Caneda Documentation

Release 0.2.0

Pablo Daniel Pareja Obregon

June 13, 2016

Contents

Caneda is an open source EDA software suite focused on ease of use and portability. Its main goal is to handle the complete design process: schematic capture, simulation and circuit layout or PCB. Caneda aims to support all kinds of circuit simulation types, e.g. DC, AC, S-parameter and harmonic balance analysis.

For more details, please visit Caneda Project Homepage.

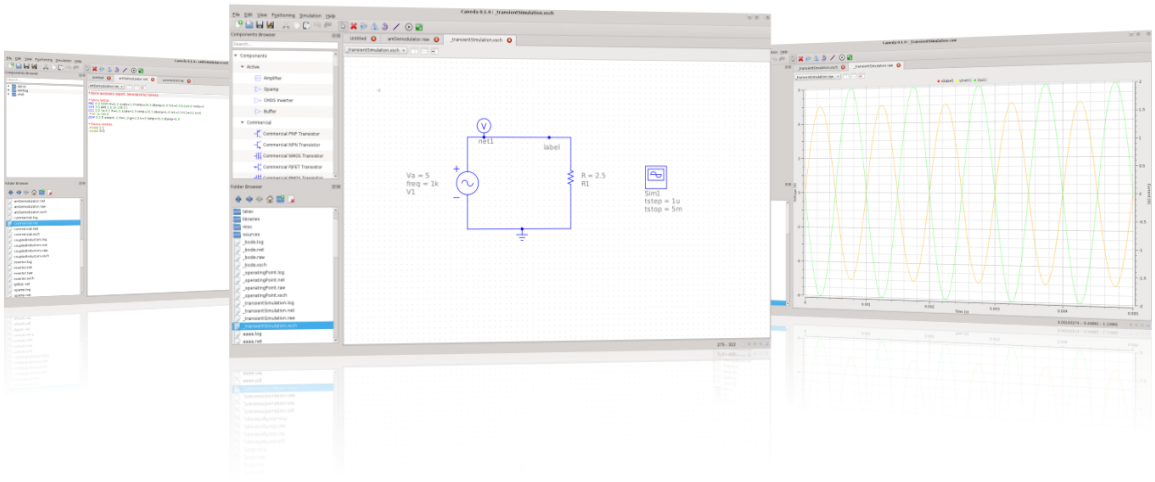


Table of Contents

1.1 Compilation

Note: This document is aimed to developers and advanced users. If you just want to start working with Caneda on your electronics projects, you may jump right into the *Installation* procedure or the *Getting Started* guide.

Caneda source code is hosted by GitHub. For more information, see Caneda on GitHub.

1.1.1 GNU/Linux

General

This section provides the basic guidelines to compile Caneda in GNU/Linux.

The distribution-specific compilation procedures are detailed later on this document, such as:

- *Debian 9 Stretch/Testing*
- *Debian 8 Jessie/Stable*

Note: To obtain the distribution name and release of a particular system, execute `lsb_release -da` from a system console.

Requirements

The following libraries must be installed on the system in order to compile Caneda:

- CMake
- Qt 5
- Qwt 6.1.2 or later

Although the package **Ngspice** is optional, it is strongly recommended to install it in order to add simulation capabilities to Caneda.

Source Code

Caneda source code repository can be locally cloned using **git** as follows:

```
$ git clone https://github.com/caneda/caneda
```

Note: The default Caneda distribution is *master* (stable). Execute `git checkout develop` to compile the *develop* (latest) distribution.

Alternatively, download either master or develop tarball from the GitHub repository and unpack it:

```
$ tar -xvf Caneda-Caneda-<version>.tar.gz
```

Compilation Steps

Create a new folder `build` at the top of Caneda source code structure and change into it:

```
$ cd caneda
$ mkdir build
$ cd build
```

Configure and compile the source package:

```
$ cmake ../
$ make
```

Finally, install Caneda into the system (requires root privileges):

```
# make install
```

Compiler Options

Some systems require special options for compilation or linking that CMake does not know about. Run `cmake --help` for details on some of the pertinent environment variables.

CMake can take initial values for configuration parameters by setting environment variables from the command line. Here is an example:

```
$ cmake CC=gcc-6 CXX=g++-6 CFLAGS=-O2 LIBS=-lposix ../
```

Installation Directory

By default, `make install` will install all the package files in `/usr/local/bin`, `/usr/local/man`, etc. You can specify an installation prefix other than `/usr/local` by giving *cmake* the option `--prefix=PATH`.

You can specify separate installation prefixes for architecture-specific files and architecture-independent files. If you give *cmake* the option `--exec-prefix=PATH`, the package will use *PATH* as the prefix for installing programs and libraries. Documentation and other data files will still use the regular prefix.

In addition, if you use a non-standard directory layout, you can give options like `--bindir=PATH` to specify different values for a particular type of files.

Files Association

To associate Caneda file types with the application itself, execute the following command:

```
$ update-mime-database /usr/local/share/mime
```

Documentation

Caneda uses **Doxygen** as its code documentation for developers. To generate the Doxygen documentation, use the Doxyfile file provided at the source root.

Debian 9 Stretch/Testing

The following instructions to compile Caneda are meant to be executed at the system console. Consider the prefix `$` for non-privileged user commands and the prefix `#` for root commands.

Use **aptitude** to install the required/suggested packages:

```
# aptitude install git cmake g++ qtbase5-dev libqt5svg5-dev qttools5-dev qttools5-dev-tools ngspice l
```

Note: The simulation engine **Ngspice** is available at the *non-free* Debian repository. Before trying to install it, edit the file **/etc/apt/sources.list** as **root** and make sure the keyword *non-free* is present in the repository details. For example: `deb http://ftp.us.debian.org/debian/ stretch main non-free`.

Create a temporary folder and change to it:

```
$ cd ~
$ mkdir temp_caneda
$ cd temp_caneda
```

Get the latest Caneda *git* snapshot:

```
$ git clone https://github.com/caneda/caneda
```

Note: The default Caneda distribution is *master* (stable). Execute `git checkout develop` to compile the *develop* (latest) distribution.

Alternatively, download either master or develop tarball from the GitHub repository and unpack it:

```
$ tar -xvf Caneda-Caneda-<version>.tar.gz
```

Create the folder `build` at the top of the source code structure and change into it:

```
$ cd caneda
$ mkdir build
$ cd build
```

Configure the source package and start the compilation process:

```
$ cmake ../
$ make
```

Install Caneda into the system by executing:

```
# make install
```

To open Caneda from the Application Launcher, open **Applications > Development** and then click on *Caneda*.

Alternatively, open Caneda from a terminal emulator by executing `caneda &`.

The temporary folder `~/temp_caneda` may be deleted at this point.

Debian 8 Jessie/Stable

The following instructions to compile Caneda are meant to be executed at the system console. Consider the prefix `$` for non-privileged user commands and the prefix `#` for root commands.

Use **aptitude** to install the required/suggested packages:

```
# aptitude install git cmake g++ qtbase5-dev libqt5svg5-dev qttools5-dev qttools5-dev-tools ngspice
```

Note: The simulation engine **Ngspice** is available at the *non-free* Debian repository. Before trying to install it, edit the file **/etc/apt/sources.list** as **root** and make sure the keyword *non-free* is present in the repository details. For example: `deb http://ftp.us.debian.org/debian/ jessie main non-free`.

Create a temporary folder and change to it:

```
$ cd ~
$ mkdir temp_caneda
$ cd temp_caneda
```

Note: Current Debian stable release Jessie does not support the Qt 5 version of Qwt yet. It is supported in Stretch (currently testing), so its source package must be downloaded and compiled in order to install Caneda, as described below.

Download the latest Qwt libraries (e.g. release 6.1.2) and unpack it:

```
$ tar -xvf qwt-6.1.2.tar.bz2
```

Change to Qwt folder to configure and compile the source code:

```
$ cd qwt-6.1.2
$ /usr/lib/x86_64-linux-gnu/qt5/bin/qmake qwt.pro
$ make
```

Install Qwt into the system:

```
# make install
```

Go back to the temporary folder created earlier:

```
$ cd ~/temp_caneda
```

Get the latest Caneda *git* snapshot:

```
$ git clone https://github.com/caneda/caneda
```

Note: The default Caneda distribution is *master* (stable). Execute `git checkout develop` to compile the *develop* (latest) distribution.

Alternatively, download either master or develop tarball from the GitHub repository and unpack it:

```
$ tar -xvf Caneda-Caneda-<version>.tar.gz
```

Create the folder `build` at the top of the source code structure and change into it:

```
$ cd caneda
$ mkdir build
$ cd build
```

Configure the source package and start the compilation process:

```
$ cmake ../
$ make
```

Install Caneda into the system by executing:

```
# make install
```

To open Caneda, point explicitly to Qwt 6.1.2 library path as follows:

```
$ LD_LIBRARY_PATH=/usr/local/qwt-6.1.2/lib/ caneda &
```

Note: Usually Caneda is opened from Application Launcher or by executing `caneda` from command line, but Qwt is installed by default into a folder where the operating system cannot locate it. For these reason, the Application Launcher shortcut will not work in Debian stable.

The temporary folder `~/temp_caneda` may be deleted at this point.

1.2 Installation

This document details how to install Caneda on *GNU/Linux* and *Microsoft Windows* operating systems.

1.2.1 GNU/Linux

The installation procedures for the most popular GNU/Linux distributions are provided here:

- *Debian 9 Stretch/Testing*
- *Debian 8 Jessie/Stable*

Note: To obtain the distribution name and release of a particular system, execute `lsb_release -da` from a system console.

Debian 9 Stretch/Testing

Caneda can be downloaded and installed right from the Debian repository by using the package manager **aptitude**. For more information, go to Debian Packages.

The following instructions to install Caneda are meant to be executed at the system console. Consider the prefix `$` for non-privileged user commands and the prefix `#` for root commands.

Make sure the system package list is updated by executing:

```
# aptitude update
```

Install **Caneda** (main) and **Ngspice** (simulation engine) by simply executing:

```
# aptitude install caneda ngspice
```

Note: The simulation engine **Ngspice** is available at the *non-free* Debian repository. Before trying to install it, edit the file **/etc/apt/sources.list** as **root** and make sure the keyword *non-free* is present in the repository details. For example: `deb http://ftp.us.debian.org/debian/ stretch main non-free`.

