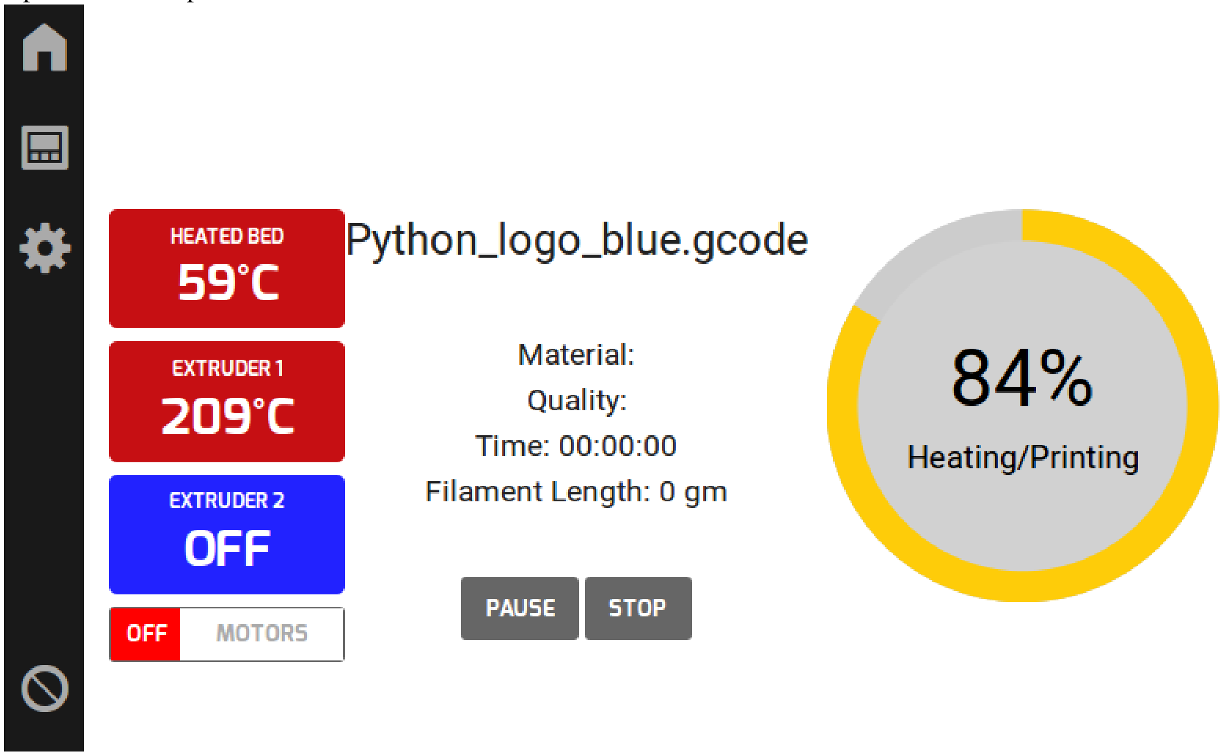
opengb Documentation

Release 0.22.3

re3D

December 30, 2016

OpenGB is a 3D printer control interface.



OpenGB was originally created for the Open Gigabot open-source 3D printer designed and built by re:3D:

- **Interface Design:** Katy Jeremko
- **Backend Development:** James Stewart
- **Frontend Development:** Levi Lalla

OpenGB is also be compatible with other printers running Marlin firmware and a wider range of open-source firmware will be supported in the future.

CHAPTER 1

Installation

OpenGB is a Python application for Linux.

1.1 Using apt

If using Debian or derivates (Ubuntu, Raspbian etc) OpenGB is best installed via a package from the official apt repository.

