Startup		Server	✓	none	BuiltIn script. Runs on the server computer when the application starts.

Image 4: Scripts Tasks

Scripts written as Classes:

- ODSS_Files.
- ODSS_Brain_Equipament.
- ODSS_Analytics.
- ODSS_LoadFlow.



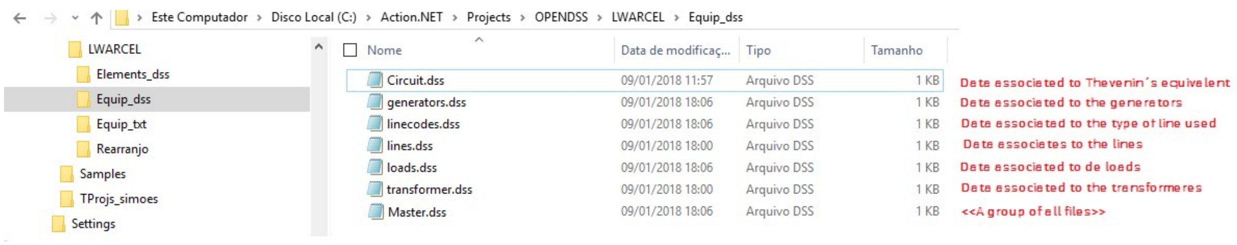
Name	Code	Domain	BuildStatus	BuildErrors	Description
NotesClient	CSharp	Client	✓	none	
NotesServer	CSharp	Server	✓	none	
ODSS_Files	CSharp	Server	✓	none	OpenDSS
ODSS_Brain_Equipament	CSharp	Server	✓	none	OpenDSS
ODSS_Analytics	CSharp	Server	✓	none	OpenDSS
ODSS_LoadFlow	CSharp	Server	✓	1 warning	Class run the EPRI Load_Flow
Watchdog	CSharp	Server	✓	none	
SignallingCommand	CSharp	Server	✓	none	
ClientMain	CSharp	Client	✓	none	
ServerMain	CSharp	Server	✓	none	

Image 5: Scripts Classes.

Other relevant information about this integration:

(1) At the application's startup 4 directories are created and they are responsible for housing different files:

- **Equip_txt**: For each object created is generated one txt file with the attributes of the object.
- **Rearrange**: When you turn-off some circuit breakers, you remove the downstream objects to the system and these objects are moved from the directory Equip.txt to the directory Rearrange. The load flow will consider only the objects existent in Equip_txt.
- **Equip_dss**: Contains all equip.txt objects in a file extension DSS that is used to present all attributes of the object when the user press configuration button of one individual object.
- **Elements_dss**: each time load flow run, it considers only the objects available in the circuit, that is, the equip.txt objects less the rearrange objects. In the element_dss directory, for each type of object there is one file that contains all objects of that type. The master file groups all objects. The image below show this directory.


Este Computador > Disco Local (C:) > Action.NET > Projects > OPENDSS > LWARCEL > Equip_dss
LWARCEL
Elements_dss
Equip_dss
Equip_txt
Rearranjo
Samples
TProjs_simoes
Settings

Nome	Data de modificaç...	Tipo	Tamanho	
Circuit.dss	09/01/2018 11:57	Arquivo DSS	1 KB	Data associated to Thevenin´s equivalent
generators.dss	09/01/2018 18:06	Arquivo DSS	1 KB	Data associated to the generators
linecodes.dss	09/01/2018 18:06	Arquivo DSS	1 KB	Data associated to the type of line used
lines.dss	09/01/2018 18:00	Arquivo DSS	1 KB	Data associetes to the lines
loads.dss	09/01/2018 18:06	Arquivo DSS	1 KB	Data associated to de loads
transformer.dss	09/01/2018 18:00	Arquivo DSS	1 KB	Data associated to the transformeres
Master.dss	09/01/2018 18:06	Arquivo DSS	1 KB	<<A group of all files>>

Image 6: Scripts Classes.

(2) All the circuit breakers are initialized closed (2).

(3) After the CIRCUIT (Thevenin Equivalent) configuration, two ".dss" files are generated on the folders *Equip_dss* and *Elements_dss*.

(4) The RUN ALGORITHM button generate, on the *Equip_dss folder*, the following files:

- linecodes.dss: all kinds of wiring elements existent in the electrical system.
- lines.dss: all the transmission lines existent in the electrical system.
- loads.dss: all the loads existent in the electrical system.
- transformer.dss: all the transformers existent in the electrical system.
- Master.dss: main file who calls the other files for the OpenDSS algorithm.

2.

Project Demo – LWARCEL

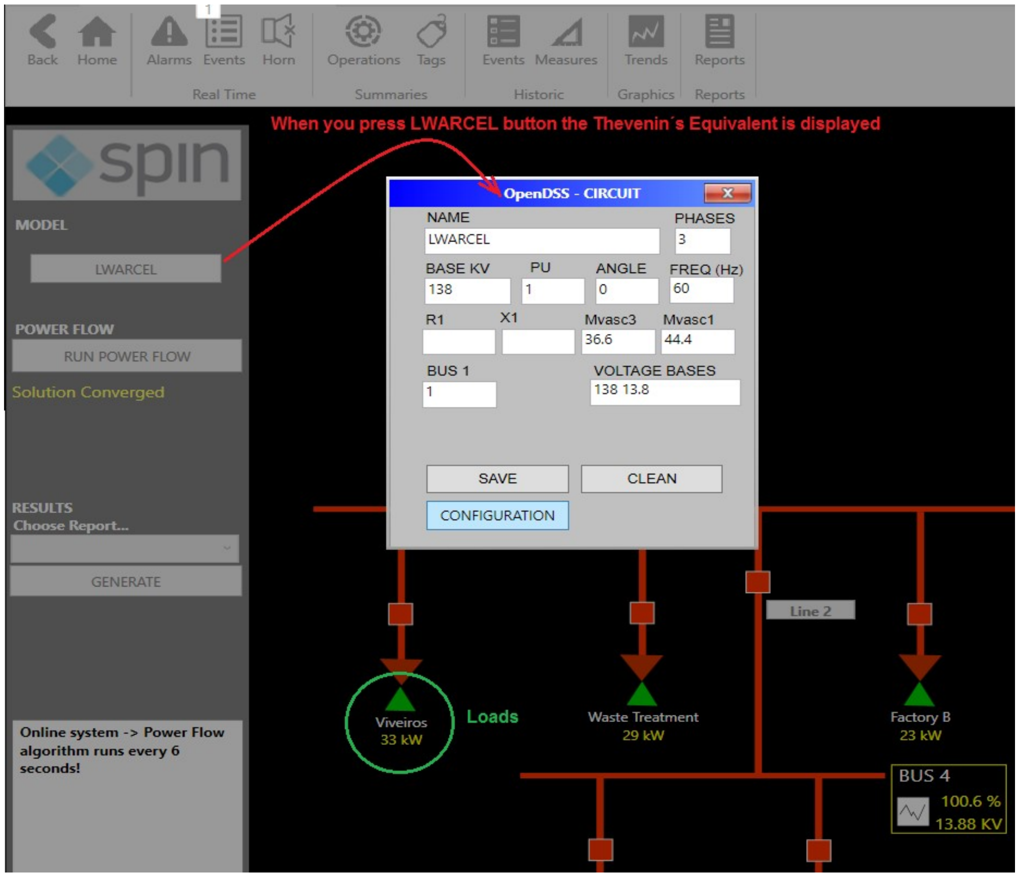
This Project was based in a real power plant from a pulp factory in São Paulo – Brazil.

In the Project, there are two simulation´s mode:

- Offline: In this mode, the power flow algorithm runs each time the RUN ALGORITHM button is pressed, and the results are based in the values pre-configured in the OpenDSS files.
- Online: In this mode, the power flow algorithm runs every 6 seconds and the results are based in the simulated field measurements.

2.1 Configuring the Elements of the Electrical System

In this Demo, we have five elements: The Thevenin’s Equivalent (Circuit), loads (11), lines (3), transformer (1) and generator (1).