To open Caneda from the Application Launcher, go to **Applications > Development** and then click on *Caneda*.

Alternatively, open Caneda from a terminal emulator by executing `caneda &`.

Debian 8 Jessie/Stable

Although Caneda is not currently available at the Debian repository for Jessie/Stable, it can be installed from its source code. Please read *Debian 8 Jessie/Stable* compilation procedure for step-by-step instructions.

1.2.2 Microsoft Windows

Caneda *stable* release is currently supported in Microsoft Windows systems.

Simply download the Caneda installation executable CanedaInstaller_<latest>.exe and open it to begin the installation process.

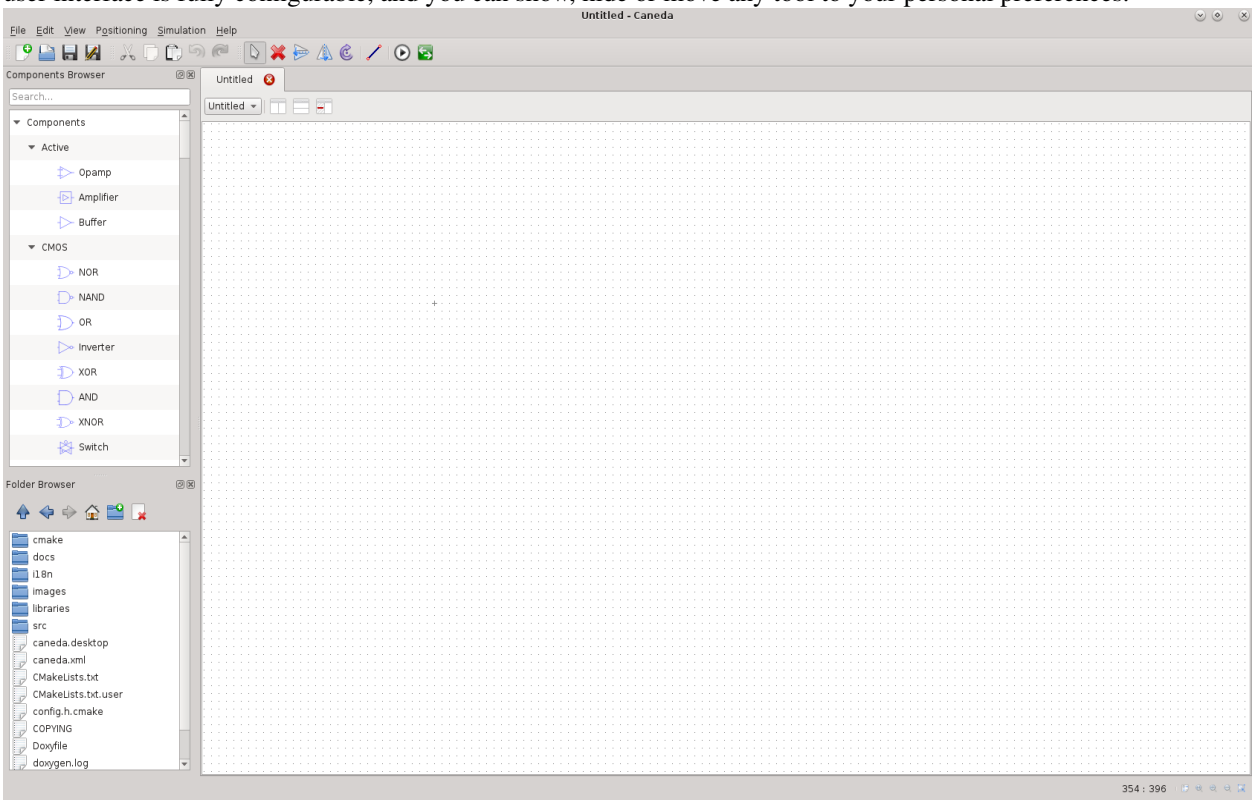
1.3 Getting Started

This is a brief introduction to Caneda's functionalities to get you ready on track. There are basically two ways to get started using Caneda right from scratch:

- Using the schematic context and creating a new design
- Using one of the circuit examples available from the *Example Circuits* repository.

1.3.1 Interface Introduction

In the following image the Caneda main window is shown, along with the initial tools and default configuration. The user interface is fully configurable, and you can show, hide or move any tool to your personal preferences.

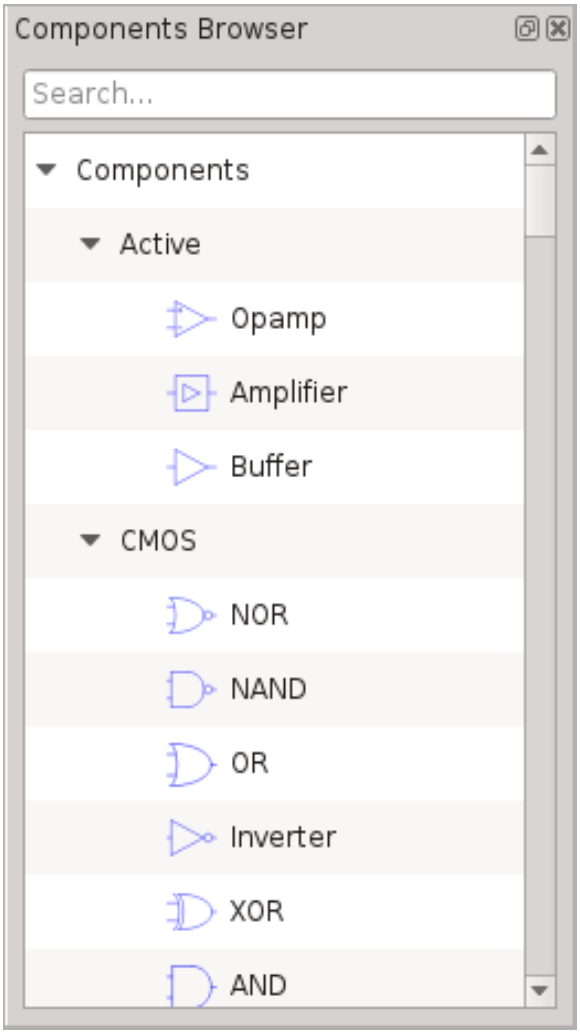


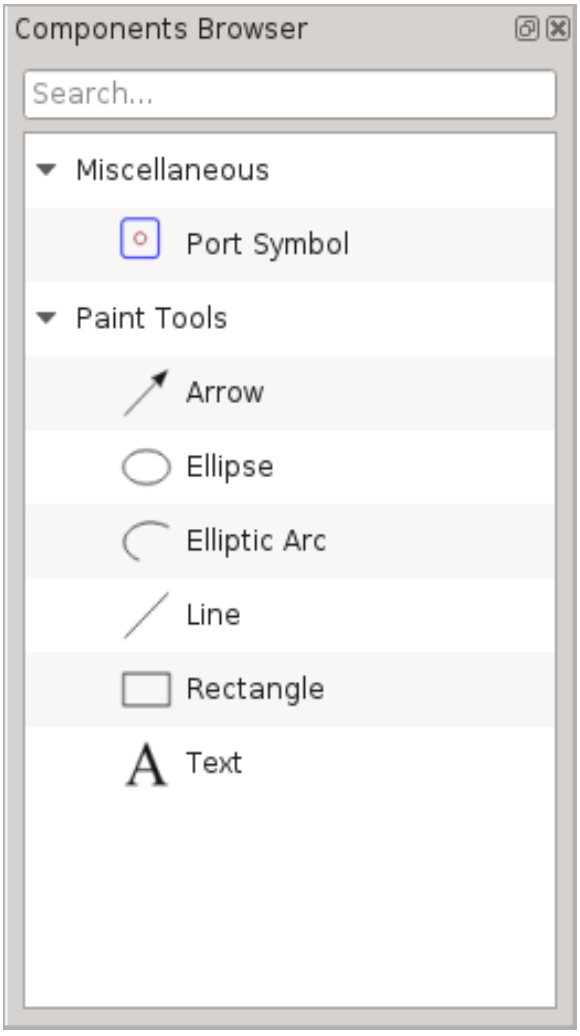
The file and edit toolbars contain general tools, available in most programs today. They don't need any introduction, as their tools are common knowledge.

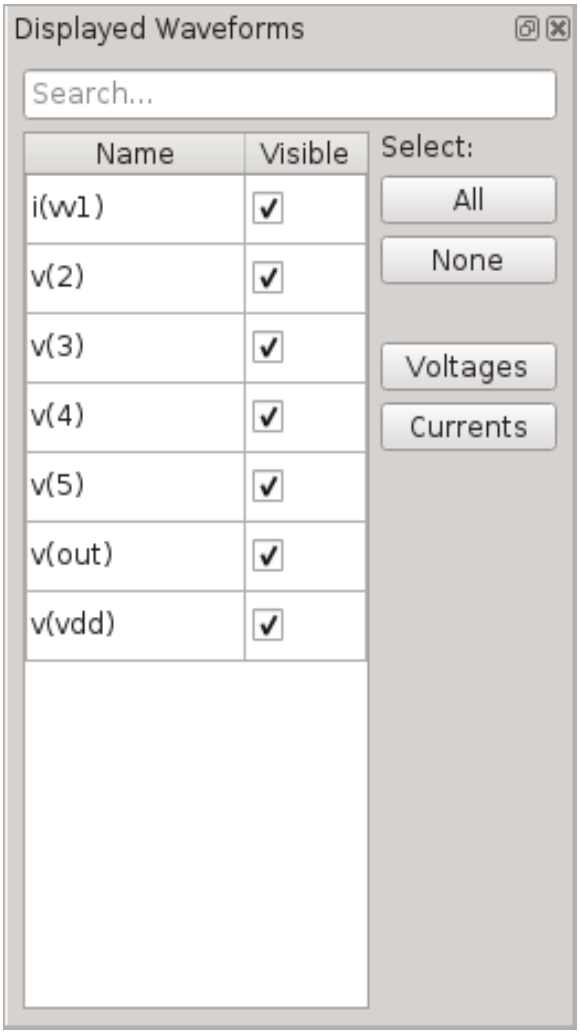
In general, there can be several contexts. In Caneda terminology, a context is a type of document you can edit for a certain task. For example, a schematic capture is one of Caneda's contexts. Other examples include a symbol editing, a simulation view, a layout editing. Every context has a set of tools and components associated with it, which you can use only in that context. For example, inserting a wire is useful if you are editing a schematic, however if you are analyzing a simulation, inserting a wire has no sense. In that case, inserting a cursor may be more applicable to that particular context. The following tools are all context sensitive, changing depending on the current context you are working on.