1.1.1 Add OpenGB Repo

:: sudo apt-key adv –keyserver hkp://keyserver.ubuntu.com:80 –recv-keys 379CE192D401AB61 echo "deb http://dl.bintray.com/amorphic/opengb jessie main" | sudo tee /etc/apt/sources.list.d/opengb.list

1.1.2 Install OpenGB

```
sudo apt-get update
sudo apt-get install opengb
```

1.1.3 Upgrade OpenGB

```
sudo apt-get update
sudo apt-get install --only-upgrade opengb
```

1.1.4 Post-Install

The OpenGB apt package will offer to perform a number of post-installation steps depending on your platform and requirements:

```
$ sudo apt-get install opengb

...

Unpacking opengb (0.21.0) ...
Setting up opengb (0.21.0) ...
Creating opengb user
Creating data directories in /var/opengb
Creating log directory in /var/log/opengb
Creating default config in /etc/opengb
Creating service in /etc/init.d/opengb
To setup the OpenGB graphical interface on this host run 'opengb-interface-setup'
```

1.1.5 Graphical Interface

After installing OpenGB you may run *opengb-interface-setup* to set up the local machine as a graphical frontend to OpenGB. This will present options to:

- Change hostname to `opengb`.
- Install the **Iceweasel** browser.
- Start the browser on boot.
- Install a Fullscreen browser extension.
- Disable screen blanking
- Disable mouse pointer

These steps are useful for configuring platforms such as the recommended Raspberry Pi + Touchscreen.

1.2 Using pip

OpenGB may be installed from scratch using pip. However this requires additional setup which is not neccessary when installing via apt.

1.2.1 Python 3.4

OpenGB requires Python 3.4 or greater.

Install Python3.4:

```
sudo apt-get install python3.4
```

1.2.2 Virtualenv

Install Virtualenv:

```
sudo apt-get install python-virtualenv
```

Create a new Python 3.4 virtualenv:

```
mkdir ~/virtualenvs
virtualenv -p python3.4 ~/virtualenvs/opengb
```

Switch to the new virtualenv:

```
source ~/virtualenvs/opengb/bin/activate
```

1.2.3 Install package

```
pip install opengb
```

1.2.4 Update package

To update OpenGB from a previous version:

```
pip install -U opengb
```

1.2.5 Create data directories

```
sudo mkdir /var/opengb
sudo mkdir /var/opengb/db
sudo mkdir /var/opengb/gcode
sudo chown -R <your_user>:<your_group> /var/opengb
```

1.2.6 Create log directory

```
sudo mkdir /var/log/opengb
sudo chown <your_user>:<your_group> /var/log/opengb
```

1.2.7 Deploy config file

Once deployed edit the config file to set the appropriate parameters for your system.

```
sudo mkdir /etc/opengb
sudo chown <your_user>:<your_group> /etc/opengb
sudo cp ~/virtualenvs/opengb/lib/python3.4/site-packages/opengb/etc/opengb.conf /etc/opengb/
```

1.2.8 Start

Switch to the virtualenv and start opengb:

```
source ~/virtualenvs/opengb/bin/activate
opengb
```

Navigate to http://localhost:8000 and the OpenGB interface should appear.

Configuration

Note: The various configuration steps below are only necessary if installing from scratch via pip. The OpenGB apt package performs these steps automatically post-install.

Note: OpenGB is tested with and designed to run on the Raspberry Pi Model 3B with a Raspberry Pi Touch Display, the Raspbian operating system and the Iceweasel web browser.

The instructions below are specific to this platform but in most cases should be transferrable to any host running a Linux-based operating system and Python 3.4+.

2.1 Autostart

2.1.1 OpenGB Web Interface

To start Iceweasel running the OpenGB web interface fullscreen automatically on boot:

```
sudo apt-get install iceweasel xdotool

cat << EOF >> ~/.config/lxsession/LXDE-pi/autostart
@iceweasel http://opengb.local:8000
@sleep 10
@xdotool key --clearmodifiers F11
EOF
```

