
SiviCNCDriver Documentation

Release 0.1.8

Klafyvel

Aug 23, 2017

Contents:

1	What is SiviCNCDriver ?	1
2	Screenshots	3
3	Installation	5
3.1	Using pip	5
3.2	Binary distribution (Windows)	5
4	Contribute	7
5	Custom G-Codes	9
6	License	11
7	sivicnedriver	13
7.1	sivicnedriver package	13
8	Indices and tables	21
	Python Module Index	23

CHAPTER 1

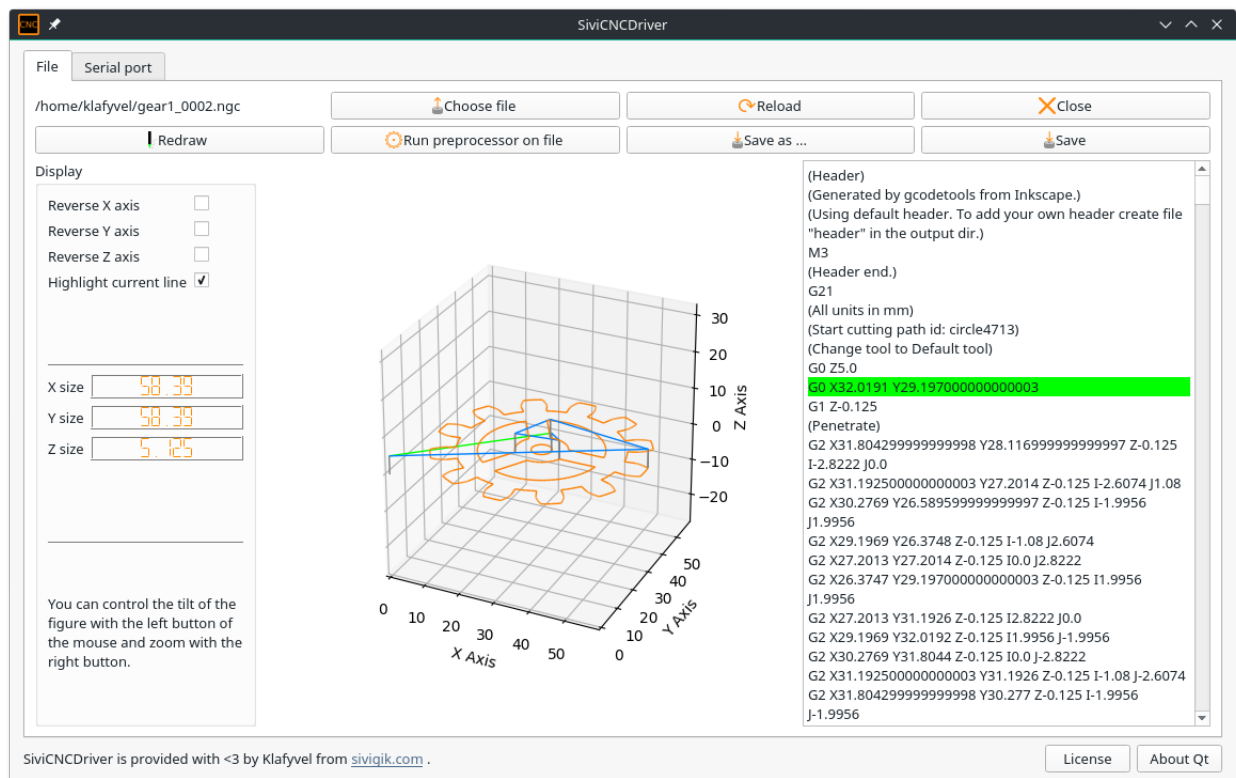
What is SiviCNCDriver ?