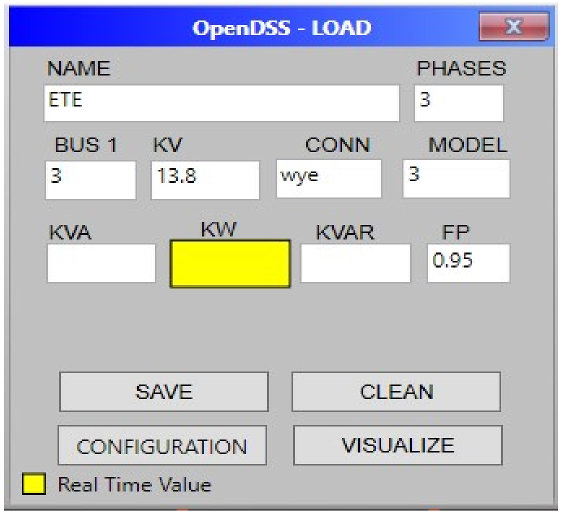
When you press LWARCEL button the Thevenin´s Equivalent is displayed

Solution Converged

Online system -> Power Flow algorithm runs every 6 seconds!

NAME	LWARCEL	PHASES	3
BASE KV	138	PU	1
ANGLE	0	FREQ (Hz)	60
R1		X1	
Mvasc3	36.6	Mvasc1	44.4
BUS 1	1	VOLTAGE BASES	138 13.8

Image 7: Thevenin’ s Equivalent configuration display.



OpenDSS - LOAD

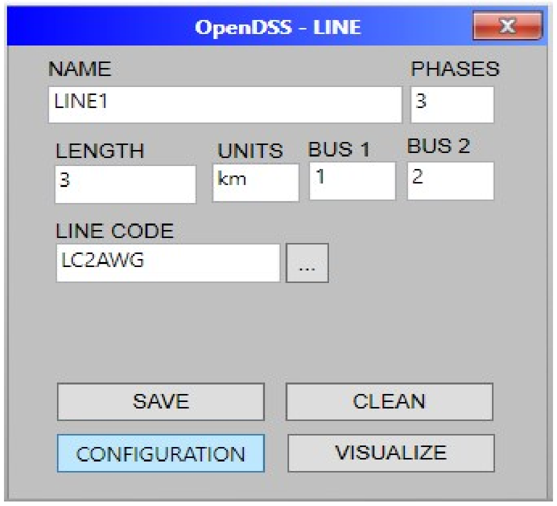
NAME	PHASES
ETE	3

BUS 1	KV	CONN	MODEL
3	13.8	wye	3

KVA	KW	KVAR	FP
	🟨		0.95

🟨 Real Time Value

Image 8: Load's configuration display.



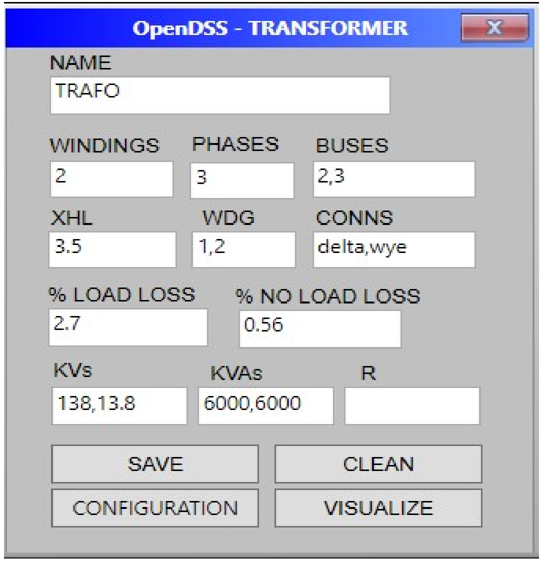
OpenDSS - LINE

NAME	PHASES
LINE1	3

LENGTH	UNITS	BUS 1	BUS 2
3	km	1	2

LINE CODE: LC2AWG

Image 9: Line's configuration display.



OpenDSS - TRANSFORMER

NAME: TRAFO

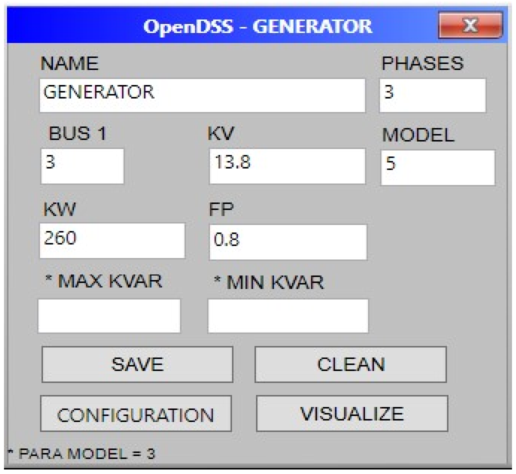
WINDINGS	PHASES	BUSES
2	3	2,3

XHL	WDG	CONNS
3.5	1,2	delta,wye

% LOAD LOSS	% NO LOAD LOSS
2.7	0.56

KVs	KVAs	R
138,13.8	6000,6000	

Image 10: Transformer's configuration display.


OpenDSS - GENERATOR

NAME: GENERATOR		PHASES: 3
BUS 1: 3	KV: 13.8	MODEL: 5
KW: 260	FP: 0.8	
* MAX KVAR:	* MIN KVAR:	

* PARA MODEL = 3

Image 11: Generator's configuration display.

2.2 Display Buttons

The display objects have four buttons:

(1) **Save**: Save the user´s attributes changes;

(2) **Clean**: Clean the attributes of the object;

(3) **Configuration**: Display all attributes of the object, i.e. not only the attributes of the window, but the ones generated by OpenDSS, as shown in image 13;

(4) **Visualize** Display three graphs: (1) current, (2) voltage and (3) power of the object, as presented in image 12. These graphs are an OpenDSS module.

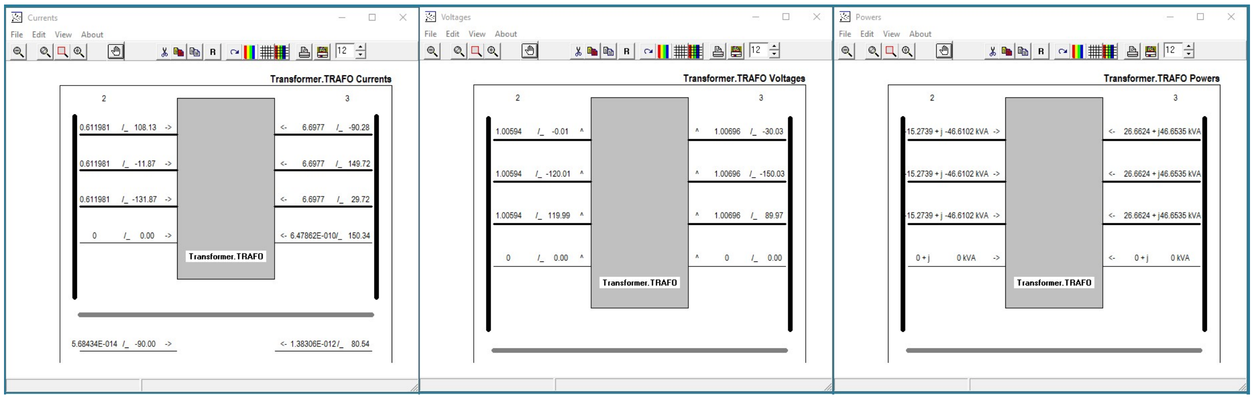
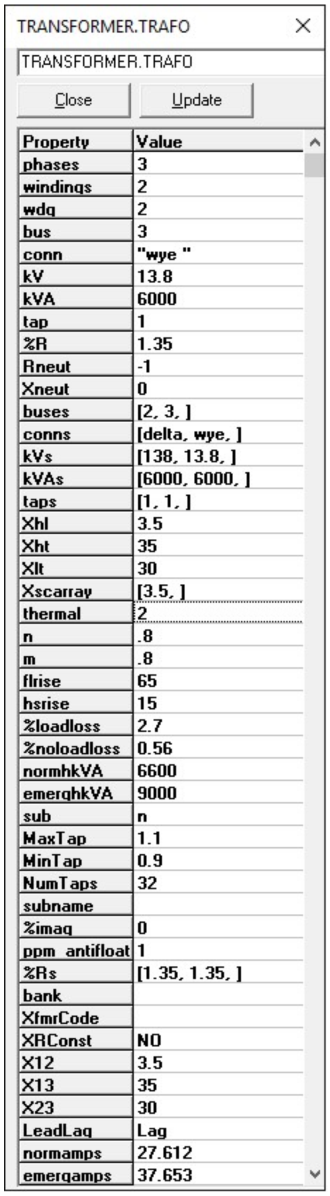


