How to use the OC Manager2 software

Starting and ending OC Manager2

After reboot, the OC Manager2 will be available in the browser at http://127.0.0.1 (localhost), if the Crontab job was configured respectively.
Therefore, best configure the browser to start with the custom URL http://127.0.0.1

If a static IP address was set, the OC Manager2 can also be opened on a remote computer within the network, simply by typing the IP address in the browser.

In case of problems to catch the errors, open a terminal window and change to the directory /home/pi/OC_manager2:

```
> cd /home/pi/OC_manager2
```

Start R:

```
> R
```

Start OC_manager2:

```
> shiny::runApp()
```

Close the browser window to end OC_manager2

Quit R:

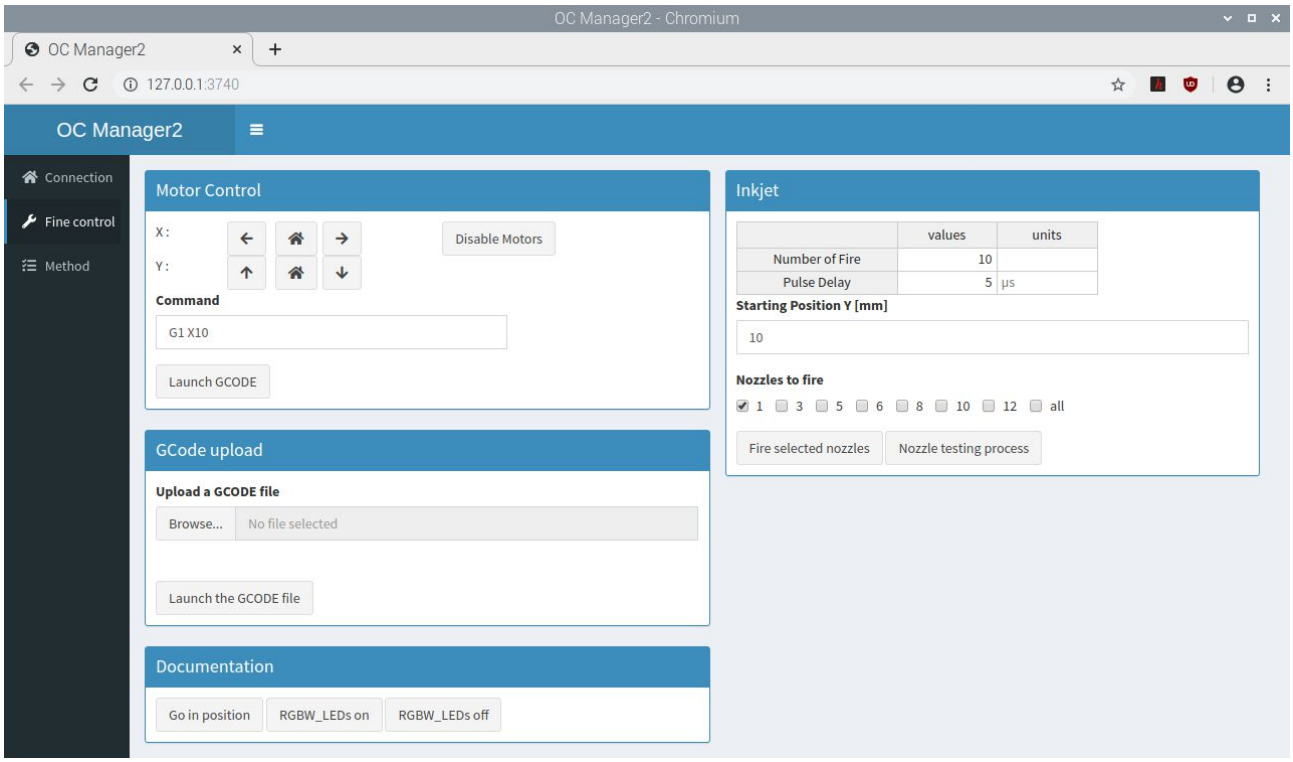
```
> q()
> n
```

The user interface

Connection

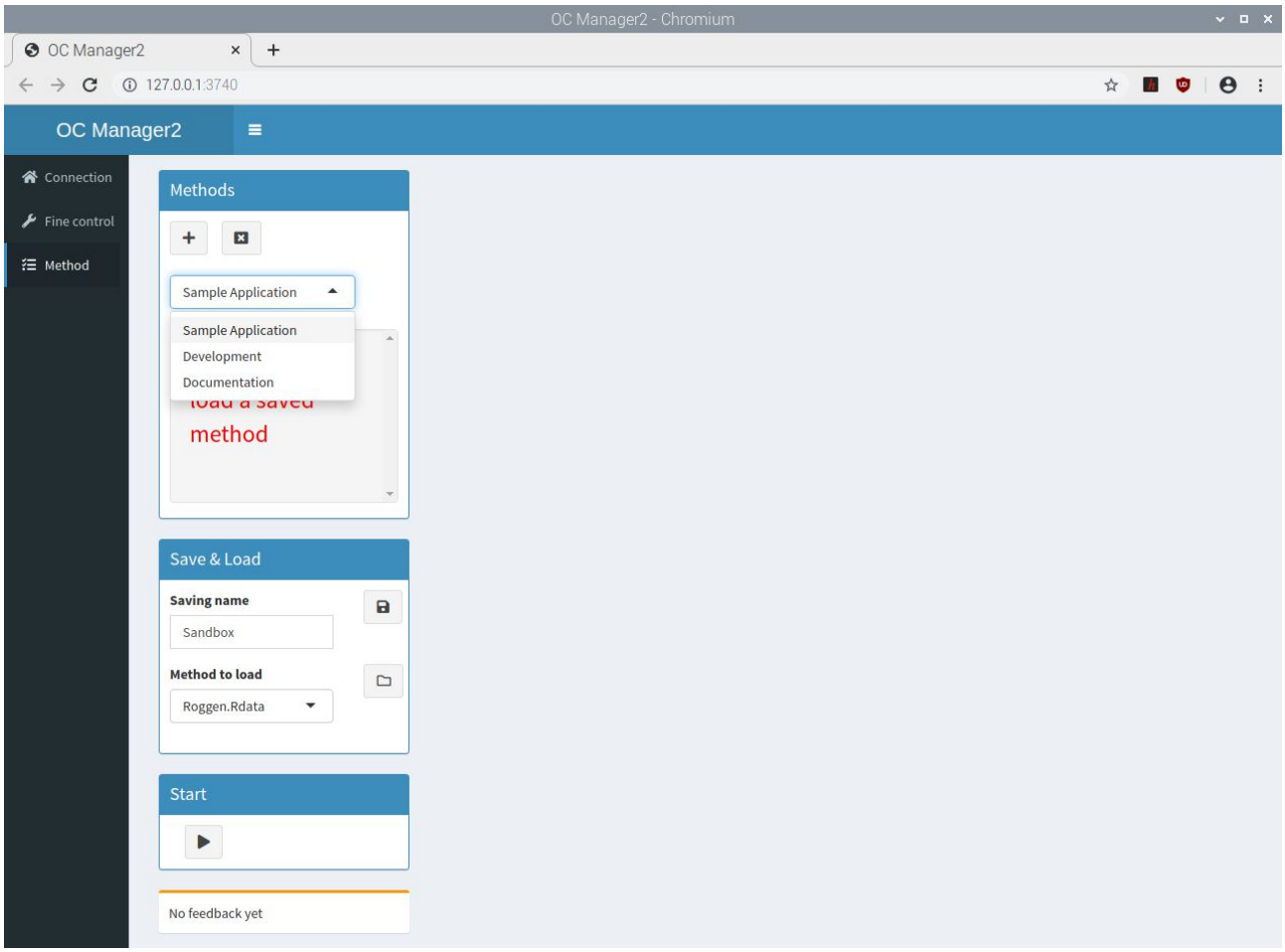
By starting the OC Manager2 it automatically connects the board and shows the Connection window. This process takes a while. If “Connect the Board” label of the button switches to “Disconnect the board”, then the board is successfully connected and you can start working. On the left side you can select other window tabs: FineControl or Method. If you want to minimize the sidebar, you can click on the button at the right of the OC Manager2 label.