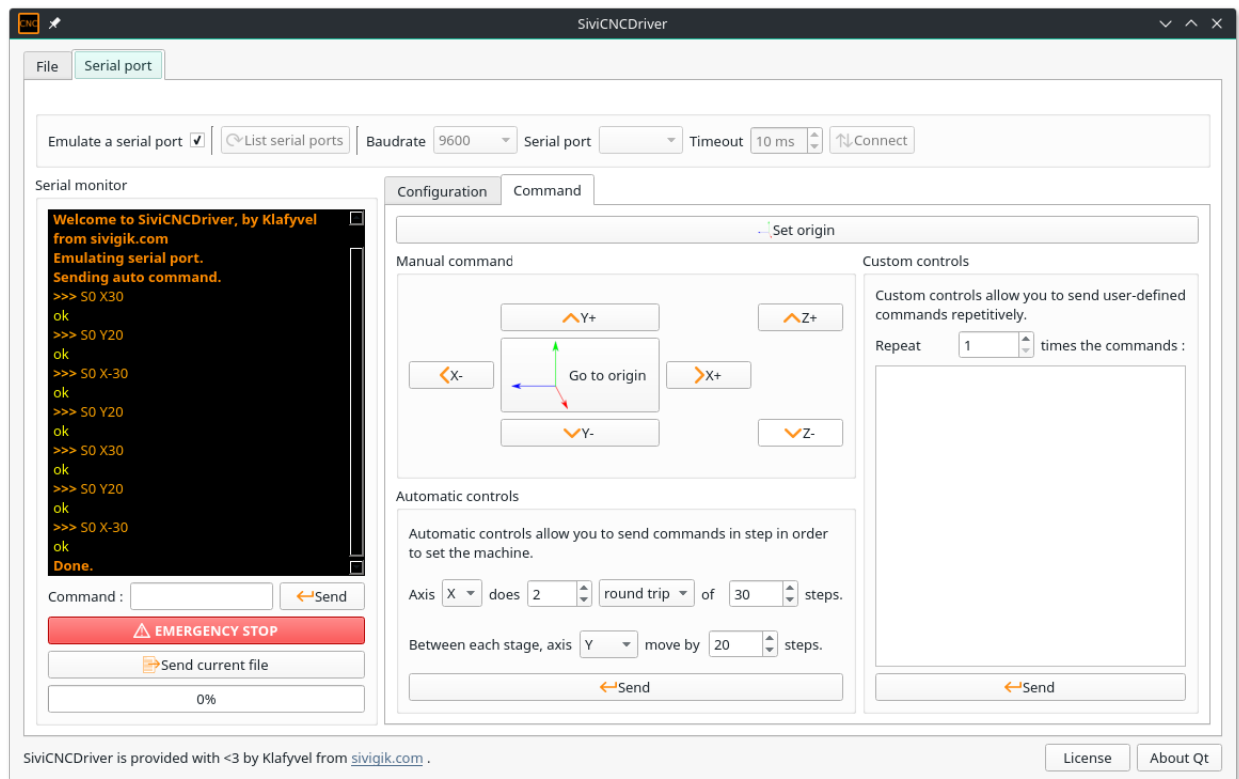
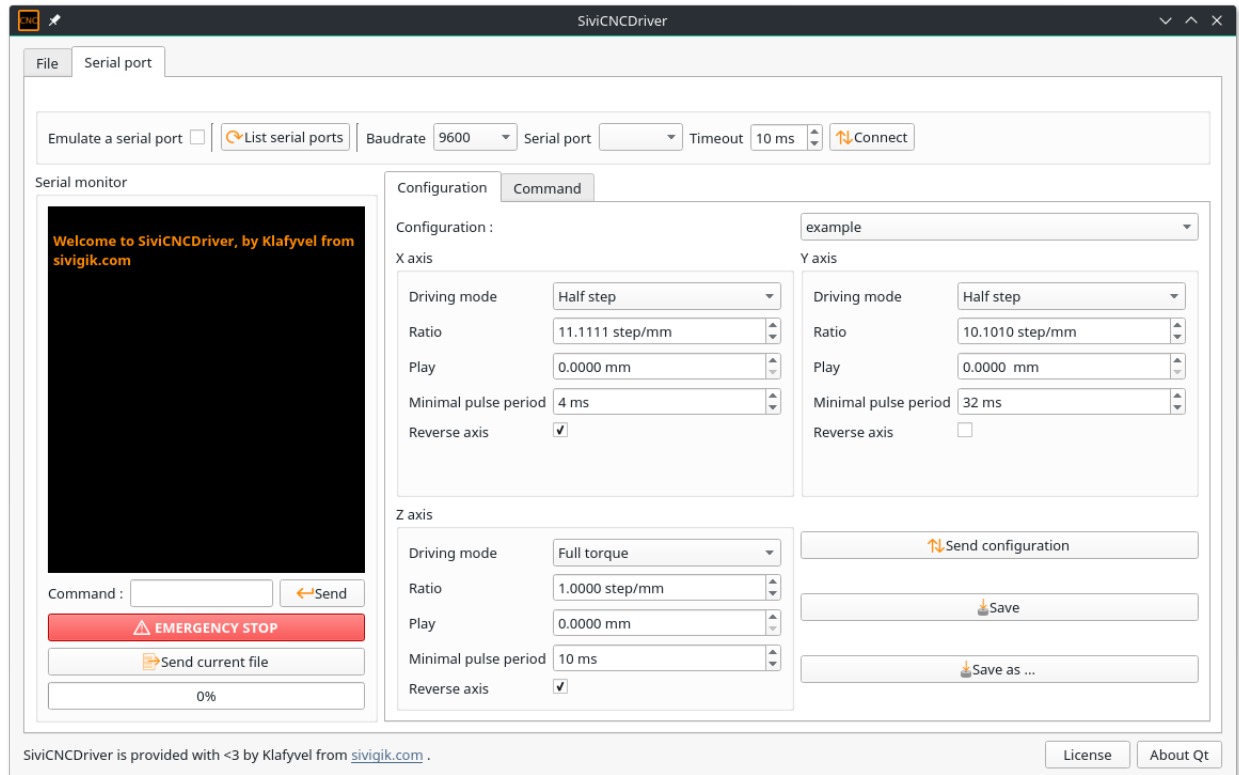
SiviCNCDriver, as its name lets you guess, is designed to drive a CNC. What does it do ?