Image 12: Graphs with three-phase current, voltage and Power of transformer

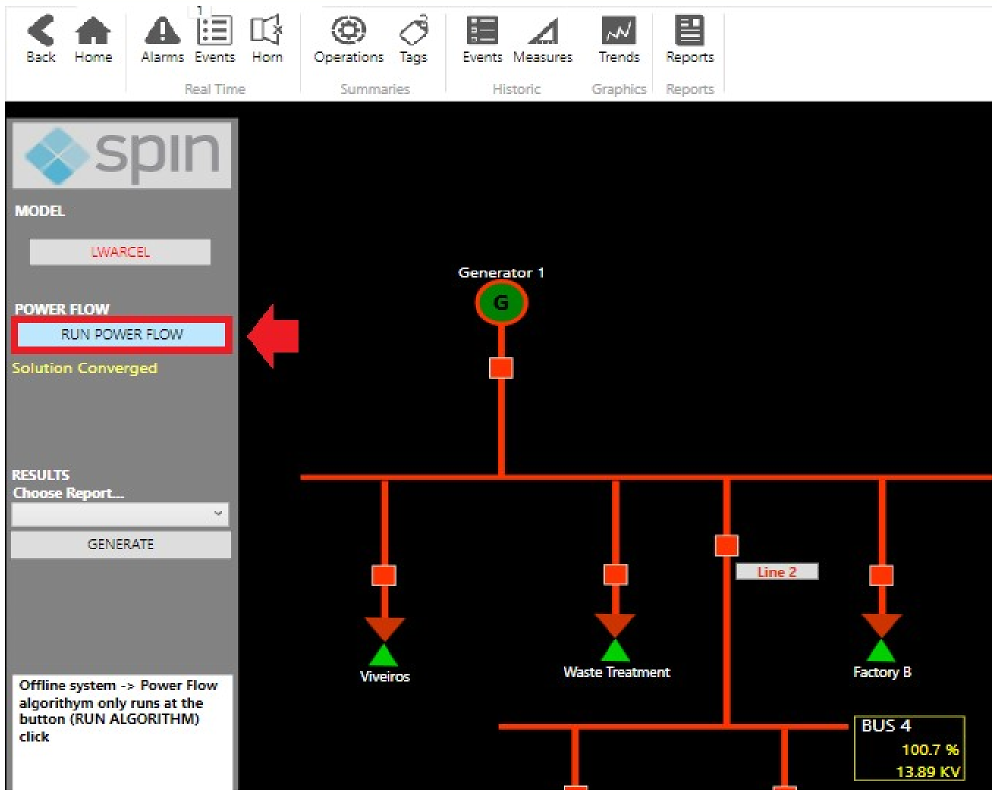


Property	Value
phases	3
windings	2
wdg	2
bus	3
conn	"wye "
kV	13.8
kVA	6000
tap	1
%R	1.35
Rneut	-1
Xneut	0
buses	[2, 3,]
conns	[delta, wye,]
kVs	[138, 13.8,]
kVAs	[6000, 6000,]
taps	[1, 1,]
Xhl	3.5
Xht	35
Xlt	30
Xscarray	[3.5,]
thermal	2
n	.8
m	.8
flrise	65
hsrise	15
%loadloss	2.7
%noloadloss	0.56
normhkVA	6600
emerghkVA	9000
sub	n
MaxTap	1.1
MinTap	0.9
NumTaps	32
subname	
%imag	0
ppm_antifloat	1
%Rs	[1.35, 1.35,]
bank	
XfmrCode	
XRConst	NO
X12	3.5
X13	35
X23	30
LeadLag	Lag
normamps	27.612
emergamps	37.653

Image 13: All Generator’s attributes

2.3 Off-Line Simulation

To run the power flow and get the voltage on the buses, you should click on the RUN POWER FLOW button.



Offline system -> Power Flow algorithym only runs at the button (RUN ALGORITHM) click

Image 14: Voltage and load percent.

After any modifications on the topology, or in the system's configuration you should click on the RUN ALGORITHM again.

The GENERATE button will create the files reports; you just need to choose which report will be created.

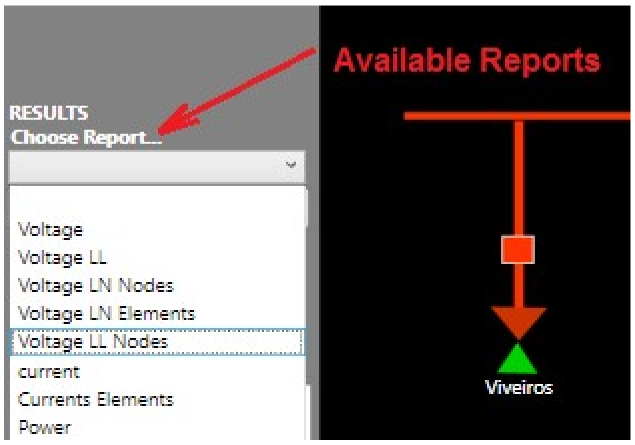
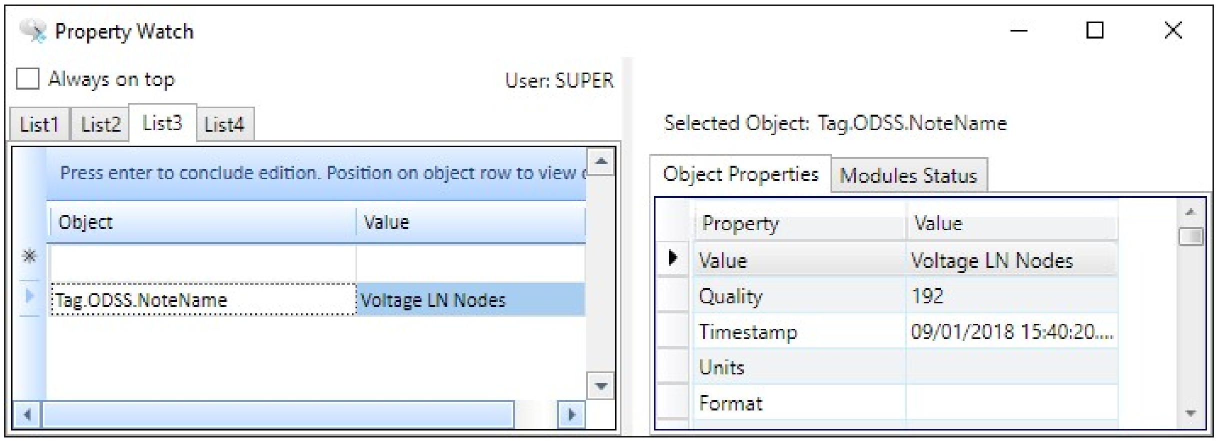


Image 15: Reports window.

When user press choose report, it´s opened a combo-box and the selected report is inserted in the variable Tag.ODSS.NoteName, as shown in image 16, and is activated the Script Class "ODSS_Analytics" that will generate the selected report in OpenDSS.


Property Watch

Always on top User: SUPER

List1 List2 List3 List4

Press enter to conclude edition. Position on object row to view

Object	Value
Tag.ODSS.NoteName	Voltage LN Nodes

Selected Object: Tag.ODSS.NoteName

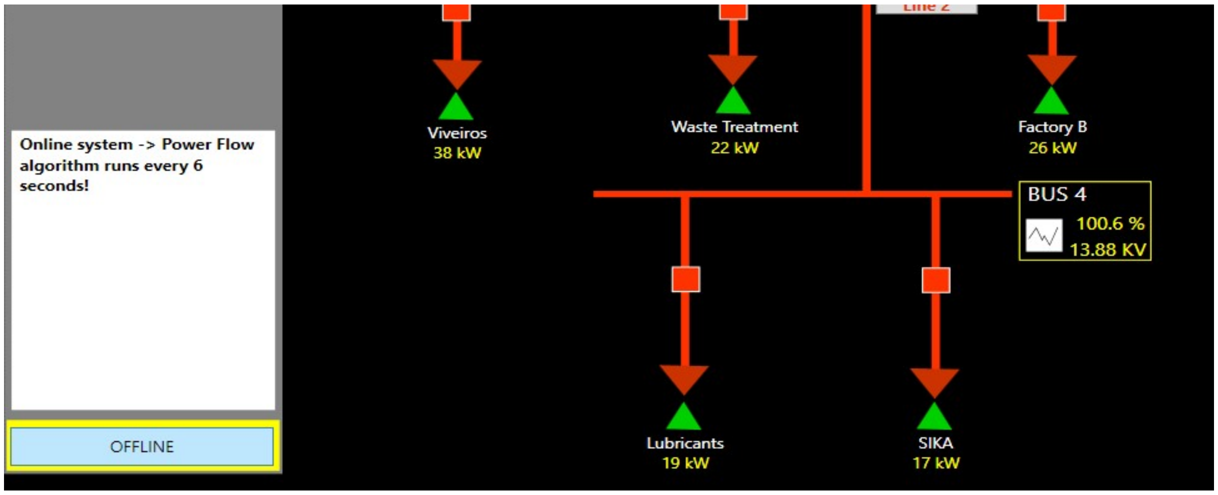
Object Properties Modules Status

Property	Value
Value	Voltage LN Nodes
Quality	192
Timestamp	09/01/2018 15:40:20....
Units	
Format	