Fine control



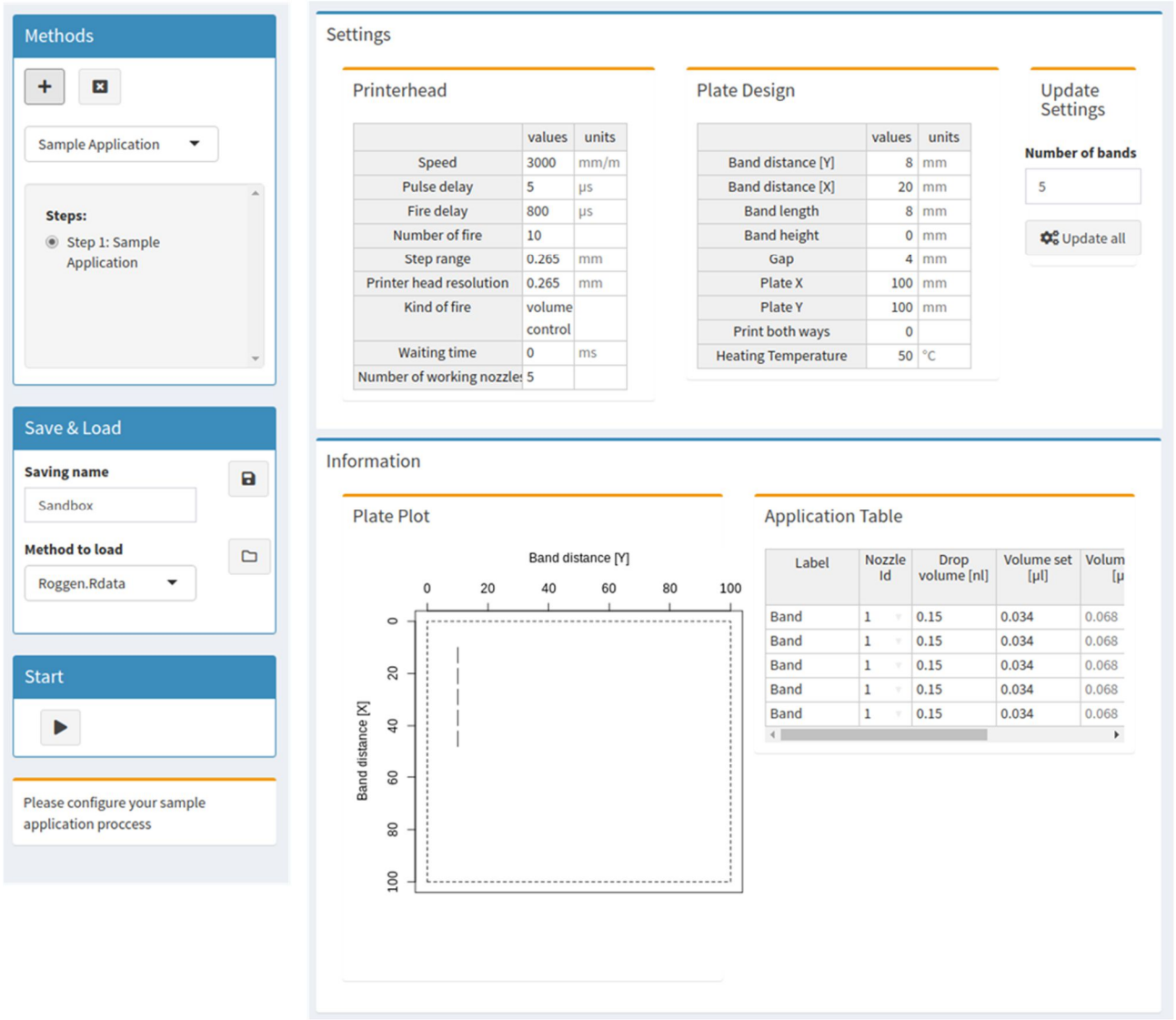
The “Fine control” window is used to manually control all parts of the instrument. In the box of “Motor Control” you can move both axes, X and Y, step by step (5 mm), bring them to their home position or disable the motors.
In the “Command box”, you can insert a GCode command and execute it by clicking Launch GCODE. It is not necessary to write in capital letters.
In the “GCode upload” box, a custom GCODE file can be uploaded and executed. Select the file with the Browse button.
In the “Documentation box”, you can move the plate into the photo position and turn on/off the RGBW LEDs.
In the “Inkjet box”, you can directly fire a nozzle or start a nozzle testing procedure to check the nozzles with chosen parameters Number of Fire (how often the same nozzle fires) and Pulse Delay (a parameter to control the energy). You can also select the Y-Position, where the Nozzle testing process will start.