- 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





Using pip

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

```
pip install sivicndriver
```

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
```

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 will 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

SiviCNCDriver Copyright (C) 2017 Hugo LEVY-FALK

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

sivicncdriver package

Subpackages

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 from code_edit.

emergency_stop()

end_preprocessor()

Manages the end of the preprocessing interface.

goto_origin()

highlight_selected_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. :param gcode: The gcode as a list of commands. :param n: a length for the sending_process. :param disable: Should the ui elements which trigger sending be

disabled ?

Parameters allow_waiting(*bool*) – If True and a thread is already running, wait for it to end before sending the command. Else stop the current thread.

save_config(*filename=None*)

Saves a configuration. :param filename: The name of the file. :type filename: str

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 new 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.

sivincndriver.ui.main_window module

class `sivincndriver.ui.main_window.Ui_MainWindow`

Bases: `object`

retranslateUi (*MainWindow*)

setupUi (*MainWindow*)

sivincndriver.ui.preprocessor module

The preprocessor module

Provides the PreprocessorDialog class.

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

Bases: `PyQt5.QtWidgets.QDialog`, `sivincndriver.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.

sivincndriver.ui.preprocessor_window module

class `sivincndriver.ui.preprocessor_window.Ui_dialog`

Bases: `object`

retranslateUi (*dialog*)

setupUi (*dialog*)

sivincndriver.ui.ressources_rc module

`sivincndriver.ui.ressources_rc.qCleanupResources` ()

`sivincndriver.ui.ressources_rc.qInitResources` ()

Module contents

Submodules

sivincndriver.app module

`sivincndriver.app.main()`

The main function of the application.

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

sivincndriver.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)
```

`sivincndriver.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.

sivincndriver.gcode module

A module to parse G-codes.

`sivincndriver.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.

sivincndriver.gcode_maker module

A bunch of command to easily create G-Codes.