Image 16: Reports name transferred to the variable Tag.ODSS.NoteName

2.4 Online Simulation

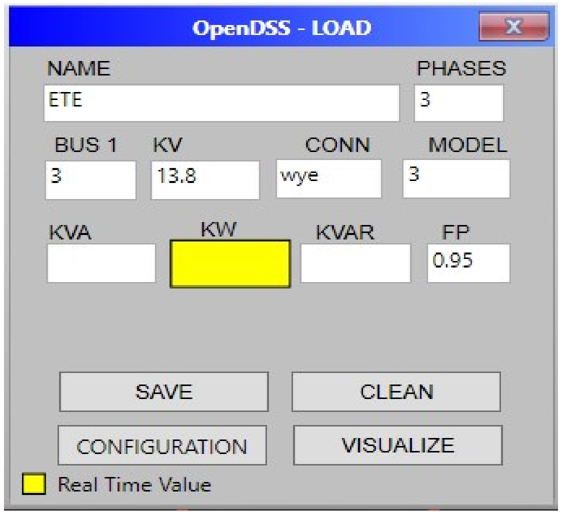
At every 6 seconds, the system runs the power flow algorithm using the field measurements in the calculation. All the offline functionalities are valid for the online simulation as well.



Online system -> Power Flow algorithm runs every 6 seconds!

Image 17: Online Simulation

When we open the load's configuration window, the load's power field is disabled.



OpenDSS - LOAD

NAME			PHASES
ETE			3

BUS 1	KV	CONN	MODEL
3	13.8	wye	3

KVA	KW	KVAR	FP
			0.95

Image 18: Online load's configuration display.

Other feature of this simulation mode is the voltage's graphic display at each bus. It is accessed by the following icon .

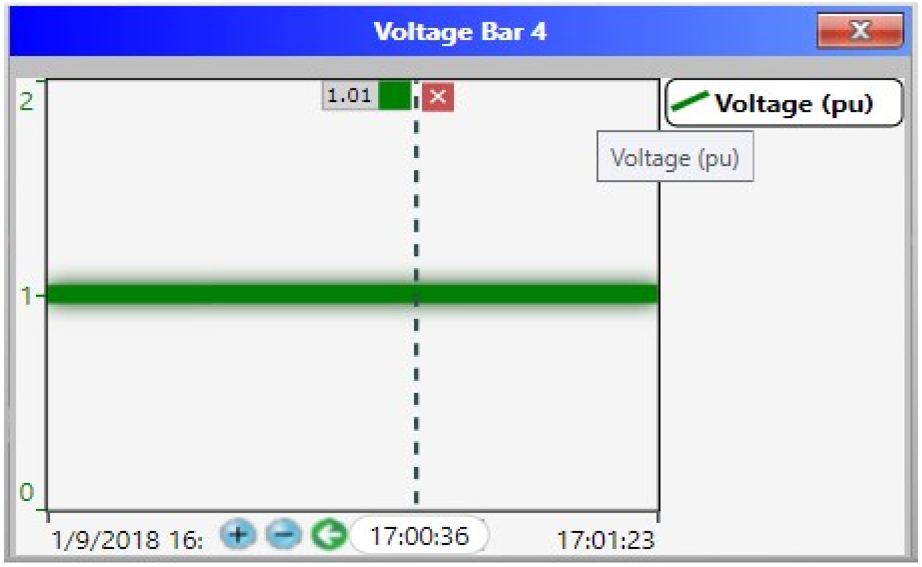


Image 19: Voltage's graphic display on the bus 4.

If you press the OFFLINE button, the system will go back to the offline simulation mode.

2.5 Project Philosophy

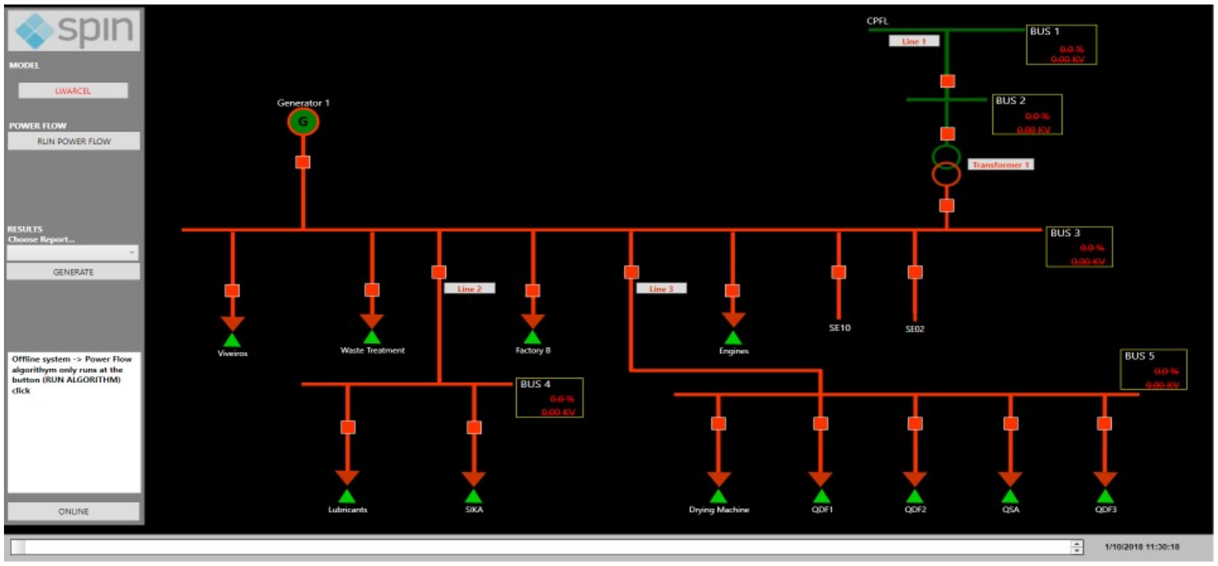
The purpose of this project was to verify the difficulty of integrating the OpenDSS library (DLL) into the Action.NET. Thus, a non-generic project was developed using a pulp substation. Once verified the effectiveness of the use of OpenDSS, a project will be created at Spin for generic use of OpenDSS integrated to SCADA.

3. Inserting a New Load at the Sample

Since this demo is not a generic application, let us show below what was designed and about this project how to add a new load. In this project, it has:

- 1 Equivalent of Thevenin representing the frontier
- 3 Lines (LT1, LT2 and LT3)
- 1 transformer (TR1)
- 13 loads (CR1 to CR13), where loads 5 and 6 are not used
- 1 Generator (GR1)
- 1 busbar table of size 20, where five BUS buses (1) are used up to Bus (5)

The Application configured as above is loaded at the beginning of the program as shown in the figure below:



Offline system -> Power Flow algorithym only runs at the button (RUN ALGORITHM) click

Image 20: Beginning of Application

If, for example, we will create the load 5 (NEWLOAD) in SE01 place of this fixed project, it will be necessary do the steps presented next:

3.1 At Design Time

We will edit the main screen (AN_MainPage) inserting the load symbol (CARGA) in the local of SE02, as shown in image 21. After that, we will associate this load to NEWLOAD (CR5), as shown in figure 22.

