SiviCNCDriver Documentation

Release 0.1.9

Klafyvel

Dec 06, 2017

Contents:

CHAPTER 1

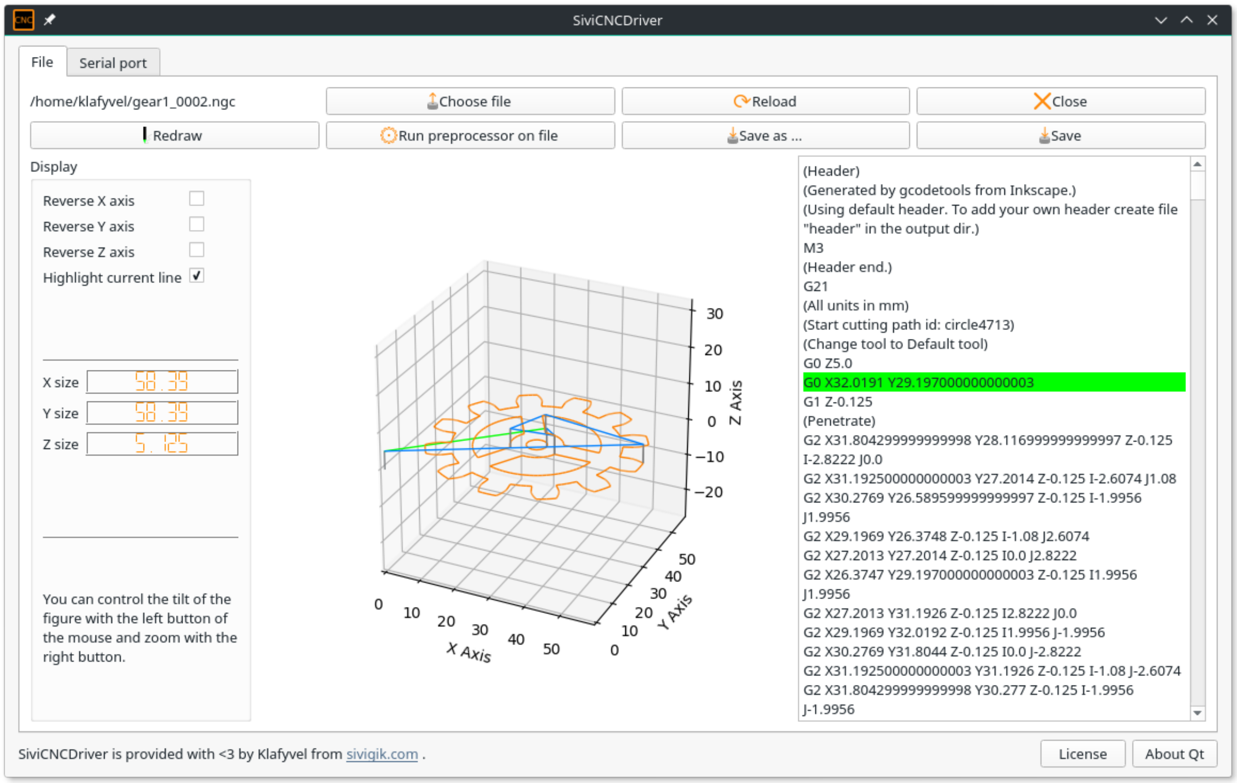
What is SiviCNCDriver ?

SiviCNCDriver, as its name lets you guess, is designed to drive a CNC. What does it do ?

- Provides a basic tool to view and edit G-Codes files. You can see which G-Code line draws which path and perform some basic edits with the preprocessor, such as finding an origin to the coordinate system which minimize the bounding box of the drawing.
- **Allows you to control manually your CNC, by :**
  - Sending your own G-Codes;
  - Sending *custom G-Codes* so the machine performs continuous movements, or step-by-step movements;
  - Sending automatic commands so the machine performs some goings and comings and you can measure the play or the steps/mm.
- Sends as *custom G-Codes* and store as JSON configuration files for your machine.