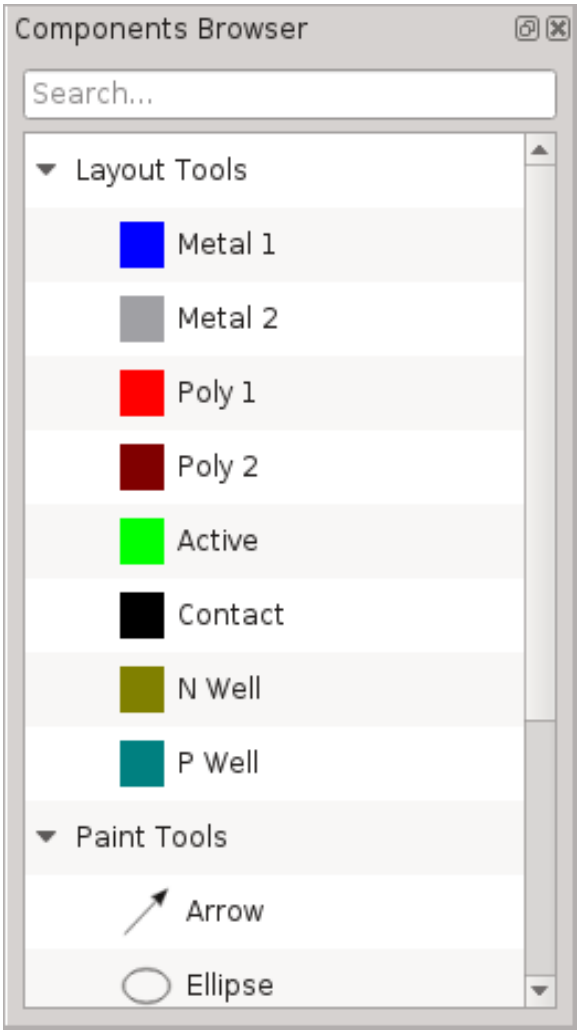
The main toolbar shows the more frequent options, giving you easy access to the needed tools for the current context.

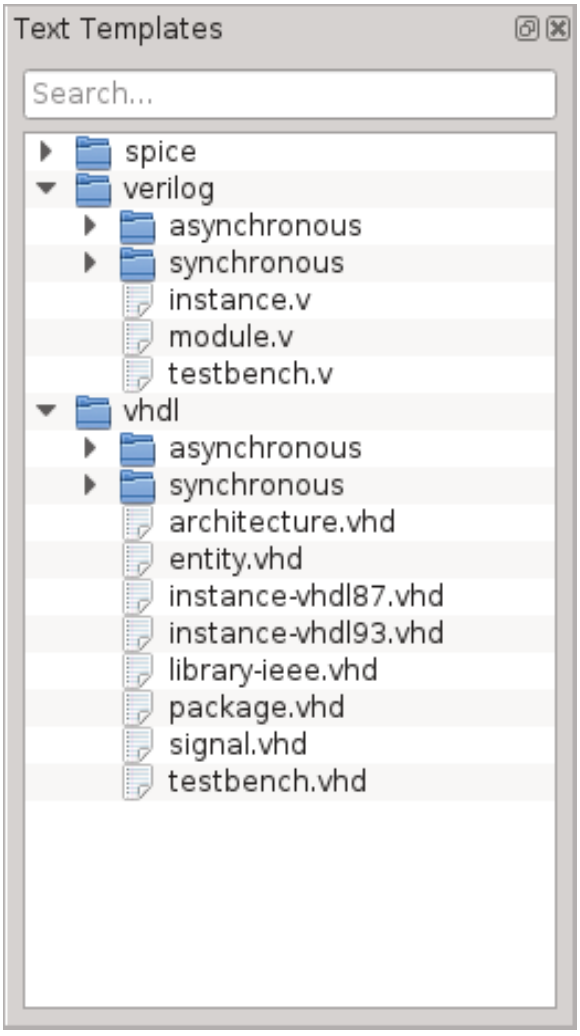
The sidebar browser is one of Caneda's most useful tools. In the following images you can see it in action for the schematic, symbol, layout and simulation contexts. To use it, simply select a component from the tool and start inserting it right away in the current view. In the case of the simulation context, you can select or deselect the waveforms of the view. The sidebar browser has a convenient filter tool, to display only those items you are interested in. As a shortcut to the component filter you can press the *c* key on your keyboard.


Components Browser
Search...
Components
Active
Opamp
Amplifier
Buffer
CMOS
NOR
NAND
OR
Inverter
XOR
AND

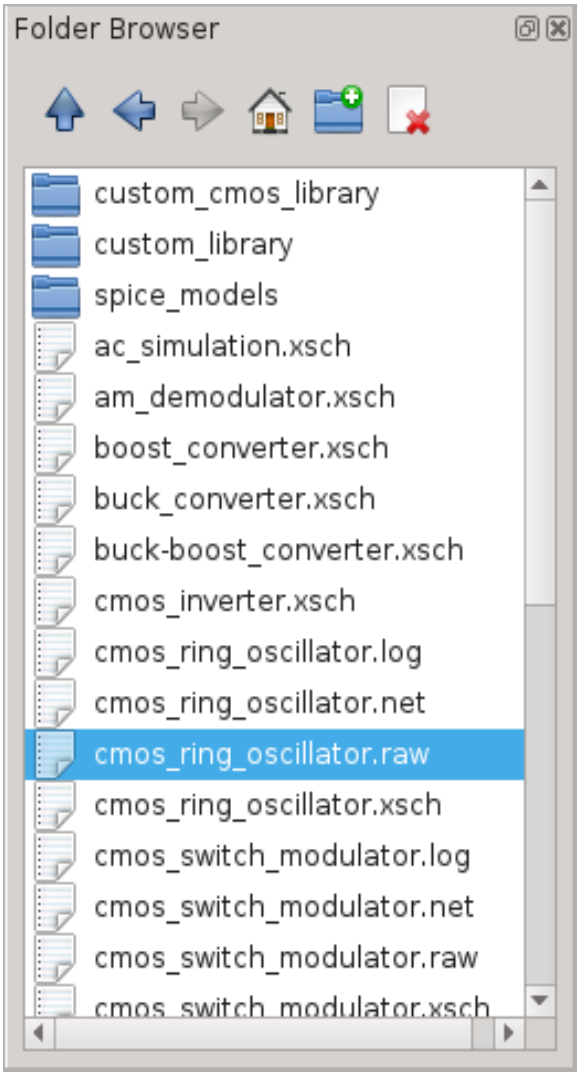

Components Browser
Search...
Miscellaneous
Port Symbol
Paint Tools
Arrow
Ellipse
Elliptic Arc
Line
Rectangle
Text


Displayed Waveforms
Search...
Name
Visible
i(vv1)
v(2)
v(3)
v(4)
v(5)
v(out)
v(vdd)
Select:
All
None
Voltages
Currents


Components Browser
Search...
Layout Tools
Metal 1
Metal 2
Poly 1
Poly 2
Active
Contact
N Well
P Well
Paint Tools
Arrow
Ellipse



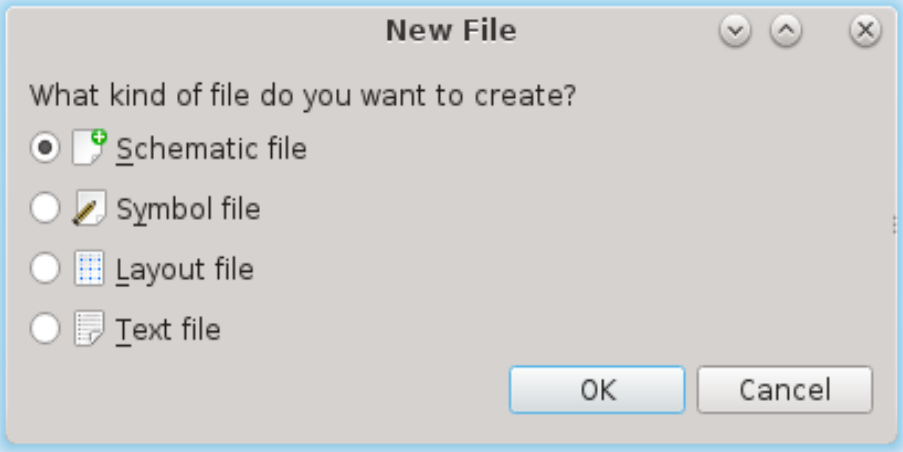
Finally, the folder browser allows you to easily access your files from a convenient place and open them right from the Caneda interface.