Methods



From the drop-down menu of the “Methods” window, a Sample Application, Development or Documentation step can be added. Therefore, select the method and press the “+” button to add a new Method. With the “X” button you can delete a selected Method. In the Save & Load Box, you save and load a method with its parameters. The method will be saved as R.data file in /OC_manager2/GUI/method/methods_to_load/.
In the box “Start”, you execute the selected step, when a GCode will be generated and sent to the instrument.

Sample application



The Application method allows to define a pattern of bands of a certain length at a certain distance.

To control the application step, there are three tables of settings:

1. Printerhead
2. Plate Design
3. Application Table

<u>1. Printerhead</u>

The printer head configs defines the parameters specific for the printer head (be careful to change the settings here without understanding the parameters and their effects).

- Speed: how fast the printer head moves.
  Notice in accelerated movements the printer head cannot move faster than 4000 mm/min.
- Pulse delay: Time of the voltage impulse throw the resistor in the inkjet circuit board, which defines the energy.
- Fire delay: the delay between each fire impulse for the same nozzle.
  Number of fire: number of fire impulses in a row on the same position.
- Step range: the distance between each drop position.
- Printer head resolution: the distance between each nozzle in the nozzle plate of the circuit board.

- Kind of fire:
  volume control: printer head waits before firing the nozzle until movement is finished.
  spraying: printer head fires the whole time without waiting (used especially for development).
- Waiting time: time before the next path is printed, to give the sample the opportunity to dry.
- Number of working nozzles: because nozzles can fail, the calculated real volume will be adjusted in the case of firing with all working nozzles.

2. Plate Design

In the Plate Design table, the position of the first band, the band width and band height, the distance between the bands (Gap), the plate size and finally the plate temperature are defined. Then the number of bands have to set, followed by clicking “Update all”. The settings will be updated to driver and displayed in the Information section, the plate layout and the sample application table, which finally has to filled with sample names and sample volumes, followed by “Update all” once more.
In the last three rows (not fully visible in the above printscreen), just information (not writable) is presented, concerning the calculated real volume in µL, which will be applied, and the calculated band start and band end in mm.
The application starts, when you click the “Start” button.

If plate heating is selected, the plate holder should be covered by a glass plate, just leaving space to apply the samples.

However, before starting, the plate holder has to be zeroed.
Therefore,

- first zero the instrument (Home X and Home Y) in the window Fine Control,
- then, using nozzle 1, print a single 6-mm band with a colored compound at Band distance [X] = 20 and Band distance [Y] = 20,
- measure the real distances [X] and [Y],
- open the Python file OC_manager2/oc_driver/OCDriver.py.