CHAPTER 2

Screenshots



/home/klafyvel/gear1_0002.ngc

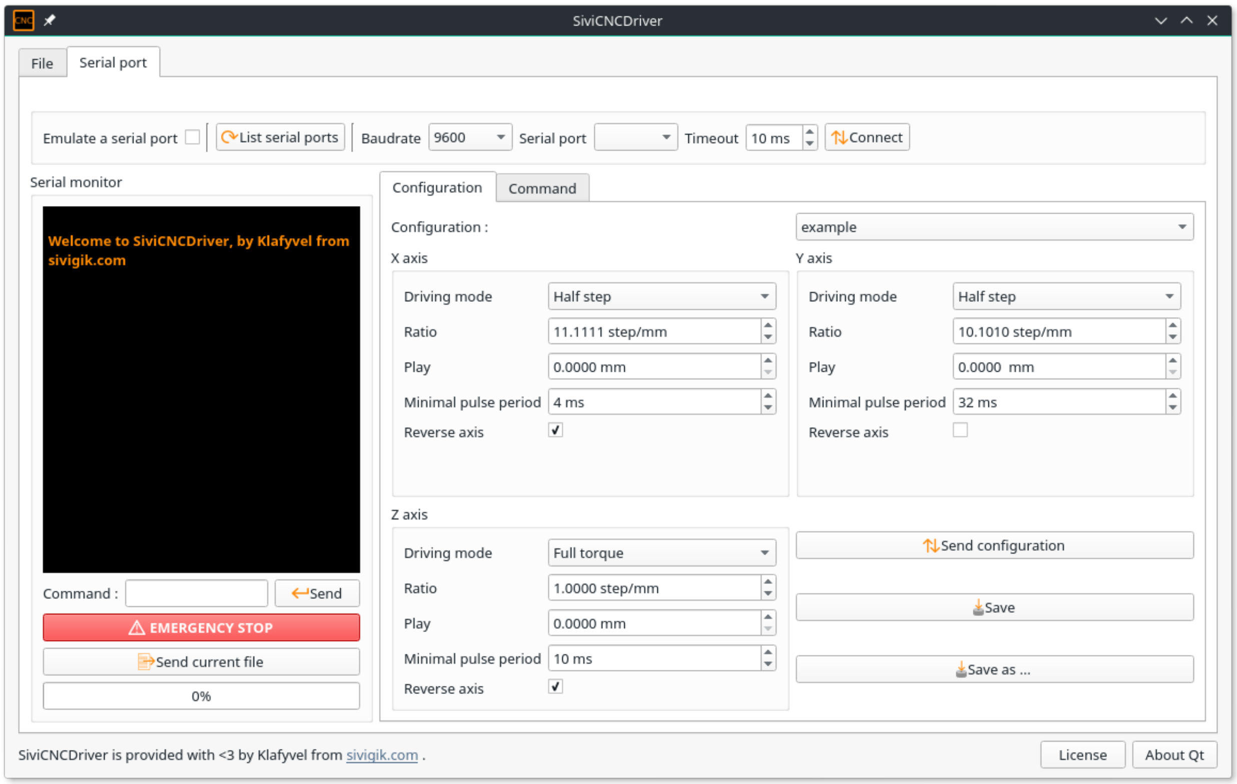
Reverse X axis ☐
Reverse Y axis ☐
Reverse Z axis ☐
Highlight current line ☑

X size 58.39
Y size 58.39
Z size 5.125

You can control the tilt of the figure with the left button of the mouse and zoom with the right button.

```
(Header)
(Generated by gcodetools from Inkscape.)
(Using default header. To add your own header create file
"header" in the output dir.)
M3
(Header end.)
G21
(All units in mm)
(Start cutting path id: circle4713)
(Change tool to Default tool)
G0 Z5.0
G0 X32.0191 Y29.197000000000003
G1 Z-0.125
(Penetrate)
G2 X31.804299999999998 Y28.116999999999997 Z-0.125
I-2.8222 J0.0
G2 X31.192500000000003 Y27.2014 Z-0.125 I-2.6074 J1.08
G2 X30.2769 Y26.589599999999997 Z-0.125 I-1.9956
J1.9956
G2 X29.1969 Y26.3748 Z-0.125 I-1.08 J2.6074
G2 X27.2013 Y27.2014 Z-0.125 I0.0 J2.8222
G2 X26.3747 Y29.197000000000003 Z-0.125 I1.9956
J1.9956
G2 X27.2013 Y31.1926 Z-0.125 I2.8222 J0.0
G2 X29.1969 Y32.0192 Z-0.125 I1.9956 J-1.9956
G2 X30.2769 Y31.8044 Z-0.125 I0.0 J-2.8222
G2 X31.192500000000003 Y31.1926 Z-0.125 I-1.08 J-2.6074
G2 X31.804299999999998 Y30.277 Z-0.125 I-1.9956
J-1.9956
```

SiviCNCDriver is provided with <3 by Klafyvel from sivigik.com .