1.3.2 Create a New Design

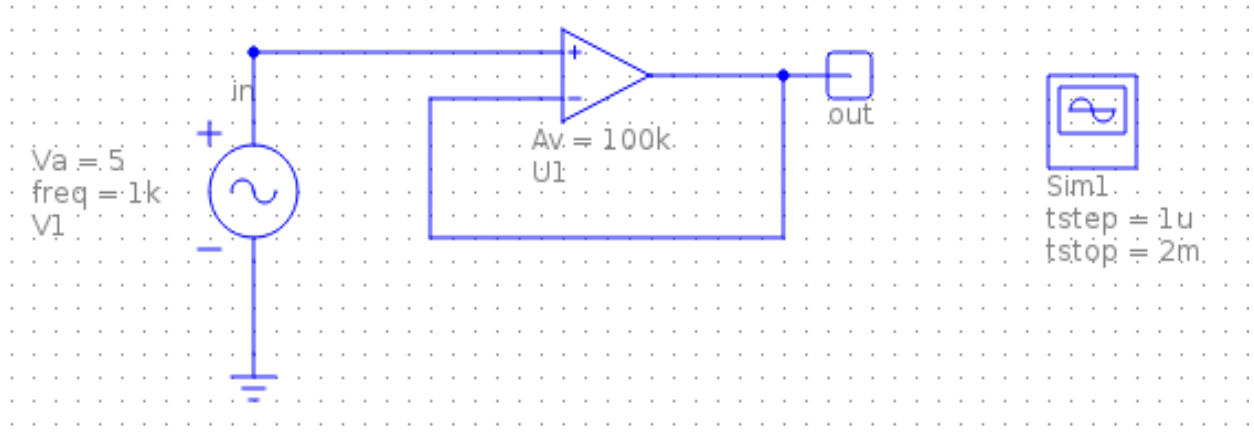
In the following example, we will introduce the basic usage by creating a simple opamp based amplifier. To start a new design, click on the *New* icon from the file toolbar.

A *New File* dialog will open, giving you the option to choose between several file types. For this example, we will create a new schematic file. Select the *Schematic file* option and click *OK*.

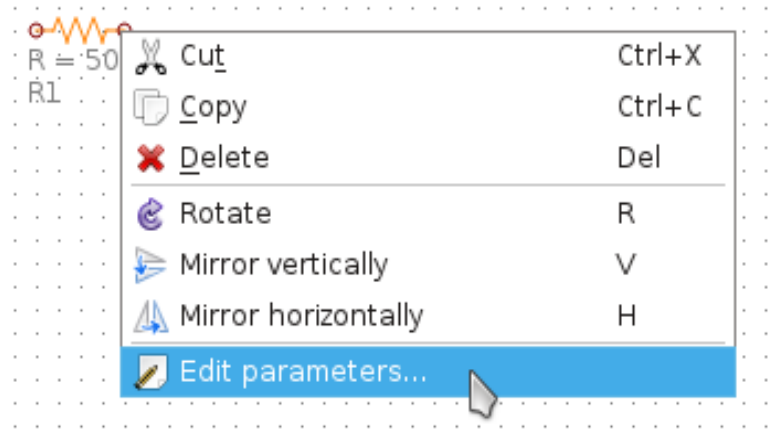


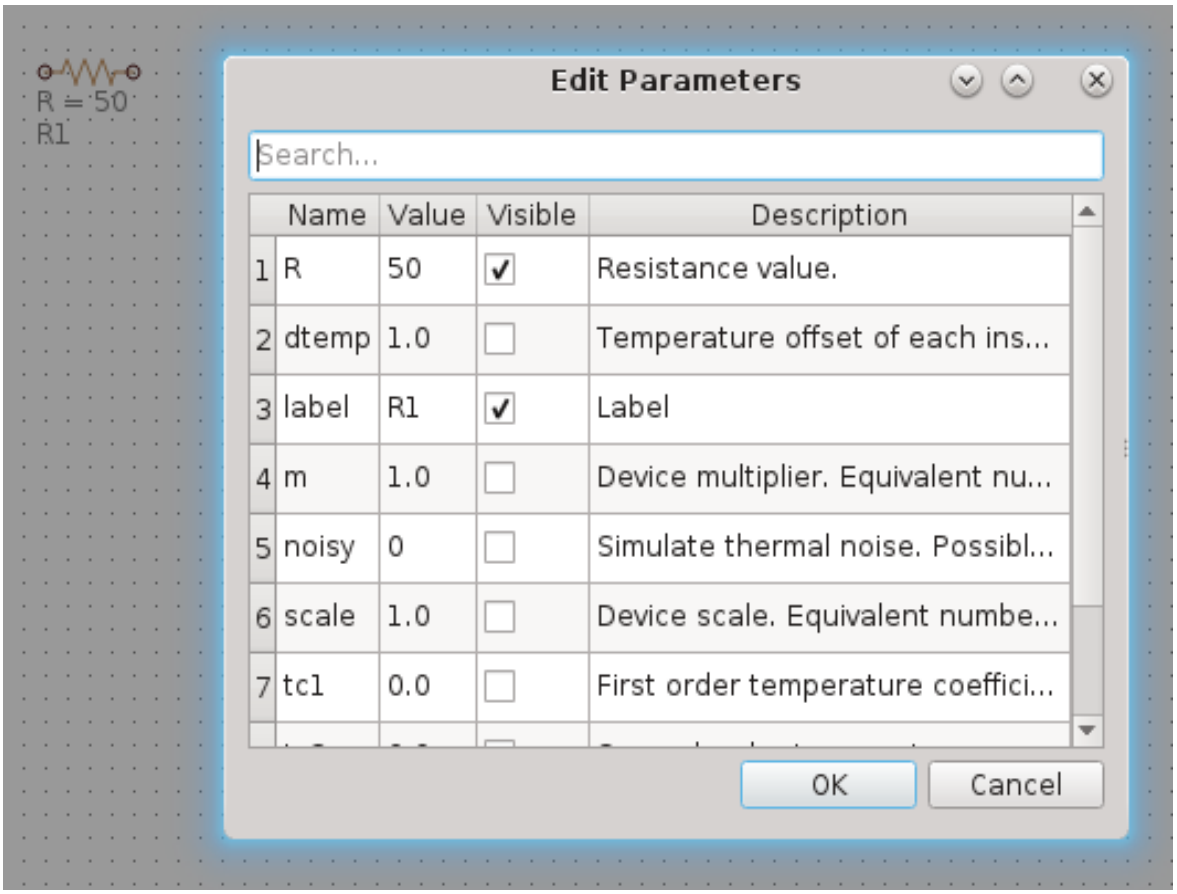
Alternatively, if you just started the program, you could have used the already opened new schematic document. Start editing you schematic, by adding components from the sidebar browser and wires from the main toolbar. The following image shows the wire tool. When you are in doubt what action performs each tool, just leave the mouse on the selected tool for a couple of seconds, and a tooltip will appear.

Using the previous tools, create an schematic as the shown in the following image.

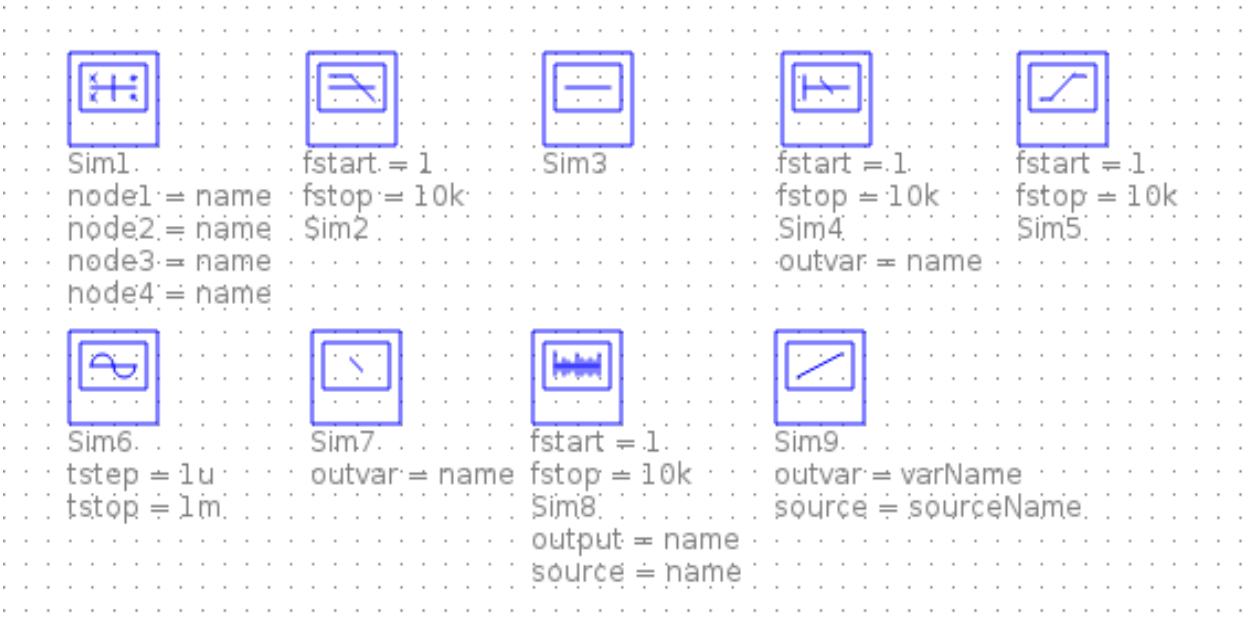


To edit a component property, for example the impedance value in a resistor, double click on the inserted component or select the *Edit parameters...* option on the context menu (by left mouse clicking on the component).

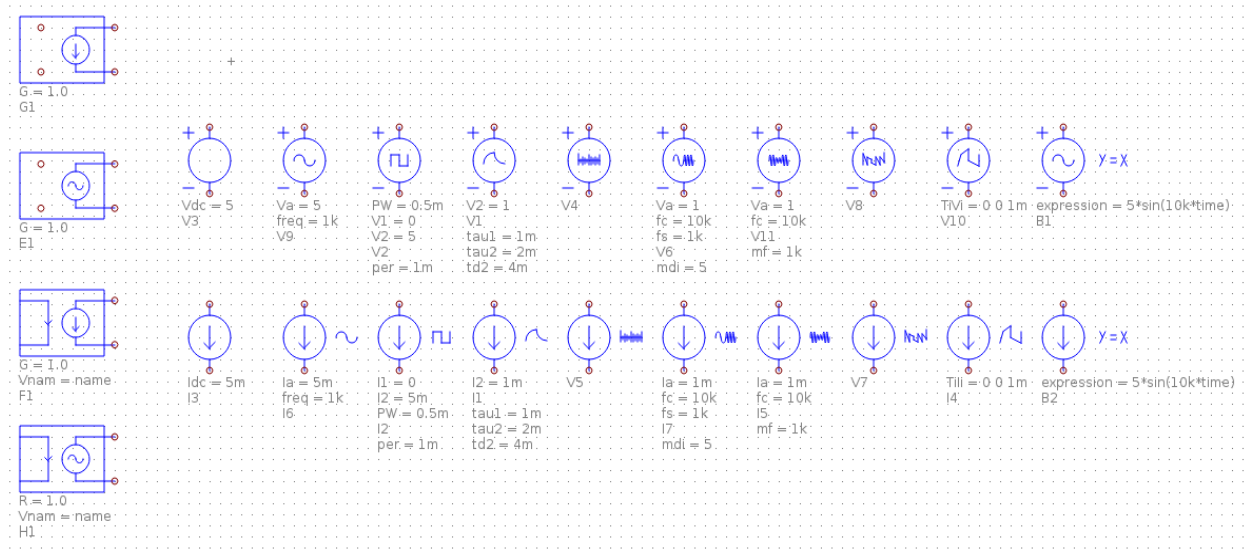




To perform any simulation it is always important to add, besides your desired components, at least one ground component and a simulation profile.