`sivincndriver.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.

`sivincndriver.gcode_maker.emergency_stop()`

Stop every axis.

`sivincndriver.gcode_maker.goto_origin()`

Go to the origin.

`sivincndriver.gcode_maker.set_origin()`

Register the current position as the origin.

`sivincndriver.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

`sivincndriver.gcode_maker.start_continuous_x_backward()`

Start a continuous movement on X axis backward

`sivincndriver.gcode_maker.start_continuous_x_forward()`

Start a continuous movement on X axis forward.

`sivincndriver.gcode_maker.start_continuous_y_backward()`

Start a continuous movement on Y axis backward

`sivincndriver.gcode_maker.start_continuous_y_forward()`

Start a continuous movement on Y axis forward.

`sivincndriver.gcode_maker.start_continuous_z_backward()`

Start a continuous movement on Z axis backward

`sivincndriver.gcode_maker.start_continuous_z_forward()`

Start a continuous movement on Z axis forward.

`sivincndriver.gcode_maker.step(axis, n)`

Moves the given axis of n steps.

Parameters

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

`sivincndriver.gcode_maker.step_x(n)`

Moves the X axis oh n steps. :param n: The number of steps :type n: int

`sivincndriver.gcode_maker.step_y(n)`

Moves the Y axis oh n steps. :param n: The number of steps :type n: int

`sivincndriver.gcode_maker.step_z(n)`

Moves the Z axis oh n steps. :param n: The number of steps :type n: int

`sivincndriver.gcode_maker.stop(axis)`

Stop any movement on the given axis.

`sivincndriver.gcode_maker.stop_x()`

Stop any movement on the X axis.

`sivincndriver.gcode_maker.stop_y()`

Stop any movement on the Y axis.

`sivincndriver.gcode_maker.stop_z()`

Stop any movement on the Z axis.

sivincndriver.serial_list module

`sivincndriver.serial_list.serial_ports()`

Lists serial ports

Raises `EnvironmentError` – On unsupported or unknown platforms

Returns A list of available serial ports

sivincndriver.serial_manager module

The serial_manager module

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

```
class sivincndriver.serial_manager.SerialManager (serial, fake_mode=False)
    Bases: PyQt5.QtCore.QObject
        close ()
        open (baudrate, serial_port, timeout)
        readMsg ()
        sendMsg (msg)
        send_confirm
        send_print
```

sivincndriver.settings module

sivincndriver.thread_send module

```
class sivincndriver.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 ?
        read_allowed
        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.
        update_progress
```

Module contents

The SiviCNCDriver Package

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

Example

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


CHAPTER 8

Indices and tables

- `genindex`
- `modindex`
- `search`

S

- `sivicncdriver`, [20](#)
- `sivicncdriver.app`, [17](#)
- `sivicncdriver.arc_calculator`, [17](#)
- `sivicncdriver.gcode`, [18](#)
- `sivicncdriver.gcode_maker`, [18](#)
- `sivicncdriver.serial_list`, [19](#)
- `sivicncdriver.serial_manager`, [20](#)
- `sivicncdriver.settings`, [20](#)
- `sivicncdriver.thread_send`, [20](#)
- `sivicncdriver.ui`, [17](#)
- `sivicncdriver.ui.interface`, [13](#)
- `sivicncdriver.ui.main_window`, [16](#)
- `sivicncdriver.ui.preprocessor`, [16](#)
- `sivicncdriver.ui.preprocessor_window`,
[16](#)
- `sivicncdriver.ui.ressources_rc`, [16](#)

A

about_license() (sivicncdriver.ui.interface.MainWindow method), 13
about_qt() (sivicncdriver.ui.interface.MainWindow method), 13
accept() (sivicncdriver.ui.preprocessor.PreprocessorDialog method), 16
arc_to_segments() (in module sivicncdriver.arc_calculator), 17
auto_cmd() (sivicncdriver.ui.interface.MainWindow method), 13