Emulate a serial port ☐ | Baudrate 9600 | Serial port | Timeout 10 ms

```
Welcome to SiviCNCDriver, by Klafyvel from
sivigik.com
```

Command :

0%

Configuration : example

X axis

Driving mode	Half step
Ratio	11.1111 step/mm
Play	0.0000 mm
Minimal pulse period	4 ms
Reverse axis	✓

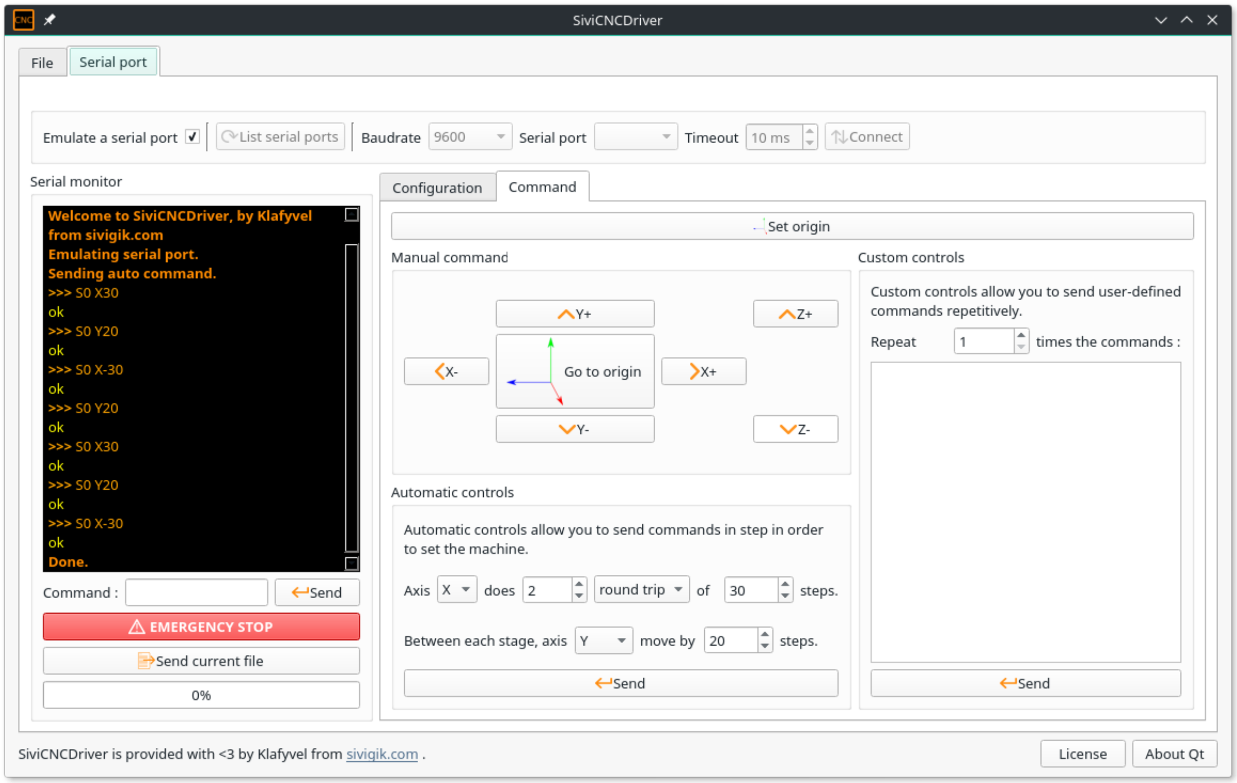
Y axis

Driving mode	Half step
Ratio	10.1010 step/mm
Play	0.0000 mm
Minimal pulse period	32 ms
Reverse axis	☐

Z axis

Driving mode	Full torque
Ratio	1.0000 step/mm
Play	0.0000 mm
Minimal pulse period	10 ms
Reverse axis	✓

SiviCNCDriver is provided with <3 by Klafyvel from sivigik.com .



Emulate a serial port ✓ | Baudrate 9600 | Serial port | Timeout 10 ms

```
Welcome to SiviCNCDriver, by Klafyvel
from sivigik.com
Emulating serial port.
Sending auto command.
>>> S0 X30
ok
>>> S0 Y20
ok
>>> S0 X-30
ok
>>> S0 Y20
ok
>>> S0 X30
ok
>>> S0 Y20
ok
>>> S0 X-30
ok
Done.
```

Command :

0%

Automatic controls allow you to send commands in step in order to set the machine.