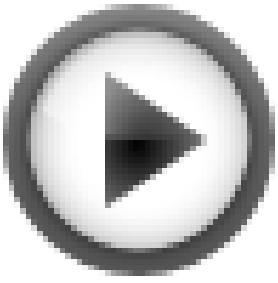
To get a meaningful result, you should also add a source.



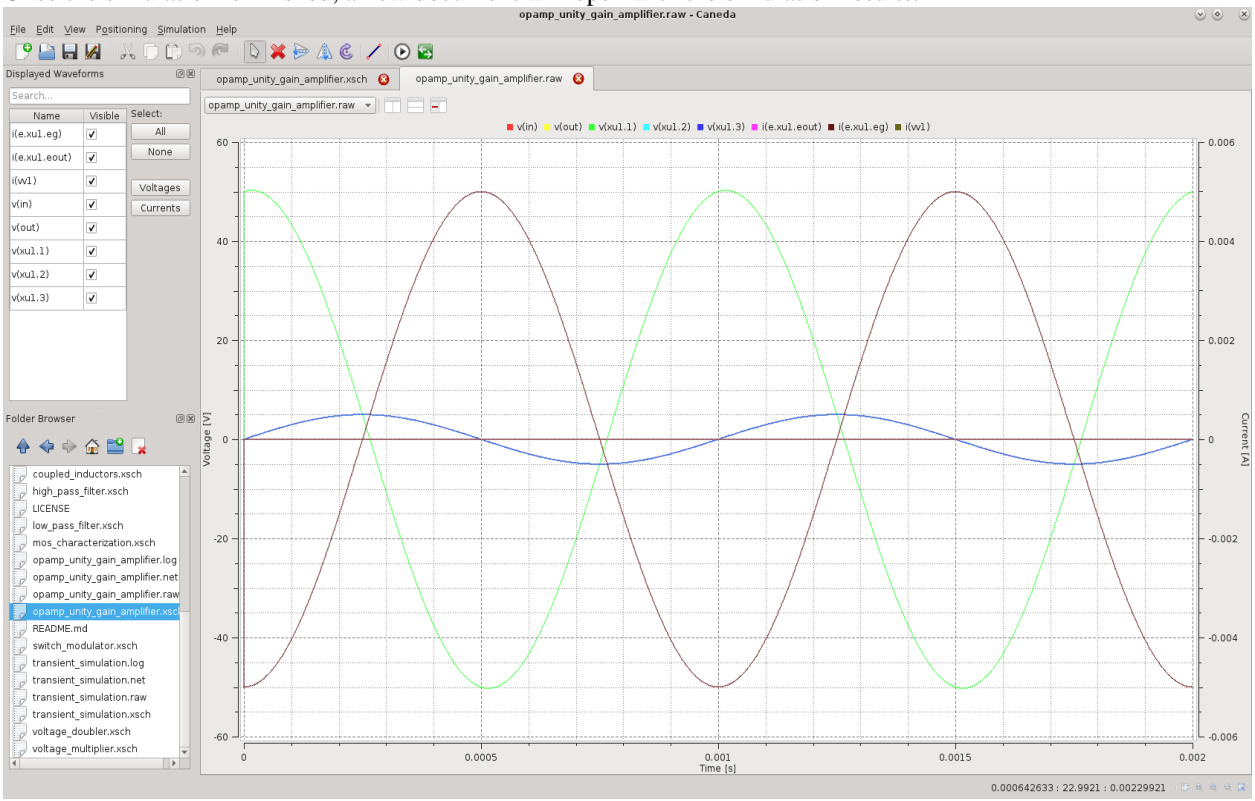
Once you are done, you are ready to perform your first simulation in Caneda.

1.3.3 Performing Simulations

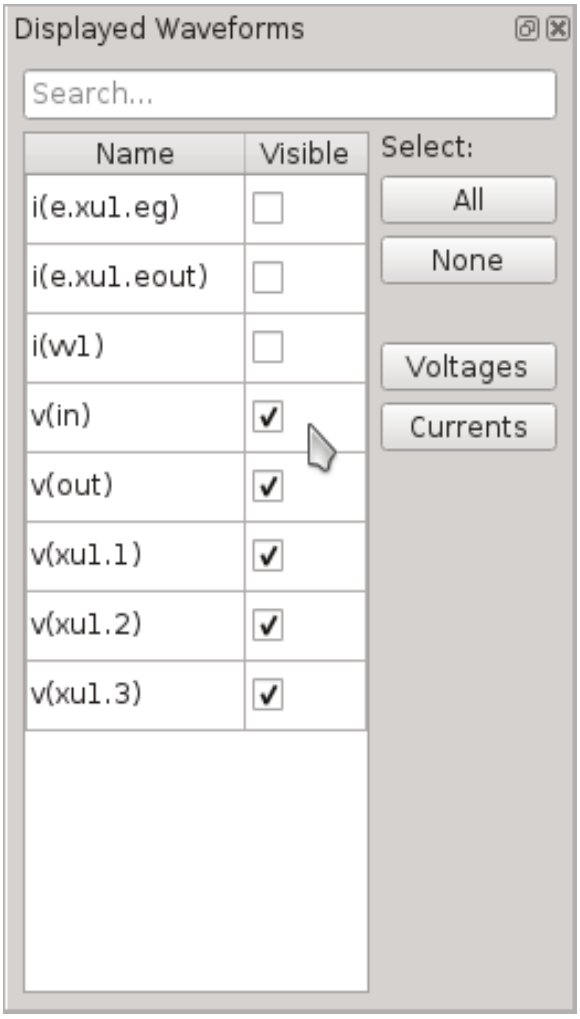
To start a new simulation, click on the simulate tool in the main toolbar.



Once the simulation is finished, a new document will open with the simulation results.

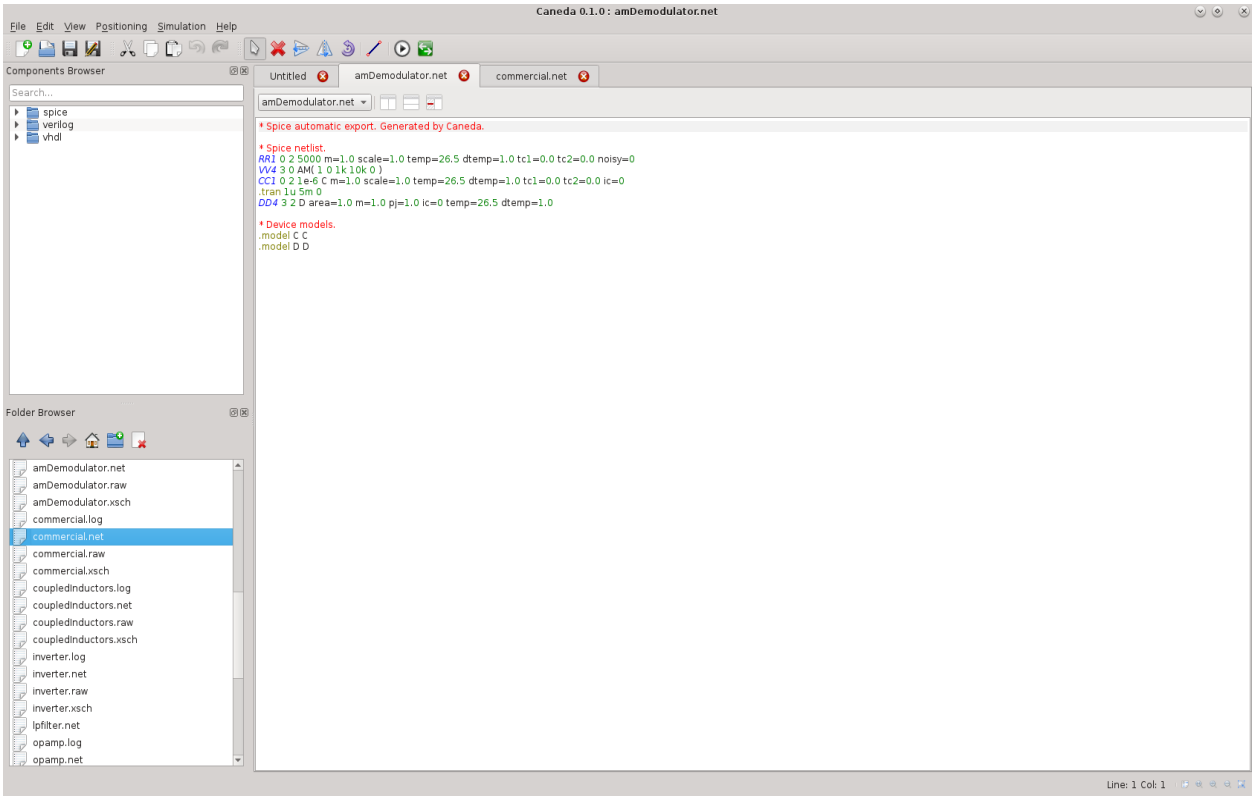


To select the displayed waveforms, click on the *Visible* option from the simulation sidebar.