C

cancel() (sivicncdriver.ui.preprocessor.PreprocessorDialog method), 16
choose_file() (sivicncdriver.ui.interface.MainWindow method), 13
close() (sivicncdriver.serial_manager.SerialManager method), 20
close_file() (sivicncdriver.ui.interface.MainWindow method), 13
config_as_dict() (sivicncdriver.ui.interface.MainWindow method), 14
config_as_gcode() (in module sivicncdriver.gcode_maker), 18
confirm() (sivicncdriver.thread_send.SendThread method), 20
connectUi() (sivicncdriver.ui.interface.MainWindow method), 14

D

draw_file() (sivicncdriver.ui.interface.MainWindow method), 14

E

emergency_stop() (in module sivicncdriver.gcode_maker), 18
emergency_stop() (sivicncdriver.ui.interface.MainWindow method), 14

end_preprocessor() (sivicncdriver.ui.interface.MainWindow method), 14

G

get_minimize_bounding_box() (sivicncdriver.ui.preprocessor.PreprocessorDialog method), 16
goto_origin() (in module sivicncdriver.gcode_maker), 18
goto_origin() (sivicncdriver.ui.interface.MainWindow method), 14

H

highlight_selected_path() (sivicncdriver.ui.interface.MainWindow method), 14

L

list_configs() (sivicncdriver.ui.interface.MainWindow method), 14
list_serials() (sivicncdriver.ui.interface.MainWindow method), 14
load_file() (sivicncdriver.ui.interface.MainWindow method), 14

M

main() (in module sivicncdriver.app), 17
MainWindow (class in sivicncdriver.ui.interface), 13
manage_auto_cmd_number() (sivicncdriver.ui.interface.MainWindow method), 14
manage_connection() (sivicncdriver.ui.interface.MainWindow method), 14
manage_emulate_serial_port() (sivicncdriver.ui.interface.MainWindow method), 14

O

`open()` (sivincndriver.serial_manager.SerialManager method), 20

P

`parse()` (in module sivincndriver.gcode), 18

`parse_error()` (sivincndriver.ui.interface.MainWindow method), 14

`PreprocessorDialog` (class in sivincndriver.ui.preprocessor), 16

`print()` (sivincndriver.ui.interface.MainWindow method), 14

Q

`qCleanupResources()` (in module sivincndriver.ui.ressources_rc), 16

`qInitResources()` (in module sivincndriver.ui.ressources_rc), 16

R

`read_allowed` (sivincndriver.thread_send.SendThread attribute), 20

`readMsg()` (sivincndriver.serial_manager.SerialManager method), 20

`remove_useless()` (sivincndriver.ui.preprocessor.PreprocessorDialog method), 16

`reset_config()` (sivincndriver.ui.interface.MainWindow method), 14

`retranslateUi()` (sivincndriver.ui.main_window.Ui_MainWindow method), 16

`retranslateUi()` (sivincndriver.ui.preprocessor_window.Ui_dialog method), 16

`run()` (sivincndriver.thread_send.SendThread method), 20

`run_custom_cmd()` (sivincndriver.ui.interface.MainWindow method), 14

`run_preprocessor()` (sivincndriver.ui.interface.MainWindow method), 14

`run_preprocessor()` (sivincndriver.ui.preprocessor.PreprocessorDialog method), 16