2.2 Multicast DNS

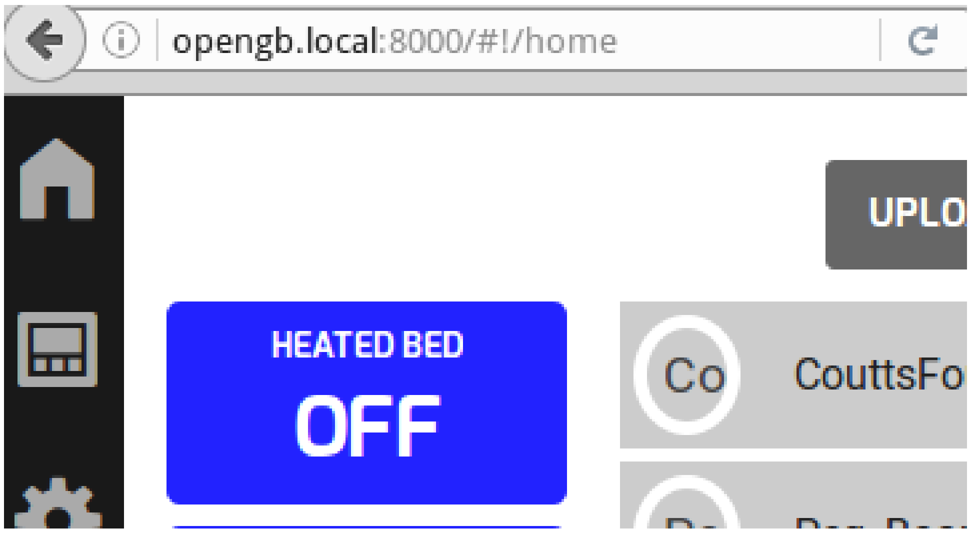
Multicast DNS (also known as “mDNS” or “ZeroConf”) allows a host to be accessed by name rather than IP addresss. Raspbian provides mDNS services via the *avahi-daemon* which is enabled by default.

2.2.1 Change hostname to OpenGB

To change the Raspbian hostname from the default *raspberrypi* to *opengb*:

```
sudo sed -i -e 's/raspberrypi/opengb/g' /etc/hosts
sudo sed -i -e 's/raspberrypi/opengb/g' /etc/hostname
sudo reboot
```

Upon reboot you should be able to navigate to opengb via http://opengb.local:8000.



2.2.2 Multiple OpenGB Instances

If you are running multiple instances of OpenGB on the same network simply choose a different hostname for each instance. E.g.:

OpenGB instance #1:

```
sudo sed -i -e 's/opengb/opengb1/g' /etc/hosts
sudo sed -i -e 's/opengb/opengb1/g' /etc/hostname
sudo reboot
```

OpenGB instance #2:

```
sudo sed -i -e 's/opengb/opengb2/g' /etc/hosts
sudo sed -i -e 's/opengb/opengb2/g' /etc/hostname
sudo reboot
```

OpenGB instance #3:

```
sudo sed -i -e 's/opengb/opengb3/g' /etc/hosts
sudo sed -i -e 's/opengb/opengb3/g' /etc/hostname
sudo reboot
```

2.3 Touchscreen

2.3.1 Screensaver

To disable screen blanking:

```
cat << EOF >> ~/.config/lxsession/LXDE-pi/autostart
@xset s noblank
@xset s off
@xset -dpms
EOF
```