1.3.4 Text editor

Caneda has an embedded text editor, which allows you to create and design all kinds of text code based simulations. The most common forms are spice and hdl (verilog and vhdl) simulations. To help you realize your design, a ready to use set of templates is available from the text sidebar. Once you have your design ready, just click on the simulate tool as in the above example and Caneda will perform the simulation for you, using the appropiate tools.



1.3.5 Circuit Examples

There is a repository hosted at Example Circuits which contains example circuits for Caneda, ready to use. The example circuits are continuously updated and improved, and periodically new circuits are added.

To try any circuit, simply download the selected schematic to your machine and open it using Caneda. All circuits are ready for simulation using Caneda's tools. Inside each schematic there are specific details to the circuit, where applicable.

As an alternative you can download the whole repository and save it in any local folder to have as a reference.

1.4 Component Libraries

There is a certain point in any design where the user needs to find extra components to those found in any EDA application. Caneda has several methods to add new components, which are described below.

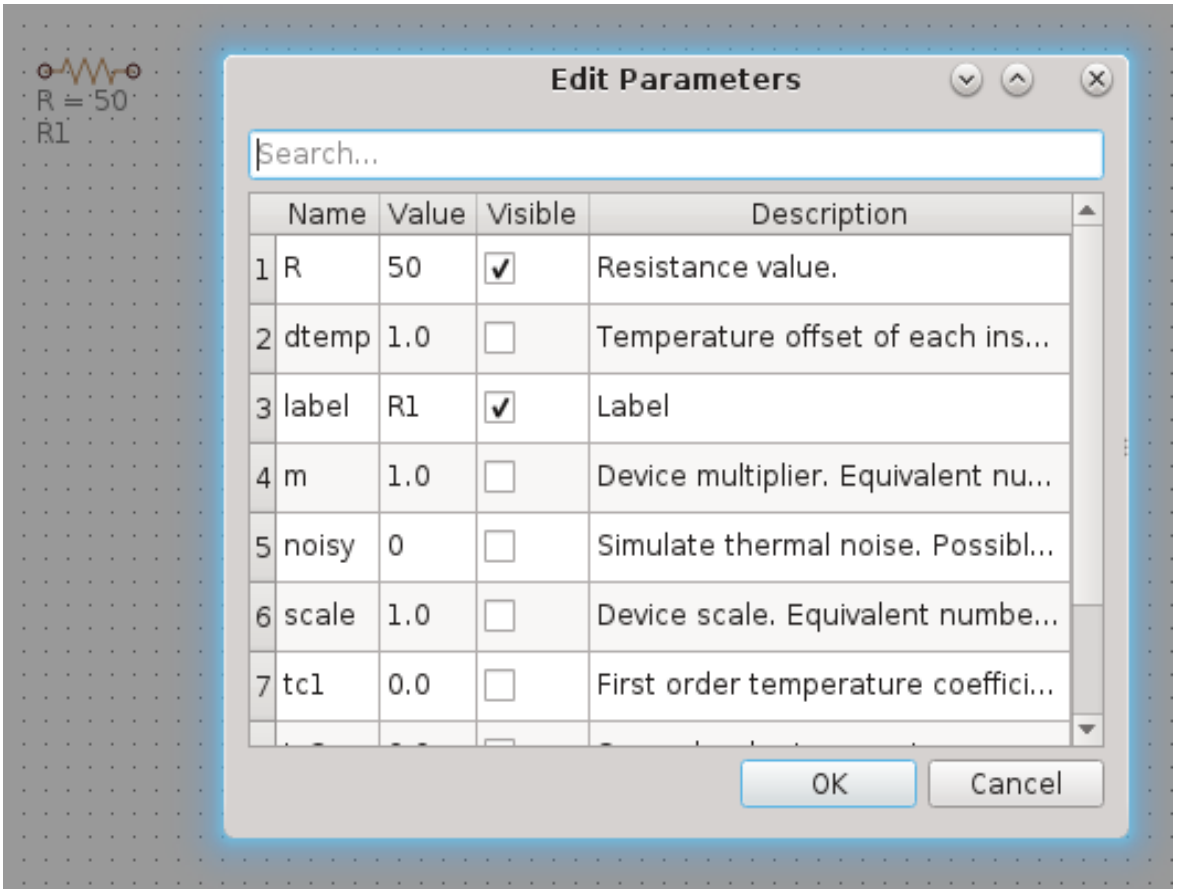
1.4.1 Spice Models

In Caneda you can modify a component's model or use an external spice model for your simulations. For every component there are basically three methods to modify a component's model:

- Modify the basic parameters of the model.
- Use a *model component* and modify the needed parameters.
- Include an external library file and use a model from the library.

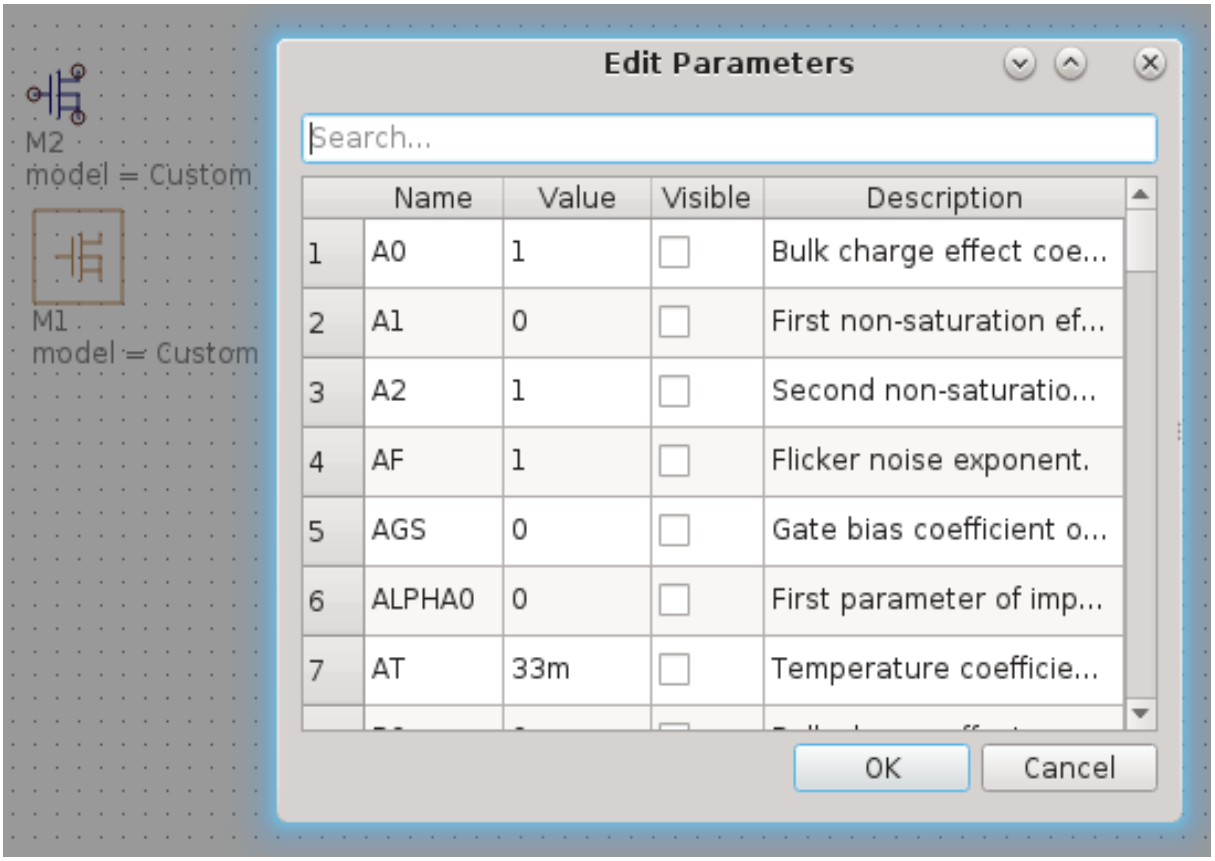
Modify the Basic Parameters of the Model

The easier way to modify a component's behaviour is to modify the basic parameters of the model. In order to do so, include a component and edit its parameters with the required values.



Use a Model Component

If you need to modify more parameters than those available in the basic parameters, you can include a *model component*. There are several *model components* for different kind of devices including (but not limited to) transistors, diodes, transmission lines, switches, capacitors and inductors. You can also use this method if you have several components of the same type and you don't want to type the same parameter's values for each component. After you include the *model component*, edit its parameters and use its `name` as the `model` field (parameter) in every instance of the component.



Include an External Library File

A similar alternative to using a *model component* is to include an external library file (a text file containing the model name and all the parameters' values) and use the model's `name` as the `model` field (parameter) in all the components needed. To include the external library file you can use the *library* or the *include* component. The main difference between an *include* and a *library* is that while in an *include* you can have only one reference for each model (defining its parameters), in a *library* you can have multiple references for each model (one in each library of the library file). In this way, for example, you could have a library with the typical, a library with the maximum and a library with the minimum parameters of a component (all in one file) and include the library you need in the simulation you are performing.

The advantage of using an external library file over the previous methods is that you can download the needed model or library from the web and use it directly in your design without needing to manually copy every parameter to a *model component*. This method is similar to that found in commercial simulation programs.

When using library files it is important to note that when using complex components described by spice subcircuits both the number of ports and the number of parameters must be equal between the symbol used and the model included in the library file. If using for example an opamp, and the number of pins between the spice library file and the symbol is not the same, a new opamp symbol must be created with the same number of pins as the component in the spice

library file. This also holds true for the number of parameters (usually zero for a downloaded commercial model).

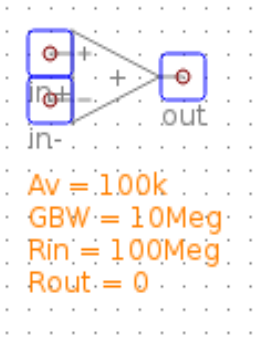
In the repository folder hosted at Spice Models you will find some basic examples using the previously described methods. The included models in the provided examples are not based on real data, but rather manually modified for educational purposes only.