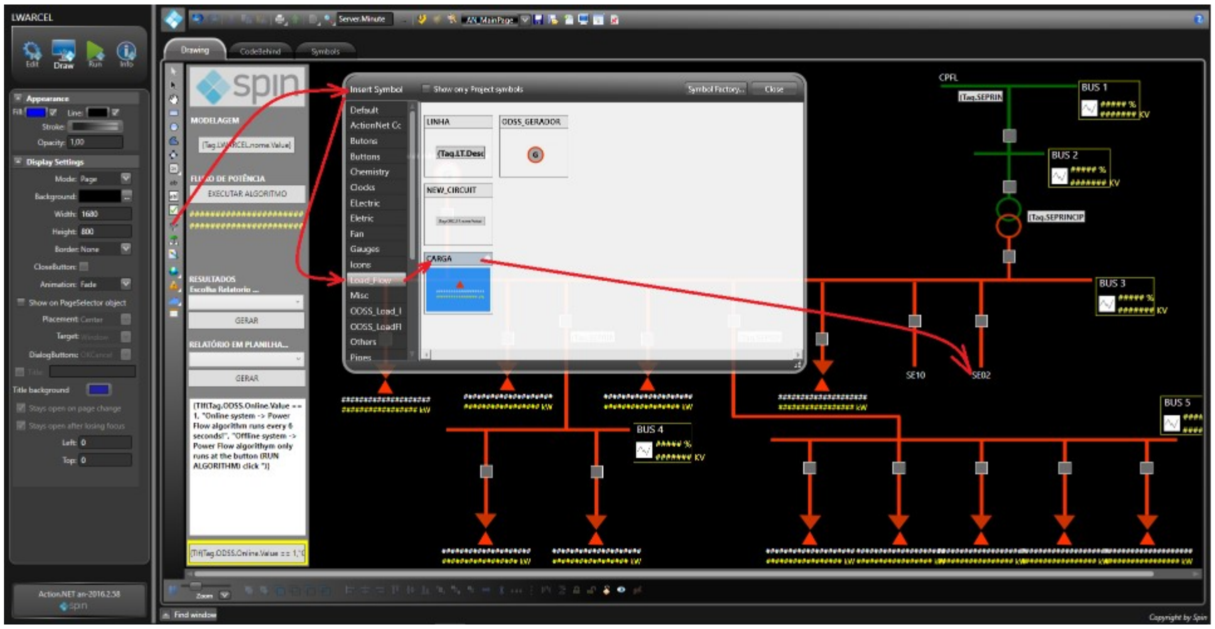


Image 21: Insert Load Symbol

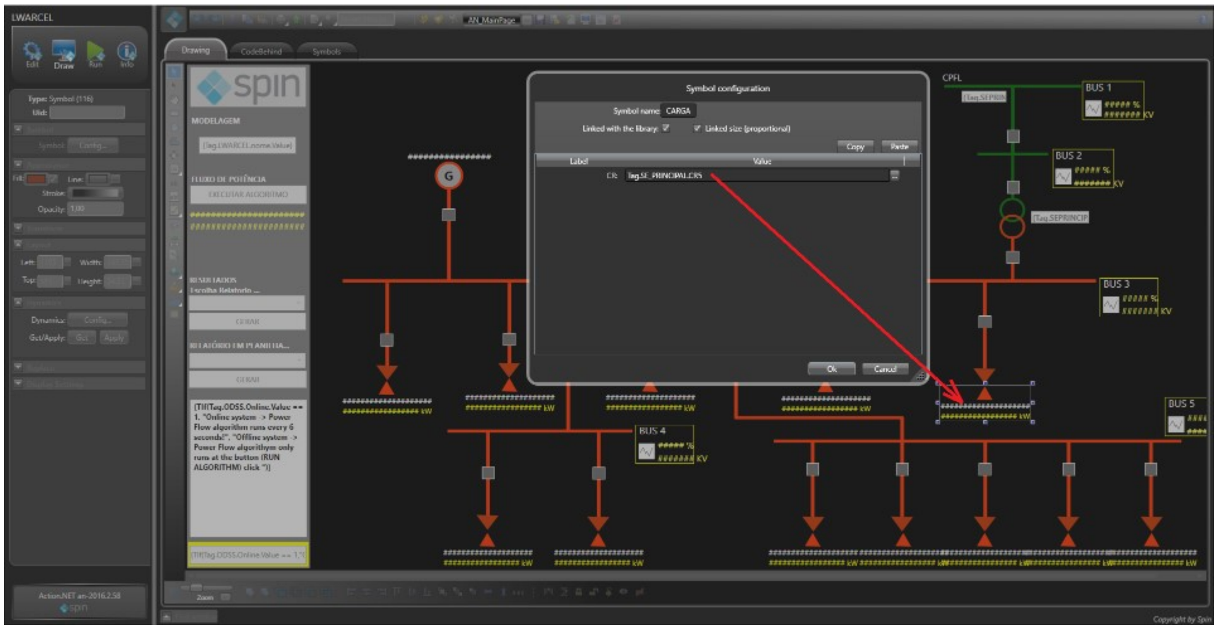
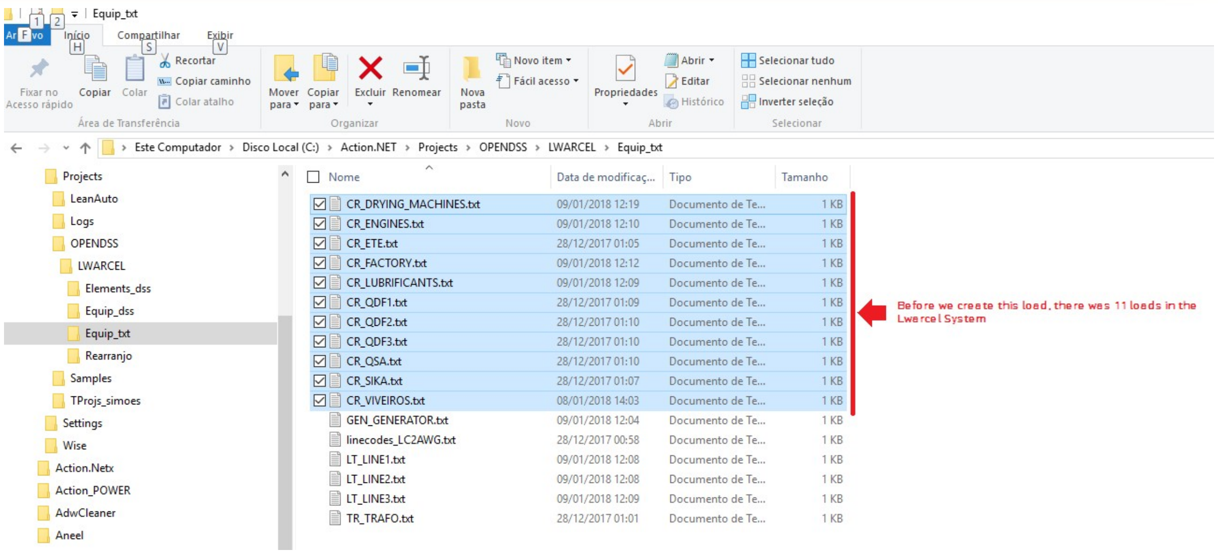


Image 22: Associate the NEWLOAD of Load5 (CR5)

As you can see below, in the directory Equip_txt, before we create the NEWLOAD, there are only 11 loads in the system.


Equip_txt

Este Computador > Disco Local (C:) > Action.NET > Projects > OPENDSS > LWARCEL > Equip_txt