`run_thread()` (sivincndriver.ui.interface.MainWindow method), 15

S

`save_config()` (sivincndriver.ui.interface.MainWindow method), 15

`save_config_as()` (sivincndriver.ui.interface.MainWindow method), 15

`save_file()` (sivincndriver.ui.interface.MainWindow method), 15

`save_file_as()` (sivincndriver.ui.interface.MainWindow method), 15

`send_cmd()` (sivincndriver.ui.interface.MainWindow method), 15

`send_config()` (sivincndriver.ui.interface.MainWindow method), 15

`send_confirm` (sivincndriver.serial_manager.SerialManager attribute), 20

`send_file()` (sivincndriver.ui.interface.MainWindow method), 15

`send_print` (sivincndriver.serial_manager.SerialManager attribute), 20

`sending_end()` (sivincndriver.ui.interface.MainWindow method), 15

`sendMsg()` (sivincndriver.serial_manager.SerialManager method), 20

`SendThread` (class in sivincndriver.thread_send), 20

`serial_ports()` (in module sivincndriver.serial_list), 19

`SerialManager` (class in sivincndriver.serial_manager), 20

`set_origin()` (in module sivincndriver.gcode_maker), 18

`set_origin()` (sivincndriver.ui.interface.MainWindow method), 15

`set_serial_mode()` (sivincndriver.ui.interface.MainWindow method), 15

`setupUi()` (sivincndriver.ui.main_window.Ui_MainWindow method), 16

`setupUi()` (sivincndriver.ui.preprocessor_window.Ui_dialog method), 16

`sivincndriver` (module), 20

`sivincndriver.app` (module), 17

`sivincndriver.arc_calculator` (module), 17

`sivincndriver.gcode` (module), 18

`sivincndriver.gcode_maker` (module), 18

`sivincndriver.serial_list` (module), 19

`sivincndriver.serial_manager` (module), 20

`sivincndriver.settings` (module), 20

`sivincndriver.thread_send` (module), 20

`sivincndriver.ui` (module), 17

`sivincndriver.ui.interface` (module), 13

`sivincndriver.ui.main_window` (module), 16

`sivincndriver.ui.preprocessor` (module), 16

`sivincndriver.ui.preprocessor_window` (module), 16

`sivincndriver.ui.ressources_rc` (module), 16

`start_continuous()` (in module sivincndriver.gcode_maker), 19

`start_continuous_x_backward()` (in module sivincndriver.gcode_maker), 19

`start_continuous_x_backward()` (sivincndriver.ui.interface.MainWindow method), 15

`start_continuous_x_forward()` (in module sivinc-

driver.gcode_maker), 19
 start_continuous_x_forward() (sivincn-
 driver.ui.interface.MainWindow
 method), 15
 start_continuous_y_backward() (in module sivincn-
 driver.gcode_maker), 19
 start_continuous_y_backward() (sivincn-
 driver.ui.interface.MainWindow
 method), 15
 start_continuous_y_forward() (in module sivincn-
 driver.gcode_maker), 19
 start_continuous_y_forward() (sivincn-
 driver.ui.interface.MainWindow
 method), 15
 start_continuous_z_backward() (in module sivincn-
 driver.gcode_maker), 19
 start_continuous_z_backward() (sivincn-
 driver.ui.interface.MainWindow
 method), 15
 start_continuous_z_forward() (in module sivincn-
 driver.gcode_maker), 19
 start_continuous_z_forward() (sivincn-
 driver.ui.interface.MainWindow
 method), 15
 step() (in module sivincndriver.gcode_maker), 19
 step_x() (in module sivincndriver.gcode_maker), 19
 step_y() (in module sivincndriver.gcode_maker), 19
 step_z() (in module sivincndriver.gcode_maker), 19
 stop() (in module sivincndriver.gcode_maker), 19
 stop() (sivincndriver.thread_send.SendThread
 method), 20
 stop_x() (in module sivincndriver.gcode_maker), 19
 stop_x() (sivincndriver.ui.interface.MainWindow
 method), 15
 stop_y() (in module sivincndriver.gcode_maker), 19
 stop_y() (sivincndriver.ui.interface.MainWindow
 method), 15
 stop_z() (in module sivincndriver.gcode_maker), 19
 stop_z() (sivincndriver.ui.interface.MainWindow
 method), 15

U

Ui_dialog (class in sivincn-
 driver.ui.preprocessor_window), 16
 Ui_MainWindow (class in sivincn-
 driver.ui.main_window), 16
 update_config() (sivincndriver.ui.interface.MainWindow
 method), 15
 update_drawing() (sivincn-
 driver.ui.interface.MainWindow
 method), 15
 update_progress (sivincndriver.thread_send.SendThread
 attribute), 20