1.4.2 Custom Libraries

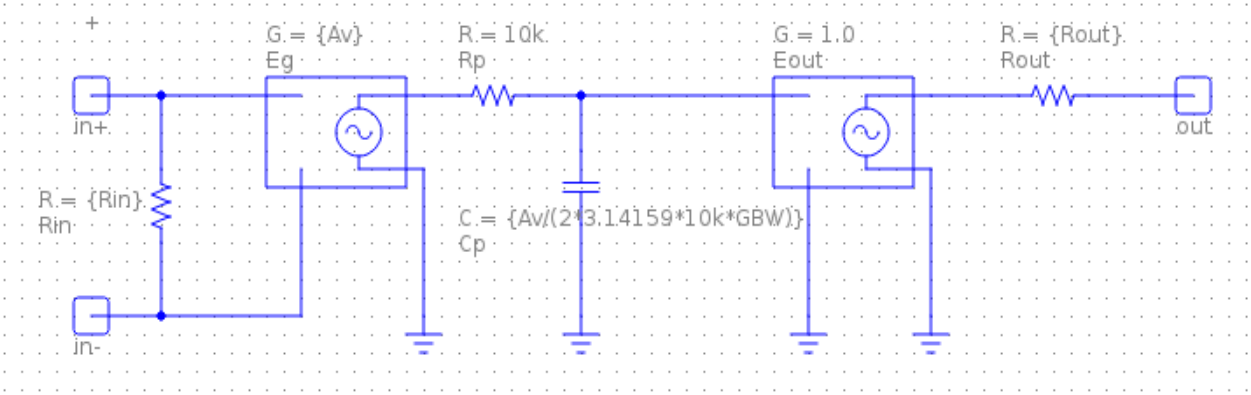
In Caneda, libraries are simply folders with schematics and symbols inside. For each component created, both a schematic describing the electric circuit and a symbol with the component representation must exist. The connection between the schematic and the symbol is performed by the use of ports in both files. Each port in the schematic will be connected with the corresponding port in the symbol, matching both ports by name. Parameters may be exposed in the symbol and each parameter will be matched to a parameter in the schematic. In this case, the parameter name only in the schematic must be enclosed by brackets `{}` indicating a spice parameter operation. Mathematical operations are supported, giving extra flexibility.

For example, if a custom resistor was to be created, the symbol would contain the drawing, two ports and a parameter named `{R_custom}`. On the other hand, the schematic would contain the same two ports and a resistor (describing the simple electric circuit of a resistor) with a value `R={R_custom}`. In the same way, the value could be `R={R_custom*2/2}` using mathematical operations together with the parameter.

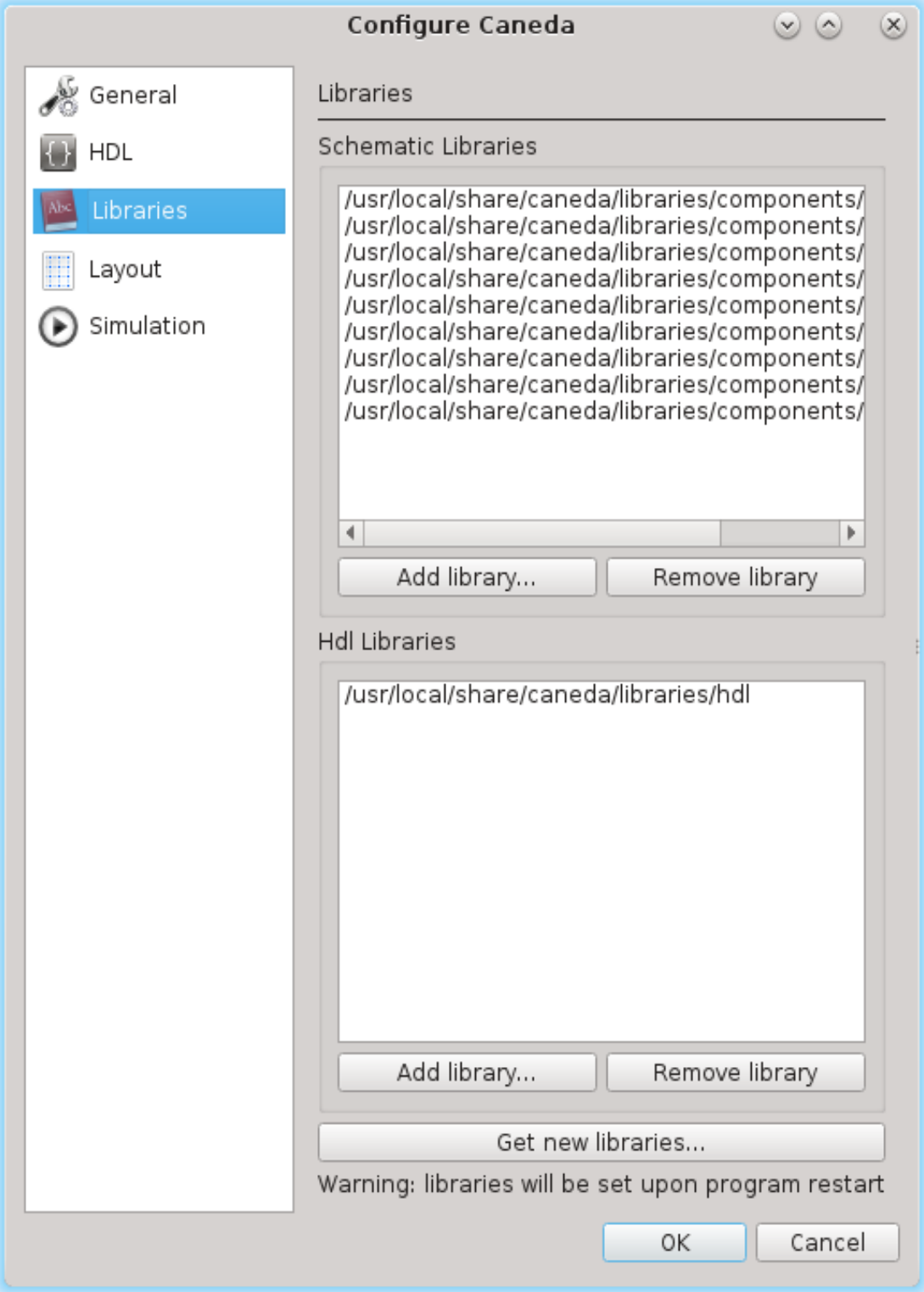
In the following image, a custom symbol for an operational amplifier is shown:



Corresponding to the previous symbol (using the same filename), in the following image the custom schematic for the same operational amplifier is shown:



Finally, to add the recently created library to Caneda, open the *Application Settings* dialog and in the *Libraries* section add the custom created library by selecting its folder. Any successive change in the library will be reflected upon the next program restart.


Configure Caneda
General
HDL
Libraries
Layout
Simulation
Libraries
Schematic Libraries
/usr/local/share/caneda/libraries/components/
/usr/local/share/caneda/libraries/components/
/usr/local/share/caneda/libraries/components/
/usr/local/share/caneda/libraries/components/
/usr/local/share/caneda/libraries/components/
/usr/local/share/caneda/libraries/components/
/usr/local/share/caneda/libraries/components/
/usr/local/share/caneda/libraries/components/
/usr/local/share/caneda/libraries/components/
Add library...
Remove library
Hdl Libraries
/usr/local/share/caneda/libraries/hdl
Add library...
Remove library
Get new libraries...
Warning: libraries will be set upon program restart
OK
Cancel

In the repository folder hosted at Custom Library, you will find a couple of components created to show the previously described procedure. The example components are ready to use, and if you download the whole folder you can include it in your project by adding the folder as a library to Caneda.

1.4.3 Extra Libraries Repository

Finally Caneda has a special repository which holds extra libraries and components, ready to use. If you download a whole folder from that repository you can include it in your project by adding the folder as a library to Caneda.

Caneda's libraries repository is hosted at:

https://github.com/Caneda/Libraries

To use any library, you must download the selected folder and add it to your library list through the *Application Settings* dialog in the *Libraries* section. Any successive change in the library will be reflected upon the next program restart.

1.5 Troubleshooting

To locate an issue, please enter a few keywords at **Search docs** or use the document tree on the left.

1.5.1 Libraries

NO COMPONENTS ARE LISTED AFTER REINSTALLING CANEDA INTO A DIFFERENT FOLDER

After reinstalling Caneda to a different system folder, it shows few or no components available at the sidebar browser.

This happens is because an obsolete configuration file is pointing to an empty library path.

Go to *Settings* and delete every library path one by one. Close the application and then open it again. Caneda will complete the missing library paths with default values and the components will be shown again at the sidebar browser.

1.5.2 Compilation

BASH: CMAKE: COMMAND NOT FOUND

After executing `cmake ../` to install Caneda, the following error occurs:

```
bash: cmake: command not found
```

Check if package `cmake` is installed by executing `aptitude show cmake`. If it is actually not installed, execute `aptitude install cmake` as root. Then try again.

It is recommended to install the following packages at once, to avoid further compilation errors:

```
cmake
g++
qtbase5-dev
libqt5svg5-dev
qttools5-dev
qttools5-dev-tools
```

It is also recommended to install the optional package `ngspice` (simulation engine) as follows: `aptitude install ngspice`.

To install a package from the *non-free* Debian repository, first edit **/etc/apt/sources.list**, add the keyword **non-free** at the end of the distribution description (e.g. `deb http://ftp.us.debian.org/debian/ stable main non-free`) and then execute `aptitude update` as root. Finally, execute `aptitude install <package>`.

NO CMAKE_CXX_COMPILER COULD BE FOUND

After executing `cmake ../` to install Caneda, the following error occurs:

```
-- The C compiler identification is GNU 4.9.2
-- The CXX compiler identification is unknown
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
CMake Error at CMakeLists.txt:3 (PROJECT):
  No CMAKE_CXX_COMPILER could be found.

  Tell CMake where to find the compiler by setting either the environment
  variable "CXX" or the CMake cache entry CMAKE_CXX_COMPILER to the full path
  to the compiler, or to the compiler name if it is in the PATH.


-- Configuring incomplete, errors occurred!
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeOutput.log".
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeError.log".
```

Check if C++ compiler is installed by executing `aptitude show g++`. If it is actually now installed, execute `aptitude install g++` as root. Then try again.