2.3.2 Pointer

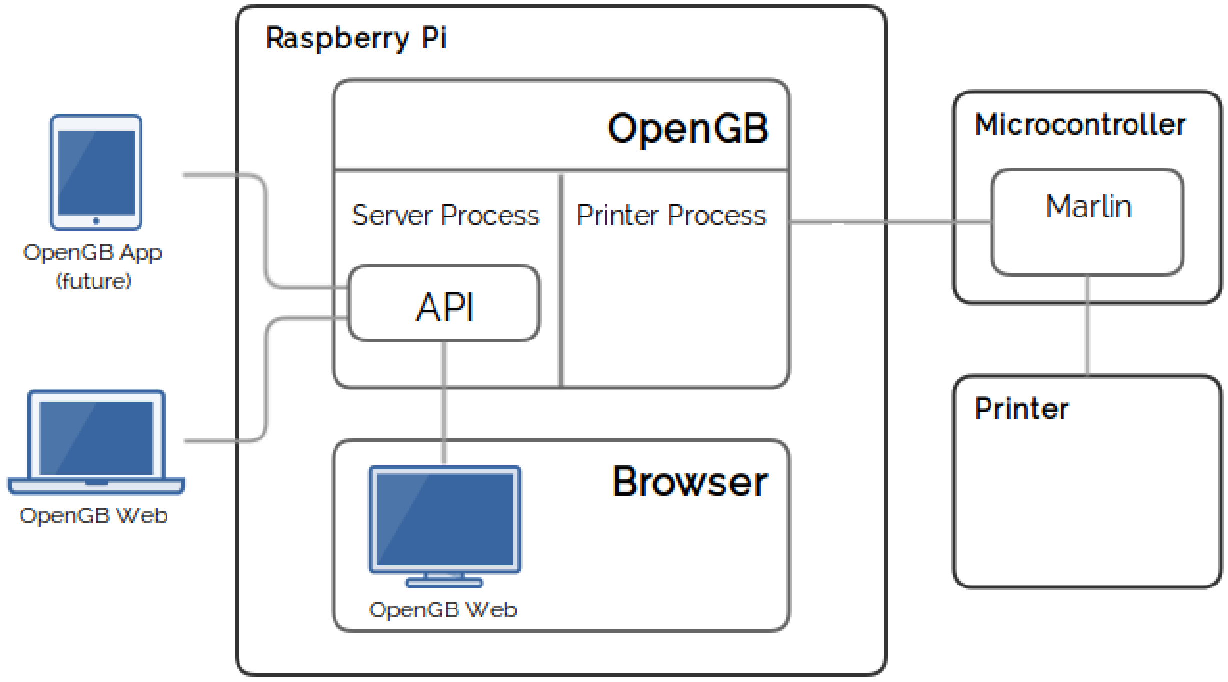
To disable the mouse pointer:

```
sudo apt-get install unclutter
echo "unclutter -idle 5" >> ~/.config/lxsession/LXDE-pi/autostart
```

CHAPTER 3

Architecture

OpenGB employs a multi-process architecture to ensure reliable, uninterrupted printing.



3.1 Server Process

The server process:

- Exposes printer functions to clients via JSON-RPC 2.0 *methods*.
- Broadcasts printer events to clients via JSON-RPC 2.0 *events*.
- Serves the OpenGB web interface.

It is based on the Tornado asynchronous web framework.

3.2 Printer Process

The printer process:

- Translates messages sent from the Server process and forwards them to the printer.
- Translates messages sent from the Printer into events and forwards them to the Server process.
- Performs routine requests (for printer metrics etc).

The printer process is extensible, allowing support for additional firmware interfaces to be added in the future.

CHAPTER 4

API

OpenGB functionality is exposed via a JSON-RPC 2.0 API over HTTP using WebSockets.

4.1 Connecting

The OpenGB web server exposes a WebSocket at *http://<hostname>:<port>/ws*.

To connect using Javascript:

```
var socket = new WebSocket("ws://" + window.location.host + "/ws");
socket.onmessage = function (message) {
  parseMessage(message)
};
```

4.2 Message Types

In accordance with the JSON-RPC 2.0 spec there are two distinct message types used by the API. The payloads of both are encoded as JSON objects.

4.2.1 Method

The client sends a JSON-RPC 2.0 request to the server defining a method call and the server replies with a JSON-RPC 2.0 response.

Example request (setting target temperatures):

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "set_temp",
    "params": {
        "bed": 105,
        "nozzle1": 206,
        "nozzle2": 203
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.2.2 Event

The server sends an unsolicited JSON-RPC 2.0 message to the client defining an event which has occured.

Example event (temperature update):

```
{
    "jsonrpc": "2.0",
    "event": "temp",
    "params": {
        "bed_current": 203,
        "bed_target": 105,
        "nozzle2_target": 203,
        "nozzle1_current": 104,
        "nozzle2_current": 108,
        "nozzle1_target": 206
    }
}
```

4.3 Methods

The `opengb.server.MessageHandler` class contains the methods exposed to JSON-RPC 2.0 clients.

4.3.1 set_temp

`MessageHandler.`**`set_temp`**(*bed=None*, *nozzle1=None*, *nozzle2=None*)

Set printer target temperatures.

Unspecified target temperatures will remain unchanged.