Axis X does 2 round trip of 30 steps.

Between each stage, axis Y move by 20 steps.

Custom controls allow you to send user-defined commands repetitively.

Repeat 1 times the commands :

SiviCNCDriver is provided with <3 by Klafyvel from sivigik.com .

CHAPTER 3

Installation

3.1 Using pip

On any operating system with a python and pip installated, use pip (you may need superuser privilege)

```
pip install sivicncdriver
```

Then you should be able to run the program with a simple:

```
sivicnc
```

You can get the development version using pip, although it is not recommended.

```
pip install git+git://github.com/Klafyvel/SiviCNCDriver
```

3.2 Binary distribution (Windows)

If, for some reasons, you can't or don't want to use pip, a binary is available here .

CHAPTER 4

Contribute

The project has its own Git repository on GitHub.

You nill need `virtualenv`

```
pip install --user virtualenv
```

Create a directory in which we will work.

```
mkdir SiviCNCDriver
cd SiviCNCDriver
```

Clone the project

```
git clone https://github.com/Klafyvel/SiviCNCDriver.git
```

Then create the virtual environment

```
virtualenv ENV
```

Activate it

```
source ENV/bin/activate
```

Download the dependencies

```
cd SiviCNCDriver
pip install -r requirements.txt
```

You can code ! To test the code, run the application as package

```
python -m sivicncdriver
```

If you need to re-create the ui after editing it with QtCreator, you can use *make_ui.sh* or directly *pyuic5*.

CHAPTER 5

Custom G-Codes

SiviCNCDriver uses several custom G-Codes, they may change in the future.

Command	Explanation
`S0 Xnnn Ynnn Znnn`	Perform a straight line with `nnn` in steps on the given axes. A negative number make the axis go backward.
`S1 X Y Z`	Trigger continuous advancement forward on the given axes.
`S2 X Y Z`	Trigger continuous advancement backward on the given axes.
`S3 X Y Z`	Stop continuous advancement (if exists) on the given axes.
`S5 X Y Z`	Set driving mode to normal on the given axes.
`S6 X Y Z`	Set driving mode to max torque on the given axes.
`S7 X Y Z`	Set driving mode to half steps on the given axes.
`S8 Xnnn Ynnn Znnn`	Set the play of the given axes, with `nnn` in millimeters.
`S9 X Y Z`	Set the given axes sense to reverse.
`S10 X Y Z`	Set the given axes sense to normal.
`S11 Xnnn Ynnn Znnn`	Set the minimal duration between two pulses for the given axes.

CHAPTER 6

License

CHAPTER 7

sivicncdriver

7.1 sivicncdriver package

7.1.1 Subpackages

sivicncdriver.gcode package

Submodules

sivicncdriver.gcode.arc_calculator module

The arc_calculator module

It creates small segments from an arc given with G-codes parameters : origin, end, path to center and sense of rotation.

Example

```
>>> from arc_calculator import arc_to_segments
>>> a = arc_to_segments((0,0),(5,0),(10,0))
>>> for x,y in a:
...     print((x,y))
...
(0.0, 6.12323e-16)
(0.09966, -0.993334)
(0.39469, -1.94708)
(0.87331, -2.8232)
(1.51646, -3.58677)
(2.29848, -4.20735)
(3.1882, -4.66019)
(4.15015, -4.92725)
(5.14598, -4.99787)
(6.136, -4.86924)
(7.08072, -4.54649)
```

```
(7.94249, -4.04249)
(8.68696, -3.37733)
(9.28444, -2.57752)
(9.71111, -1.67495)
(10, 0)
```

`sivicncdriver.gcode.arc_calculator.`**`arc_to_segments`**(*start*, *vect_to_center*, *end*, *clockwise=False*, *length=1*)

Creates small segments from an arc.

Uses Decimal for better precision. It yields the vertices.

Parameters

- **`start`** (*`A float tuple`*) – The starting position
- **`vect_to_center`** (*`A float tuple`*) – A vector to go to the center of the arc from the starting position
- **`end`** (*`A float tuple`*) – The ending position
- **`clockwise`** (*`bool`*) – Should it go clockwise ?
- **`length`** (*`float`*) – length of the segments

Returns None, it yields vertices.

sivicncdriver.gcode.gcode module

A module to parse G-codes.