In the section:

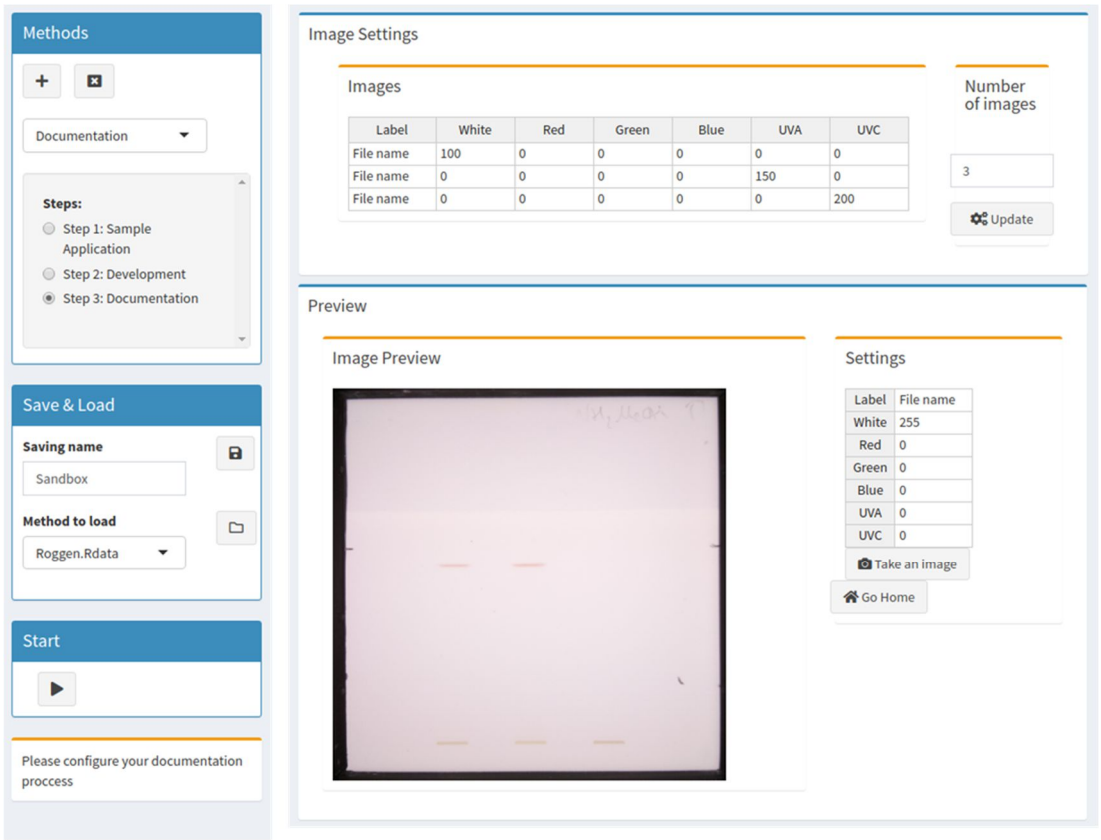
```
DEFAULT_CONFIG = {
   'connection_string': "/dev/ttyACM0",
   'baud_rate': 115200,
   'calibration_x': 5,
   'calibration_y': 6,
   'dpi': 96,
   'log_file_path': "./"
}
```

correct the calibration_x and calibration_y values.

For example: if the measured Band distance [X] was 17 mm instead 20 mm and the measured Band distance [Y] was 22 mm instead of 20 mm, change calibration_x and calibration_y to 3 (5-2) and 8 (6+3), respectively. It is a little bit confusing, but in the scripts, x and y is reversed, corresponding to the real x and y axes of the Plate Plot.
Finally, restart OC_manager2 and check, if the setting s are correct. If it did not work, delete the file OCDriver.py**c** and restart OC_manager2.