Parameters

- **`bed`** (`float`) – Bed target temperature.
- **`nozzle1`** (`float`) – Nozzle1 target temperature.
- **`nozzle2`** (`float`) – Nozzle2 target temperature.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "set_temp",
    "params":{
        "bed": 105,
        "nozzle1": 206,
        "nozzle2": 203
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.2 move_head_relative

`MessageHandler.`**`move_head_relative`**`(`*x=0*, *y=0*, *z=0*, *rate=300*`)`

Move the print head along one or more axes relative to the current position.

Parameters

- **x** (`float`) – Millimeters to move along the X axis.
- **y** (`float`) – Millimeters to move along the Y axis.
- **z** (`float`) – Millimeters to move along the Z axis.
- **rate** (`float`) – Rate at which to move in mm/s.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "move_head_relative",
    "params": {
        "x":    13.2,
        "y":    -2,
        "z":    0.03,
        "rate": 60
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.3 move_head_absolute

`MessageHandler.`**`move_head_absolute`**`(`*x=0*, *y=0*, *z=0*, *rate=300*`)`

Move the print head along one or more axes to an absolute position.

Parameters

- **x** (`float`) – Position to move to along the X axis.
- **y** (`float`) – Position to move to along the Y axis.
- **z** (`float`) – Position to move to along the Z axis.
- **rate** (`float`) – Rate at which to move in mm/s.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "move_head_absolute",
    "params": {
        "x":    105,
        "y":    80,
        "z":    20,
        "rate": 60
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.4 home_head

`MessageHandler.`**`home_head`**(*x=True*, *y=True*, *z=True*)

Home the print head along one or more axes.

Parameters

- **x** (`bool`) – Whether or not to home the X axis.
- **y** (`bool`) – Whether or not to home the Y axis.
- **z** (`bool`) – Whether or not to home the Z axis.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "home_head",
    "params":{
        "x": true,
        "y": true,
        "z": false
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.5 retract_filament

`MessageHandler.`**`retract_filament`**(*head*, *length*, *rate=300*)

Retract filament into the print head.

Parameters

- **`head`** (`int`) – Print head to retract (0 or 1).
- **`length`** (`float`) – Amount of filament to retract in mm.
- **`rate`** (`float`) – Rate at which to retract in mm/s.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "retract_filament",
    "params": {
        "head": 0,
        "length": 5,
        "rate": 300
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.6 unretract_filament

`MessageHandler.`**`unretract_filament`**(*head*, *length*, *rate=300*)

Unretract filament from the print head.

Parameters

- **`head`** (`int`) – Print head to unretract (0 or 1).
- **`length`** (`float`) – Amount of filament to unretract in mm.
- **`rate`** (`float`) – Rate at which to unretract in mm/s.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "unretract_filament",
    "params": {
        "head": 0,
        "length": 5,
        "rate": 300
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.7 set_extrude_override

MessageHandler.**set_extrude_override**(*percent*)

Set percentage override applied to extrusion commands.

Parameters **percent** (`float`) – Percentage by which extrusion should be overridden.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "set_extrude_override",
    "params": {
        "percent":120
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.8 set_speed_override

MessageHandler.**set_speed_override**(*percent*)

Set speed percentage override applied to movement commands.

Parameters **percent** (`float`) – Percentage by which movement speed should be overridden.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "set_speed_override",
    "params": {
        "percent":120
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.9 set_fan_speed

MessageHandler.**set_fan_speed**(*fan*, *percent*)

Set fan speed.

Parameters

- **fan** (class:*int* (0-2)) – Number of fan for which to set speed.
- **percent** (float) – Percentage of maximum speed to set.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "set_fan_speed",
    "params": {
        "fan": 1,
        "percent": 75
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.10 filament_swap_begin

MessageHandler.**filament_swap_begin**()

Begin filament swap procedure.

Normally this will pause an ongoing print, move the print head/bed and retract filament ready to be swapped.

Once filament swap is complete `filament_swap_complete()` should be called to return print head to previous position and continue printing.

Note: This is an experimental method and may be implemented differently for various printer + firmware combinations.

This method may be called internally as the result of a filament detection script being triggered.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "filament_swap_begin",
    "params": {}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.11 filament_swap_complete

`MessageHandler.`**`filament_swap_complete`**`()`

Complete filament swap procedure.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "filament_swap_complete",
    "params": {}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.12 enable_steppers

`MessageHandler.`**`enable_steppers`**`()`

Enable stepper motors.