`sivicncdriver.gcode.gcode.`**`parse`**(*gcode*)

Parse gcode.

It yields a dict for each line with :

name name of the code (G, M)

value an integer

args a dict with the code arguments, ex : {‘Y’:3.0}

Parameters **`gcode`** (*`str`*) – The gcode which is to be parsed.

sivicncdriver.gcode.gcode_maker module

A bunch of command to easily create G-Codes.

`sivicncdriver.gcode.gcode_maker.`**`config_as_gcode`**(***kwargs*)

Make a set of commands to save the configuration.

Parameters

- **`x_ratio`** (*`float`*) – The X axis ratio (mm/step)
- **`y_ratio`** (*`float`*) – The Y axis ratio (mm/step)
- **`z_ratio`** (*`float`*) – The Z axis ratio (mm/step)
- **`x_drive`** (*`int`*) – X axis drive mode (0:normal, 1:full torque, 2:half step)
- **`y_drive`** (*`int`*) – Y axis drive mode (0:normal, 1:full torque, 2:half step)

- **`z_drive`** (*`int`*) – Z axis drive mode (0:normal, 1:full torque, 2:half step)
- **`x_play`** (*`float`*) – X axis play
- **`y_play`** (*`float`*) – Y axis play
- **`z_play`** (*`float`*) – Z axis play
- **`x_reverse`** (*`bool`*) – Should the X axis be reversed ?
- **`y_reverse`** (*`bool`*) – Should the Y axis be reversed ?
- **`z_reverse`** (*`bool`*) – Should the Z axis be reversed ?
- **`x_min_time`** (*`int`*) – The minimal duration between 2 pulse for the x axis in milliseconds.
- **`y_min_time`** (*`int`*) – The minimal duration between 2 pulse for the y axis in milliseconds.
- **`z_min_time`** (*`int`*) – The minimal duration between 2 pulse for the z axis in milliseconds.

`sivicncdriver.gcode.gcode_maker.`**`emergency_stop`**`()`
Stop every axis.

`sivicncdriver.gcode.gcode_maker.`**`goto_origin`**`()`
Go to the origin.

`sivicncdriver.gcode.gcode_maker.`**`set_origin`**`()`
Register the current position as the origin.

`sivicncdriver.gcode.gcode_maker.`**`start_continuous`**`(`*axis*, *direction='forward'*`)`
Start a continuous movement in the given direction. :param axis: The axis which is to move. :param direction: The direction. :type axis: str :type direction: str

`sivicncdriver.gcode.gcode_maker.`**`start_continuous_x_backward`**`()`
Start a continuous movement on X axis backward

`sivicncdriver.gcode.gcode_maker.`**`start_continuous_x_forward`**`()`
Start a continuous movement on X axis forward.

`sivicncdriver.gcode.gcode_maker.`**`start_continuous_y_backward`**`()`
Start a continuous movement on Y axis backward

`sivicncdriver.gcode.gcode_maker.`**`start_continuous_y_forward`**`()`
Start a continuous movement on Y axis forward.

`sivicncdriver.gcode.gcode_maker.`**`start_continuous_z_backward`**`()`
Start a continuous movement on Z axis backward

`sivicncdriver.gcode.gcode_maker.`**`start_continuous_z_forward`**`()`
Start a continuous movement on Z axis forward.

`sivicncdriver.gcode.gcode_maker.`**`step`**`(`*axis*, *n*`)`
Moves the given axis of n steps.

Parameters

- **`axis`** (*`str`*) – The axis
- **`n`** (*`int`*) – number of steps

`sivicncdriver.gcode.gcode_maker.`**`step_x`**`(`*n*`)`
Moves the X axis oh n steps. :param n: The number of steps :type n: int

`sivicncdriver.gcode.gcode_maker.`**`step_y`**`(`*n*`)`
Moves the Y axis oh n steps. :param n: The number of steps :type n: int

`sivicncdriver.gcode.gcode_maker.`**`step_z`**`(`*n*`)`
Moves the Z axis oh n steps. :param n: The number of steps :type n: int

`sivicncdriver.gcode.gcode_maker.`**`stop`**`(`*axis*`)`
Stop any movement on the given axis.

`sivicncdriver.gcode.gcode_maker.`**`stop_x`**`()`
Stop any movement on the X axis.