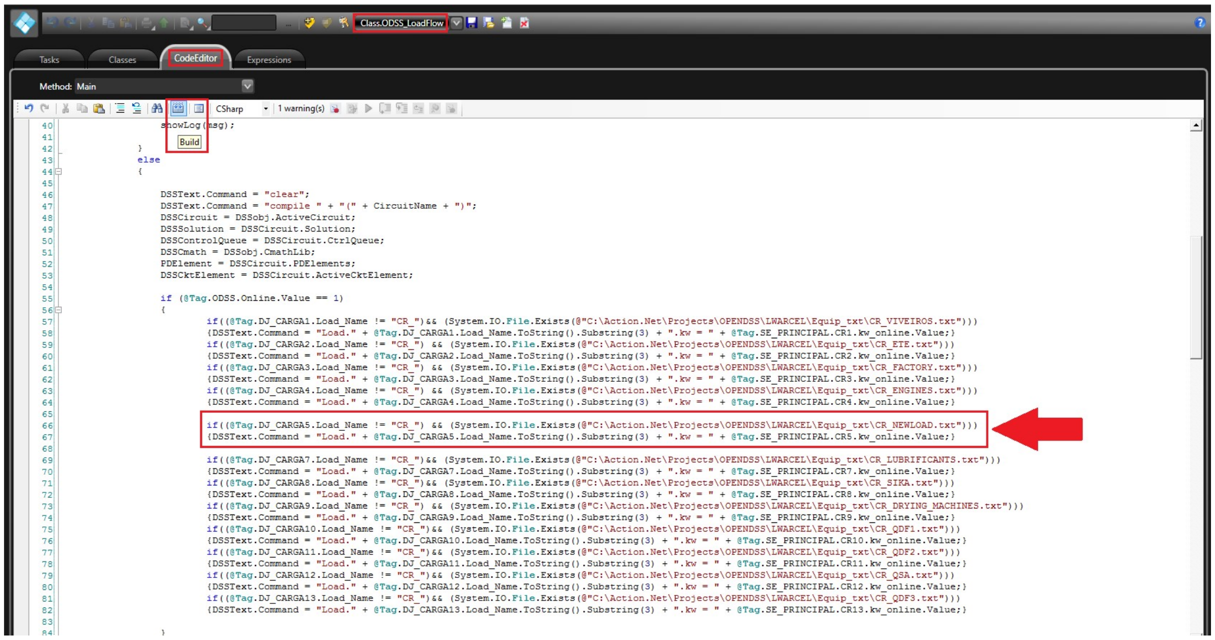
Nome	Data de modificaç...	Tipo	Tamanho
☑ CR_DRYING_MACHINES.txt	09/01/2018 12:19	Documento de Te...	1 KB
☑ CR_ENGINES.txt	09/01/2018 12:10	Documento de Te...	1 KB
☑ CR_ETE.txt	28/12/2017 01:05	Documento de Te...	1 KB
☑ CR_FACTORY.txt	09/01/2018 12:12	Documento de Te...	1 KB
☑ CR_LUBRIFICANTS.txt	09/01/2018 12:09	Documento de Te...	1 KB
☑ CR_QDF1.txt	28/12/2017 01:09	Documento de Te...	1 KB
☑ CR_QDF2.txt	28/12/2017 01:10	Documento de Te...	1 KB
☑ CR_QDF3.txt	28/12/2017 01:10	Documento de Te...	1 KB
☑ CR_QSA.txt	28/12/2017 01:10	Documento de Te...	1 KB
☑ CR_SIKA.txt	28/12/2017 01:07	Documento de Te...	1 KB
☑ CR_VIVEIROS.txt	08/01/2018 14:03	Documento de Te...	1 KB
GEN_GENERATOR.txt	09/01/2018 12:04	Documento de Te...	1 KB
linecodes_LC2AWG.txt	28/12/2017 00:58	Documento de Te...	1 KB
LT_LINE1.txt	09/01/2018 12:08	Documento de Te...	1 KB
LT_LINE2.txt	09/01/2018 12:08	Documento de Te...	1 KB
LT_LINE3.txt	09/01/2018 12:09	Documento de Te...	1 KB
TR_TRAFO.txt	28/12/2017 01:01	Documento de Te...	1 KB

Before we create this load, there was 11 loads in the Lwarcel System

Image 23: Loads (CR) before NEWLOAD

Once, as said before, the project was a non-flexible project, we will need to change some Scripts to insert the Newload.

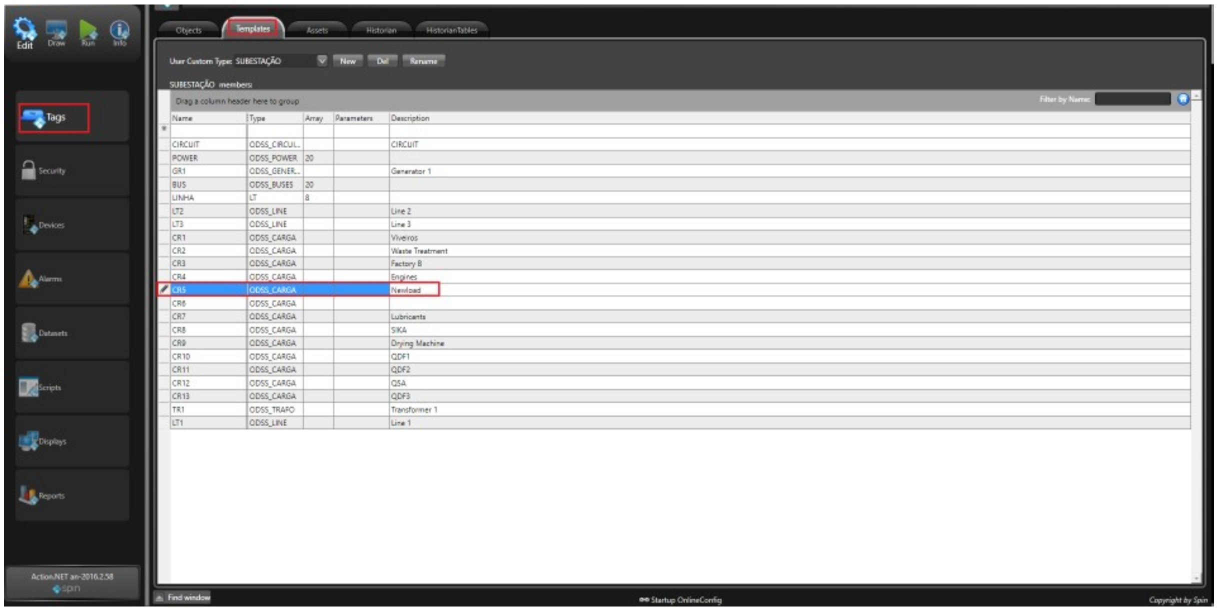
In the Script Class_ODSS_Loadflow we will insert a code, as showed to insert this load at the directory Equip_txt.



```
                    showLog(msg);

                }
                else
                {

                    DSSText.Command = "clear";
                    DSSText.Command = "compile " + "(" + CircuitName + ")";
                    DSSCircuit = DSSobj.ActiveCircuit;
                    DSSSolution = DSSCircuit.Solution;
                    DSSControlQueue = DSSCircuit.CtrlQueue;
                    DSSCmath = DSSobj.CmathLib;
                    PDElement = DSSCircuit.PDElements;
                    DSSCktElement = DSSCircuit.ActiveCktElement;

                    if (@Tag.ODSS.Online.Value == 1)
                    {
                            if((@Tag.DJ_CARGA1.Load_Name != "CR_")&& (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_VIVEIROS.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA1.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR1.kw_online.Value;}
                            if((@Tag.DJ_CARGA2.Load_Name != "CR_") && (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_ETE.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA2.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR2.kw_online.Value;}
                            if((@Tag.DJ_CARGA3.Load_Name != "CR_") && (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_FACTORY.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA3.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR3.kw_online.Value;}
                            if((@Tag.DJ_CARGA4.Load_Name != "CR_") && (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_ENGINES.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA4.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR4.kw_online.Value;}

                            if((@Tag.DJ_CARGA5.Load_Name != "CR_") && (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_NEWLOAD.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA5.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR5.kw_online.Value;}

                            if((@Tag.DJ_CARGA7.Load_Name != "CR_")&& (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_LUBRIFICANTS.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA7.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR7.kw_online.Value;}
                            if((@Tag.DJ_CARGA8.Load_Name != "CR_")&& (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_SIKA.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA8.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR8.kw_online.Value;}
                            if((@Tag.DJ_CARGA9.Load_Name != "CR_") && (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_DRYING_MACHINES.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA9.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR9.kw_online.Value;}
                            if((@Tag.DJ_CARGA10.Load_Name != "CR_")&& (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_QDF1.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA10.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR10.kw_online.Value;}
                            if((@Tag.DJ_CARGA11.Load_Name != "CR_")&& (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_QDF2.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA11.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR11.kw_online.Value;}
                            if((@Tag.DJ_CARGA12.Load_Name != "CR_")&& (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_QSA.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA12.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR12.kw_online.Value;}
                            if((@Tag.DJ_CARGA13.Load_Name != "CR_")&& (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDSS\LWARCEL\Equip_txt\CR_QDF3.txt")))
                            {DSSText.Command = "Load." + @Tag.DJ_CARGA13.Load_Name.ToString().Substring(3) + ".kw = " + @Tag.SE_PRINCIPAL.CR13.kw_online.Value;}

                    }
```

Image 24: Code changed in Class_ODSS_LoadFlow

The name of the load is loaded from the field description associated to the load, so we need to put the name "NEWLOAD" in the system (Substation), as shown below.