It is recommended to install the following packages at once, to avoid further compilation errors:

```
g++
qtbase5-dev
libqt5svg5-dev
qttools5-dev
qttools5-dev-tools
```

It is also recommended to install the optional package `ngspice` (simulation engine) as follows: `aptitude install ngspice`.

To install a package from the *non-free* Debian repository, first edit **/etc/apt/sources.list**, add the keyword **non-free** at the end of the distribution description (e.g. `deb http://ftp.us.debian.org/debian/ stable main non-free`) and then execute `aptitude update` as root. Finally, execute `aptitude install <package>`.

BY NOT PROVIDING "FINDQT5WIDGETS.CMAKE" IN CMAKE_MODULE_PATH...

After executing `cmake ../` to install Caneda, the following error occurs:

```
-- The CXX compiler identification is GNU 4.9.2
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
CMake Error at CMakeLists.txt:14 (FIND_PACKAGE):
  By not providing "FindQt5Widgets.cmake" in CMAKE_MODULE_PATH this project
  has asked CMake to find a package configuration file provided by
  "Qt5Widgets", but CMake did not find one.

  Could not find a package configuration file provided by "Qt5Widgets"
  (requested version 5.3.2) with any of the following names:
```

```
    Qt5WidgetsConfig.cmake
    qt5widgets-config.cmake

  Add the installation prefix of "Qt5Widgets" to CMAKE_PREFIX_PATH or set
  "Qt5Widgets_DIR" to a directory containing one of the above files.  If
  "Qt5Widgets" provides a separate development package or SDK, be sure it has
  been installed.


-- Configuring incomplete, errors occurred!
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeOutput.log".
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeError.log".
```

Install the Qt 5 base and SVG support definitions by executing `aptitude install qtbase5-dev` as root user, which in turn will install other required Qt 5 packages. Then, try again.

It is recommended to install the following packages at once, to avoid further compilation errors:

```
qtbase5-dev
libqt5svg5-dev
qttools5-dev
qttools5-dev-tools
```

It is also recommended to install the optional package `ngspice` (simulation engine) as follows: `aptitude install ngspice`.

To install a package from the *non-free* Debian repository, first edit **/etc/apt/sources.list**, add the keyword **non-free** at the end of the distribution description (e.g. `deb http://ftp.us.debian.org/debian/ stable main non-free`) and then execute `aptitude update` as root. Finally, execute `aptitude install <package>`.

BY NOT PROVIDING "FINDQT5SVG.CMAKE" IN CMAKE_MODULE_PATH...

After executing `cmake ../` to install Caneda, the following error occurs:

```
CMake Error at CMakeLists.txt:15 (FIND_PACKAGE):
  By not providing "FindQt5Svg.cmake" in CMAKE_MODULE_PATH this project has
  asked CMake to find a package configuration file provided by "Qt5Svg", but
  CMake did not find one.

  Could not find a package configuration file provided by "Qt5Svg" (requested
  version 5.3.2) with any of the following names:

    Qt5SvgConfig.cmake
    qt5svg-config.cmake

  Add the installation prefix of "Qt5Svg" to CMAKE_PREFIX_PATH or set
  "Qt5Svg_DIR" to a directory containing one of the above files.  If "Qt5Svg"
  provides a separate development package or SDK, be sure it has been
  installed.


-- Configuring incomplete, errors occurred!
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeOutput.log".
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeError.log".
```

Install the Qt 5 SVG support definitions by executing `aptitude install libqt5svg5-dev` as root user. Then, try again.

It is recommended to install the following packages at once, to avoid further compilation errors:

```
libqt5svg5-dev
qttools5-dev
qttools5-dev-tools
```

It is also recommended to install the optional package `ngspice` (simulation engine) as follows: `aptitude install ngspice`.

To install a package from the *non-free* Debian repository, first edit **/etc/apt/sources.list**, add the keyword **non-free** at the end of the distribution description (e.g. `deb http://ftp.us.debian.org/debian/ stable main non-free`) and then execute `aptitude update` as root. Finally, execute `aptitude install <package>`.

BY NOT PROVIDING "FINDQT5LINGUISTTOOLS.CMAKE" IN CMAKE_MODULE_PATH...

After executing `cmake ../` to install Caneda, the following error occurs:

```
CMake Error at CMakeLists.txt:17 (FIND_PACKAGE):
  By not providing "FindQt5LinguistTools.cmake" in CMAKE_MODULE_PATH this
  project has asked CMake to find a package configuration file provided by
  "Qt5LinguistTools", but CMake did not find one.

  Could not find a package configuration file provided by "Qt5LinguistTools"
  (requested version 5.3.2) with any of the following names:

    Qt5LinguistToolsConfig.cmake
    qt5linguisttools-config.cmake

  Add the installation prefix of "Qt5LinguistTools" to CMAKE_PREFIX_PATH or
  set "Qt5LinguistTools_DIR" to a directory containing one of the above
  files.  If "Qt5LinguistTools" provides a separate development package or
  SDK, be sure it has been installed.


-- Configuring incomplete, errors occurred!
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeOutput.log".
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeError.log".
```

Install the Qt 5 SVG support definitions by executing `aptitude install qttools5-dev` as root user, which in turn will install other required Qt 5 package. Then, try again.

It is recommended to install the following packages at once, to avoid further compilation errors:

```
qttools5-dev
qttools5-dev-tools
```

It is also recommended to install the optional package `ngspice` (simulation engine) as follows: `aptitude install ngspice`.

To install a package from the *non-free* Debian repository, first edit **/etc/apt/sources.list**, add the keyword **non-free** at the end of the distribution description (e.g. `deb http://ftp.us.debian.org/debian/ stable main non-free`) and then execute `aptitude update` as root. Finally, execute `aptitude install <package>`.

/USR/LIB/X86_64-LINUX-GNU/CMAKE/QT5LINGUISTTOOLS/QT5LINGUISTTOOLSCONFIG.CMAKE:22...

After executing `cmake ../` to install Caneda, the following error occurs:

```
CMake Error at /usr/lib/x86_64-linux-gnu/cmake/Qt5LinguistTools/Qt5LinguistToolsConfig.cmake:22 (mess
  The package "Qt5LinguistTools" references the file

    "/usr/lib/x86_64-linux-gnu/qt5/bin/lrelease"

  but this file does not exist.  Possible reasons include:
```

```
  * The file was deleted, renamed, or moved to another location.

  * An install or uninstall procedure did not complete successfully.

  * The installation package was faulty and contained

    "/usr/lib/x86_64-linux-gnu/cmake/Qt5LinguistTools/Qt5LinguistToolsConfig.cmake"

  but not all the files it references.

Call Stack (most recent call first):
  /usr/lib/x86_64-linux-gnu/cmake/Qt5LinguistTools/Qt5LinguistToolsConfig.cmake:38 (_qt5_LinguistTool
  CMakeLists.txt:17 (FIND_PACKAGE)


-- Configuring incomplete, errors occurred!
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeOutput.log".
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeError.log".
```

Install the Qt 5 SVG support definitions by executing `aptitude install qttools5-dev-tools` as root user. Then, try again.

It is also recommended to install the optional package `ngspice` (simulation engine) as follows: `aptitude install ngspice`.

To install a package from the *non-free* Debian repository, first edit **/etc/apt/sources.list**, add the keyword **non-free** at the end of the distribution description (e.g. `deb http://ftp.us.debian.org/debian/ stable main non-free`) and then execute `aptitude update` as root. Finally, execute `aptitude install <package>`.

/USR/SHARE/CMAKE-3.0/MODULES/FINDPACKAGEHANDLESTANDARDARGS.CMAKE:136...

After executing `cmake ../` to install Caneda, the following error occurs:

```
CMake Error at /usr/share/cmake-3.0/Modules/FindPackageHandleStandardArgs.cmake:136 (message):
  Could NOT find Qwt (missing: QWT_LIBRARY QWT_INCLUDE_DIR) (Required is at
  least version "6.1.2")
Call Stack (most recent call first):
  /usr/share/cmake-3.0/Modules/FindPackageHandleStandardArgs.cmake:343 (_FPHSA_FAILURE_MESSAGE)
  cmake/Modules/FindQwt.cmake:94 (FIND_PACKAGE_HANDLE_STANDARD_ARGS)
  CMakeLists.txt:22 (FIND_PACKAGE)


-- Configuring incomplete, errors occurred!
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeOutput.log".
See also "/home/user/caneda/caneda/build/CMakeFiles/CMakeError.log".
```

Download qwt 6.1.2 and extract the tar.gz file into a temporary folder.

Compile qwt with Qt5:

```
$ /usr/lib/x86_64-linux-gnu/qt5/bin/qmake qwt.pro
$ make
```

If compilation finished without errors, install Qwt by executing `make install` with root privileges, in order to install qwt libraries into system folders.

Then, try again.

It is also recommended to install the optional package `ngspice` (simulation engine) as follows: `aptitude install ngspice`.

To install a package from the *non-free* Debian repository, first edit **/etc/apt/sources.list**, add the keyword **non-free** at the end of the distribution description (e.g. `deb http://ftp.us.debian.org/debian/ stable main non-free`) and then execute `aptitude update` as root. Finally, execute `aptitude install <package>`.

LIBQWT.SO.6: CANNOT OPEN SHARED OBJECT FILE: NO SUCH FILE OR DIRECTORY

After executing `caneda` from a terminal emulator, the following error is displayed:

```
caneda: error while loading shared libraries: libqwt.so.6: cannot open shared object file: No such fi
```

If qwt 6.1.2 was installed from its source, the operating system does not know how to reach these libraries.

The solution is to execute Caneda from command line as follows:

```
$ LD_LIBRARY_PATH=/usr/local/qwt-6.1.2/lib/ caneda
```

Then, Caneda GUI should be displayed.

CHAPTER 2

Contribute

To get a quick feedback on any issue related to compilation, installation or use of Caneda, place your report at the Issue Tracker.

This project is open for new contributors. Check out the Caneda source code repository at GitHub.

CHAPTER 3

License

This project is licensed under GPLv2.