`sivicncdriver.gcode.gcode_maker.`**`stop_y`**`()`
Stop any movement on the Y axis.

`sivicncdriver.gcode.gcode_maker.`**`stop_z`**`()`
Stop any movement on the Z axis.

Module contents

sivicncdriver.serial package

Submodules

sivicncdriver.serial.serial_list module

`sivicncdriver.serial.serial_list.`**`serial_ports`**`()`
Lists serial ports

Raises **`EnvironmentError`** – On unsupported or unknown platforms

Returns A list of available serial ports

sivicncdriver.serial.serial_manager module

The serial_manager module

Provides a class to handle the CNC machine through a serial object.

class `sivicncdriver.serial.serial_manager.`**`SerialManager`**`(`*serial*, *fake_mode=False*`)`
Bases: `PyQt5.QtCore.QObject`

A class to manage the serial port.

It will try to send what it receive and send via the send_print signal. When it receive a 'ok' from the serial it will send the send_confirm signal.

`close``()`
Closes the serial port.

`open``(`*baudrate*, *serial_port*, *timeout*`)`
Opens the serial port with the given parameters.

Parameters

- **`baudrate`** – The baudrate.
- **`serial_port`** – The port to be used.
- **`timeout`** – Timeout for reading and writing.

readMsg()
Reads a line from the serial port. And emit the send_print or send_confirm signals if necessary.

sendMsg(*msg*)
Sends a message using the serial port if fake_mode is False.

Parameters **msg** – The message to be sent.

Returns True if no error occurred, else False.

send_confirm

send_print

serial_fatal_error

sivicncdriver.serial.thread_read module

class `sivicncdriver.serial.thread_read.`**ReadThread**
Bases: `PyQt5.QtCore.QThread`

A thread to read the serial link.

run()
Runs the thread.

The commands are sent using the serial manager. If an error occurs or if the thread is stopped by the user, then it quits.

set_read_allowed(*st*)
Allows or not the thread to read.

Parameters **st** – Is it allowed ?

stop()
A simple slot to tell the thread to stop.

read

sivicncdriver.serial.thread_send module

class `sivicncdriver.serial.thread_send.`**SendThread**(*serial_manager*, *gcode*)
Bases: `PyQt5.QtCore.QThread`

A thread to send a list of instructions without blocking the main thread.

confirm(*st*)
Receive confirmation from the readThread.

Parameters **st** – Everything ok ?

run()
Runs the thread.

The commands are sent using the serial manager. If an error occurs or if the thread is stopped by the user, then it quits.

stop()
A simple slot to tell the thread to stop.

read_allowed

update_progress

Module contents

sivicncdriver.ui package

Submodules

sivicncdriver.ui.interface module

The interface module

Provides the MainWindow class.

class `sivicncdriver.ui.interface.`**`MainWindow`**

Bases: `PyQt5.QtWidgets.QMainWindow`, *`sivicncdriver.ui.main_window.Ui_MainWindow`*

The main window of the application.

`about_license``()`
Displays informations about the license.

`about_qt``()`
Displays informations about Qt.

`auto_cmd``()`
Sends auto commands using a thread if they are too long.

`choose_file``()`
Sets the gcode file.

`close_file``()`
Close the current file.

`config_as_dict``()`
Get the configuration as a dict.

Returns The configuration as a dict.

Return type dict

`connectUi``()`
Connects The UI signals and slots.

`draw_file``(`*gcode=None*`)`
Draws a gcode file.

Parameters **`gcode`** – gcode to use in place of the one form code_edit.

`emergency_stop``()`

`end_preprocessor``()`
Manages the end of the preprocessing interface.

`goto_origin``()`

`highlight_selected_path``()`
Looks for selected line in the code_edit, then updates the drawing to highlight the corresponding path.

list_configs()
Lists available configurations.

list_serials()
Lists available serials ports.

load_file()
Loads a gcode file.

manage_auto_cmd_number(*n*)
Enable the widgets for auto commands

manage_connection()
Manages the connection widgets.

manage_emulate_serial_port(*s*)
Enable widgets for serial port emulation.

parse_error(*line*)
Handles parsing errors.

Parameters **line** – The line where the error occurred.

print(*txt*, *msg_type='operator'*)
Prints a message on the application console.