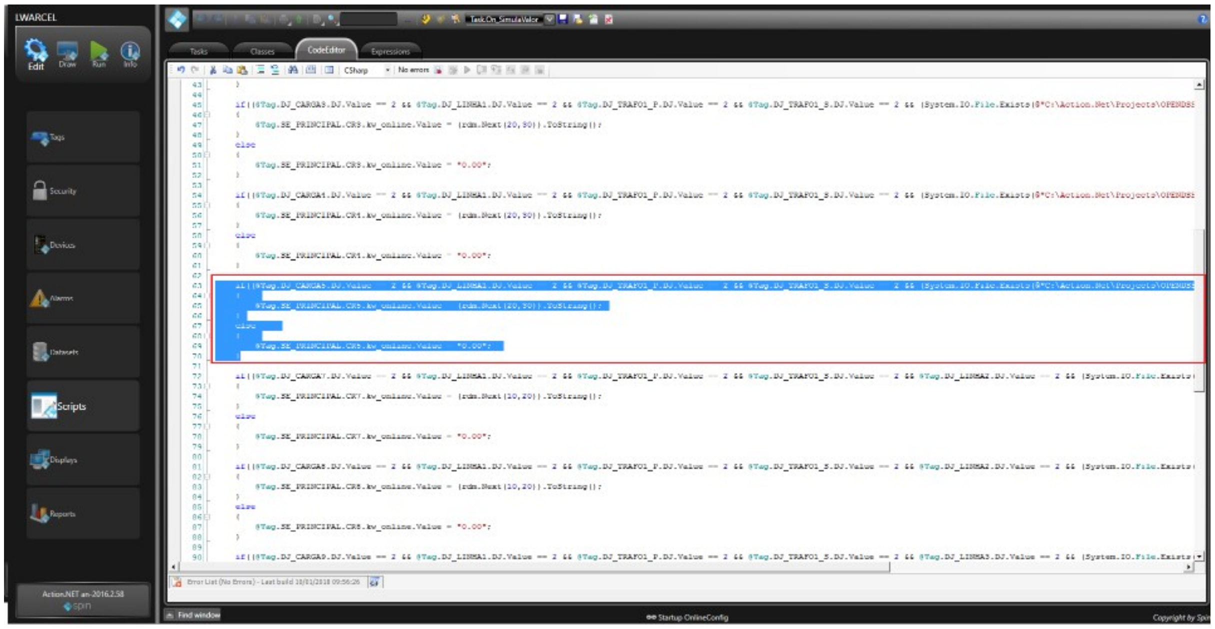
User Custom Type: SUBESTAÇÃO

SUBESTAÇÃO members:

Name	Type	Array	Parameters	Description
CIRCUIT	ODSS_CIRCUI...			CIRCUIT
POWER	ODSS_POWER	20		
GR1	ODSS_GENER...			Generator 1
BUS	ODSS_BUSES	20		
LINHA	LT	8		
LT2	ODSS_LINE			Line 2
LT3	ODSS_LINE			Line 3
CR1	ODSS_CARGA			Viveiros
CR2	ODSS_CARGA			Waste Treatment
CR3	ODSS_CARGA			Factory B
CR4	ODSS_CARGA			Engines
CR5	ODSS_CARGA			Newload
CR6	ODSS_CARGA			
CR7	ODSS_CARGA			Lubricants
CR8	ODSS_CARGA			SIKA
CR9	ODSS_CARGA			Drying Machine
CR10	ODSS_CARGA			QDF1
CR11	ODSS_CARGA			QDF2
CR12	ODSS_CARGA			Q5A
CR13	ODSS_CARGA			QDF3
TR1	ODSS_TRAFO			Transformer 1
LT1	ODSS_LINE			Line 1

Image 25: Description of CR5 ➔ “NEWLOAD”

Finally, at design time, we need to simulate the voltage value of the load5 (CR5) when the system is online. To do this we insert the code shown in the image below, in the Script Task.On_SimulaValor: “tag.SE_PRINCIPAL.CR5.kw_online.Value = “.



```
    }

    if((@Tag.DJ_CARGA3.DJ.Value == 2 && @Tag.DJ_LINHA1.DJ.Value == 2 && @Tag.DJ_TRAFO1_P.DJ.Value == 2 && @Tag.DJ_TRAFO1_S.DJ.Value == 2 && (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDS
    {
        @Tag.SE_PRINCIPAL.CR3.kw_online.Value = (rdm.Next(20,50)).ToString();
    }
    else
    {
        @Tag.SE_PRINCIPAL.CR3.kw_online.Value = "0.00";
    }

    if((@Tag.DJ_CARGA4.DJ.Value == 2 && @Tag.DJ_LINHA1.DJ.Value == 2 && @Tag.DJ_TRAFO1_P.DJ.Value == 2 && @Tag.DJ_TRAFO1_S.DJ.Value == 2 && (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDS
    {
        @Tag.SE_PRINCIPAL.CR4.kw_online.Value = (rdm.Next(20,50)).ToString();
    }
    else
    {
        @Tag.SE_PRINCIPAL.CR4.kw_online.Value = "0.00";
    }

    if((@Tag.DJ_CARGA5.DJ.Value == 2 && @Tag.DJ_LINHA1.DJ.Value == 2 && @Tag.DJ_TRAFO1_P.DJ.Value == 2 && @Tag.DJ_TRAFO1_S.DJ.Value == 2 && (System.IO.File.Exists(@"C:\Action.Net\Projects\OPENDS
    {
        @Tag.SE_PRINCIPAL.CR5.kw_online.Value = (rdm.Next(20,50)).ToString();
    }
    else
    {
        @Tag.SE_PRINCIPAL.CR5.kw_online.Value = "0.00";
    }

    if((@Tag.DJ_CARGA7.DJ.Value == 2 && @Tag.DJ_LINHA1.DJ.Value == 2 && @Tag.DJ_TRAFO1_P.DJ.Value == 2 && @Tag.DJ_TRAFO1_S.DJ.Value == 2 && @Tag.DJ_LINHA2.DJ.Value == 2 && (System.IO.File.Exists(
    {
        @Tag.SE_PRINCIPAL.CR7.kw_online.Value = (rdm.Next(10,20)).ToString();
    }
    else
    {
        @Tag.SE_PRINCIPAL.CR7.kw_online.Value = "0.00";
    }

    if((@Tag.DJ_CARGA8.DJ.Value == 2 && @Tag.DJ_LINHA1.DJ.Value == 2 && @Tag.DJ_TRAFO1_P.DJ.Value == 2 && @Tag.DJ_TRAFO1_S.DJ.Value == 2 && @Tag.DJ_LINHA2.DJ.Value == 2 && (System.IO.File.Exists(
    {
        @Tag.SE_PRINCIPAL.CR8.kw_online.Value = (rdm.Next(10,20)).ToString();
    }
    else
    {
        @Tag.SE_PRINCIPAL.CR8.kw_online.Value = "0.00";
    }

    if((@Tag.DJ_CARGA9.DJ.Value == 2 && @Tag.DJ_LINHA1.DJ.Value == 2 && @Tag.DJ_TRAFO1_P.DJ.Value == 2 && @Tag.DJ_TRAFO1_S.DJ.Value == 2 && @Tag.DJ_LINHA3.DJ.Value == 2 && (System.IO.File.Exists(
```

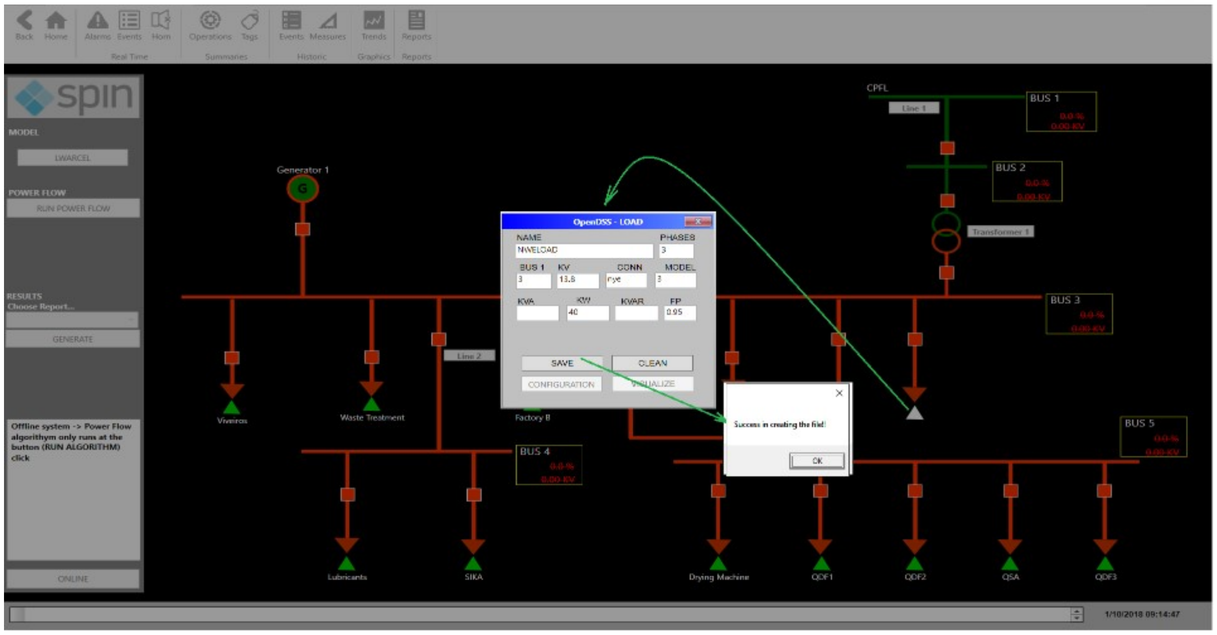
Error List (No Errors) - Last build 18/01/2018 09:56:26

Image 26: Simulate the KW of the Load, if it is connected

3.2 At Run Time

Just after start the application the user creates the Load5 (CR5) filling up the display window associated to this load.