Documentation

The method documentation is dedicated to capture images of your plate with different light settings. You can chose between white, red, green, blue, UVA, and UVC LEDs. The intensity of each LED can be selected between 0 and 255, but the UVC LEDs at least require 130. You can test your configuration with the Preview Function. Before you take an image, please home the y_moving (Go Home). You can take a couple of images (Number of images) with different illuminations in a sequence (Image Settings). Each image will be saved with its label in the directory "/home/pi/OC_manager2/www/pictures". Be careful not to overwrite your images; there is no warning.



To obtain optimized images for a special application, special camera settings can be modified. Therefore, open the file OC_manager2/oc_driver/drivers/DocumentationDriver.py (see next page). Camera settings concern, for example, framerates and image effects. To learn more about possible settings, read the official camera guide (https://magpi.raspberrypi.org/books/camera-guide).

To switch on and off the LEDs for camera tests, a respective GCODE is manually sent (tab Fine control), i.e.,M42 P44 SX, M42 P64 SX and M150 RX UX BX WX, for UVA, UVC and RGBW illuminations, where X is the value of intensity between 0 and 255.

To bring the camera settings into effect, save the DocumentationDriver.py file, delete the DocumentationDriver.py**c** file and restart the OC_manager2.

During the first use, the image possibly is not well centered.
If the black borders at the top and at the bottom differ, the cabinet is not centered and has to slightly moved to the right or to the left.
If the black borders at the left and right side differ, the y-value of positioning the plate holder has to be corrected. Therefore, open the file /home/pi/OC_manager2/oc_driver/drivers/gcodes.py and correspondingly change the y-value in GO_TO_FOTO_POSITION = "G1 Y164". Delete the compiled file gcodes.py**c** and restart the OC_manager2.

```
def take_a_picture(self, Path):
    my_file = open(Path, 'w+')
    camera = PiCamera()
    pixel = 2464
    camera.resolution = (pixel, pixel)
    sizes = self.import_from_csv()
    #print(sizes)
    y1 = sizes['width']
    x1 = sizes['height']
    w = int(pixel/(100/x1))
    h = int(pixel/(100/y1))
    c = float(0) # mm cropping
    y = y1-c #cropped width
    x = x1-c #cropped height

    #The illumination type (UVA,UVC) is passed as string.
    #The illumination type is determined in the function "make_pictures_for_documentation"
    #in DocumentationDriver.py.

    if pictureType == 'UVA':
      camera.framerate = Fraction(10, 1)
      z = int(1 / camera.framerate * 1000000)
      camera.shutter_speed = z
      camera.awb_mode = 'incandescent'  #or 'fluorescent' #can improve the contrast
      #Camera warm-up time
      sleep(30)

    elif pictureType == 'UVC':
      camera.framerate = Fraction(30, 1)
      z = int(1 / camera.framerate * 1000000)
      camera.shutter_speed = z
      camera.awb_mode = 'incandescent'
      #Camera warm-up time
      sleep(30)

    else: #visible light
      camera.framerate = Fraction(30, 1)
      z = int(1 / camera.framerate * 1000000)
      camera.shutter_speed = z
      #Camera warm-up time
      sleep(10)

    #Different camera options can be set with the following lines,
    #to be copied into the "if" sections.

    #camera.brightness = 50 #between 0 and 100
    #camera.sharpness = -100 #between -100 and 100
    #camera.saturation = -100 #betweeen -100 and 100
    #camera.exposure_mode = 'verylong' #see picamera documentation
    #camera.awb_mode = 'fluorescent' #see picamera documentation
    #camera.image_effect = 'denoise' # or 'saturation'
    #camera.contrast = -100 #between -100 and 100
    #camera.iso = 800 #see picamera documentation
    #camera.sensor_mode = 3 #see picamera documentation

    camera.exposure_mode = 'off'
    camera.zoom = (c/100, (100-y)/100, (x-c)/100, (100-2*c)/100)
    camera.capture(my_file, quality = 100, resize = (w, h))
    my_file.close()
    sleep(5)
    camera.close()
```