Parameters

- **txt** (`str`) – The message
- **msg_type** (`str`) – The type of the message. Can be “operator”, “machine”, “error” or “info”

reset_config()
Resets the configuration.

run_custom_cmd()
Sends a custom command using a thread.

run_preprocessor()
Runs the preprocessor dialog.

run_thread(*gcode*, *n=None*, *disable=True*, *allow_waiting=True*)
Run a thread to send the given gcode.

Parameters

- **gcode** (`list`) – The gcode as a list of commands.
- **n** (`int`) – A length for the sending_process progress bar.
- **disable** (`bool`) – Disable ui elements which trigger sending.
- **allow_waiting** (`bool`) – Adds the command to the waiting queue.

save_config(*filename=None*)
Saves a configuration.

Parameters **filename** (`str`) – The name of the file.

save_config_as()
Saves a configuration in a new file.

save_file()
Saves a gcode file.

save_file_as()
: Saves a gcode file in a nex file.

send_cmd()
: Sends an user command using a thread.

send_config()
: Send a configuration to the machine.

send_file()
: Send a file using a different thread.

sending_end()
: Manages the end of upload. If some commands are waiting, run them at the end.

set_origin()

set_serial_mode(*mode*)
: Change serial mode.

  Parameters **mode** (*str*) – can be “manual” or “file”

start_continuous_x_backward()

start_continuous_x_forward()

start_continuous_y_backward()

start_continuous_y_forward()

start_continuous_z_backward()

start_continuous_z_forward()

stop_x()

stop_y()

stop_z()

update_config(*i*)
: Updates the configuration widgets.

update_drawing(*highlight_line=None*)
: Updates the drawing.

  Parameters **highlight_line** (*int*) – A line which is to be highlighted.

update_progress(*s*)
: Updates the progress bar.

sivicncdriver.ui.main_window module

class sivicncdriver.ui.main_window.**Ui_MainWindow**
: Bases: object

  retranslateUi(*MainWindow*)

  setupUi(*MainWindow*)

sivicncdriver.ui.preprocessor module

The preprocessor module

Provides the PreprocessorDialog class.

class `sivicncdriver.ui.preprocessor.`**`PreprocessorDialog`**`(`*gcode*, *parent=None*`)`

Bases: `PyQt5.QtWidgets.QDialog`, *`sivicncdriver.ui.preprocessor_window.Ui_dialog`*

The preprocessor dialog.

`accept``()`

`cancel``()`

`get_minimize_bounding_box``()`

Computes a new origin for the drawing.

`remove_useless``()`

Remove useless things in the code according to the UI options.

`run_preprocessor``()`

Runs the preprocessor on the G-Code.

sivicncdriver.ui.preprocessor_window module

class `sivicncdriver.ui.preprocessor_window.`**`Ui_dialog`**

Bases: `object`

`retranslateUi``(`*dialog*`)`

`setupUi``(`*dialog*`)`

sivicncdriver.ui.ressources_rc module

`sivicncdriver.ui.ressources_rc.`**`qCleanupResources`**`()`

`sivicncdriver.ui.ressources_rc.`**`qInitResources`**`()`

sivicncdriver.ui.view3d module

class `sivicncdriver.ui.view3d.`**`View3D`**`(`*parent=None*, *width=5*, *height=4*, *dpi=100*`)`

Bases: `matplotlib.backends.backend_qt5agg.FigureCanvasQTAgg`

Prints G-Codes in 3D.

`compute_data``(`*gcode*`)`

Computes the paths generated by a gcode file.

Parameters **`gcode`** (*`str`*) – The gcode.

`draw``(`***kwargs*`)`

Parameters

- **`reverse_x`** (*`bool`*) – Should the x axis be reversed ? (default : False)
- **`reverse_y`** (*`bool`*) – Should the y axis be reversed ? (default : False)

- **reverse_z** (*bool*) – Should the z axis be reversed ? (default : False)
- **highlight_line** (*int*) – A line which is to be highlighted. (default : None)

get_bounds()

Returns the maximum and the minimum value on each axis.

parse_error

Module contents

7.1.2 Submodules

7.1.3 sivicncdriver.app module

`sivicncdriver.app.`**main**()

The main function of the application.

It will create a QApplication and a main window then run the application and exit.

7.1.4 sivicncdriver.settings module

7.1.5 Module contents

The SiviCNCDriver Package

This is the SiviCNCDriver package. To run the application directly you should use the `sivicnc` command in a shell. Alternately you can use the `main` function of the package which doesn't take any parameter.

Example

```
>>> from sivicncdriver.app import main
>>> main()
```

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index