Note: at start, the load is white, meaning it doesn´t have attributes.



Offline system -> Power Flow algorithym only runs at the button (RUN ALGORITHM) click

OpenDSS - LOAD

NAME			PHASES
NWELOAD			3
BUS 1	KV	CONN	MODEL
3	13.8	wye	3
KVA	KW	KVAR	FP
	40		0.95

Success in creating the file!

1/10/2018 09:14:47

Image 27: Create and save the load attributes

As soon as you create the NEWLOAD attributes, the system creates the file “CR_NEWLOAD.TXT” and “CR_NEWLOAD.dss” in the respective directory.

4.

Conclusions

The objective proposed in this application of OpenDSS was to present speed and computational efficiency of the software and this objective was reached.

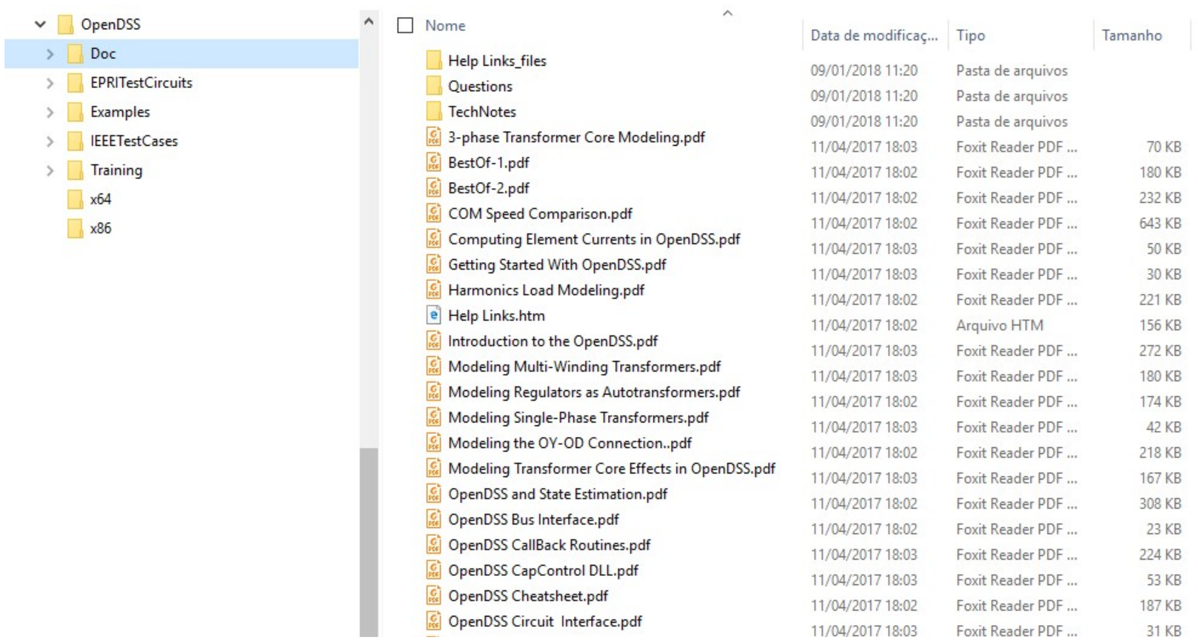
With this application, there are two possibilities of distribution integrated with SCADA:

(1) Provide the OpenDSS library integrated with SCADA and users with knowledge of the software and power system be able to develop their own applications in the same way this application was developed.

(2) Spin develops an initial application (DefaultNewProject) that already provides the user with a set of facilities to generate an application with features of Power Flow, contingency analysis, calculation of technical losses, OTS, etc.

Option (1) can be reached in a very short time by simply generating a DefaultNewProject already integrated into the "Interop.OpenDSSengine.dll" library. If on the one hand the use of OpenDSS requires the deep knowledge of power system and programming, on the other hand, its documentation is quite complete and vast. The image below shows the directory generated in the OpenDSS installation, and as can be seen there are:

- Extensive documentation;
- Technical Notes
- FAQs
- Several examples of EPRI and third parties
- Dozens of IEEE case studies
- Pre-ready training
- Website for online consultation
- Etc.



Nome	Data de modificaç...	Tipo	Tamanho
Help Links_files	09/01/2018 11:20	Pasta de arquivos	
Questions	09/01/2018 11:20	Pasta de arquivos	
TechNotes	09/01/2018 11:20	Pasta de arquivos	
3-phase Transformer Core Modeling.pdf	11/04/2017 18:03	Foxit Reader PDF ...	70 KB
BestOf-1.pdf	11/04/2017 18:02	Foxit Reader PDF ...	180 KB
BestOf-2.pdf	11/04/2017 18:02	Foxit Reader PDF ...	232 KB
COM Speed Comparison.pdf	11/04/2017 18:02	Foxit Reader PDF ...	643 KB
Computing Element Currents in OpenDSS.pdf	11/04/2017 18:03	Foxit Reader PDF ...	50 KB
Getting Started With OpenDSS.pdf	11/04/2017 18:03	Foxit Reader PDF ...	30 KB
Harmonics Load Modeling.pdf	11/04/2017 18:02	Foxit Reader PDF ...	221 KB
Help Links.htm	11/04/2017 18:02	Arquivo HTM	156 KB
Introduction to the OpenDSS.pdf	11/04/2017 18:03	Foxit Reader PDF ...	272 KB
Modeling Multi-Winding Transformers.pdf	11/04/2017 18:03	Foxit Reader PDF ...	180 KB
Modeling Regulators as Autotransformers.pdf	11/04/2017 18:02	Foxit Reader PDF ...	174 KB
Modeling Single-Phase Transformers.pdf	11/04/2017 18:03	Foxit Reader PDF ...	42 KB
Modeling the OY-OD Connection..pdf	11/04/2017 18:02	Foxit Reader PDF ...	218 KB
Modeling Transformer Core Effects in OpenDSS.pdf	11/04/2017 18:03	Foxit Reader PDF ...	167 KB
OpenDSS and State Estimation.pdf	11/04/2017 18:02	Foxit Reader PDF ...	308 KB
OpenDSS Bus Interface.pdf	11/04/2017 18:02	Foxit Reader PDF ...	23 KB
OpenDSS CallBack Routines.pdf	11/04/2017 18:03	Foxit Reader PDF ...	224 KB
OpenDSS CapControl DLL.pdf	11/04/2017 18:03	Foxit Reader PDF ...	53 KB
OpenDSS Cheatsheet.pdf	11/04/2017 18:02	Foxit Reader PDF ...	187 KB
OpenDSS Circuit Interface.pdf	11/04/2017 18:03	Foxit Reader PDF ...	31 KB

Image 28: OpenDSS directory Create after Installation

Option (2) can be reached in a few months, considering, for example, a defaultNewProject that has:

- A data structure that allows to create all the elements of a distribution system that has the main types of lines, loads, transformers, reactors, generators (hydraulic, solar, wind). On these elements, allow organizations of different types of electrical schemes contemplating the most modern practices of Smart Grid.
- The existence of a library of symbols contemplating all available elements thus with attribute windows already ready to contemplate all simulated elements.
- The existence of pre-ready reports and graphs that contemplate the information needs of the user;
- The implementation of ADMS advanced functions such as control Volt / Var, FLISR, OTS, OMS, etc.