Prevents motors and axes from moving freely.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "enable_steppers",
    "params": {}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.13 disable_steppers

`MessageHandler.`**`disable_steppers`**`()`

Disable stepper motors.

Allows motors and axes to moving freely.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "disable_steppers",
    "params": {}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.14 print_gcode_file

`MessageHandler.`**`print_gcode_file`**`(`*id*`)`

Print gcode file with given *id*.

Parameters **`id`** (`int`) – ID of the gcode file to get.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "print_gcode_file",
    "params": {
        "id": 1
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.15 pause_print

`MessageHandler.`**`pause_print`**`()`

Pause the current print job.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "pause_print",
    "params":{}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.16 resume_print

MessageHandler.**resume_print**()

Resume a paused print job.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "resume_print",
    "params":{}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.17 cancel_print

MessageHandler.**cancel_print**()

Cancel the current print job.

The current print job status and position will be discarded.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method":"cancel_print",
    "params": {}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.18 emergency_stop

`MessageHandler.`**`emergency_stop`**`()`

Halt the printer immediately.

Note: Use only in emergencies as this will most likely lock the printer and require a reboot.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "emergency_stop",
    "params": {}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.19 put_gcode_file

`MessageHandler.`**`put_gcode_file`**`(`*payload, name, print_material='‘, print_quality='‘, print_extruders='‘, print_time_sec=0, print_filament_mm=0, print_material_gm=0, thumbnail_png_base64='‘*`)`

Upload a gcode file.

Parameters

- **`payload`** (`str`) – Gcode file as ASCII text.
- **`name`** (`str`) – Gcode file name.
- **`print_material`** (`str`) – Intended print material (e.g. "PLA").
- **`print_quality`** (`str`) – Intended print quality (e.g. "High").
- **`print_extruders`** (`str`) – Extruders utilised (e.g. "Both").
- **`print_time_sec`** (`int`) – Estimated print time in seconds.
- **`print_filament_mm`** (`int`) – Estimated print filament length in millimeters.
- **`print_material_gm`** (`int`) – Estimated print material weight in grams.
- **`thumbnail_png_base64`** (`str` representing a base64-encoded png file) – Thumbnail image.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "put_gcode_file",
    "params": {
        "name": "test_cube.gco",
        "payload": "<gcode>"
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": {
        "id": 3,
        "name":"test_cube.gco"
    }
}
```

4.3.20 get_gcode_file

`MessageHandler.`**`get_gcode_file`**`(`*id*, *content=False*`)`

Get details of a single gcode file with the given *id*.

Optionally include the gcode file content.

Parameters

- **`id`** (`int`) – ID of the gcode file to get.
- **`content`** (`bool` (default False)) – Include the gcode file content in the results.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "get_gcode_file",
    "params": {
        "id": 1
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "size": 2914599,
        "id": 1,
        "name": "FE_Drakkar_Bow.gcode",
        "uploaded": "2016-04-11 19:54:31.929633"
    }
}
```

4.3.21 get_gcode_files

`MessageHandler.`**`get_gcode_files`**`()`

Get details of all gcode files.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "get_gcode_files",
    "params": {}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": {
        "gcode_files": [
            {
                "size": 2914599,
                "id": 1,
                "name": "FE_Drakkar_Bow.gcode",
                "uploaded": "2016-04-11 19:54:31.929633"
            },
            {
                "size": 24356,
                "id": 2,
                "name": "10mm_Test_Cube.gcode",
                "uploaded": "2016-04-12 13:24:15.345623"
            }
        ]
    }
}
```

4.3.22 delete_gcode_file

`MessageHandler.`**`delete_gcode_file`**`(`*id*`)`

Delete a single gcode file with the given *id*.

Parameters **`id`** (`int`) – ID of the gcode file to delete.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "delete_gcode_file",
    "params": {
        "id": 3
    }
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": true
}
```

4.3.23 get_counters

`MessageHandler.`**`get_counters`**`()`

Get printer counter values.

Counters are listed in `opengb.database.COUNTERS`.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 2,
    "method": "get_counters",
    "params":{}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 2,
    "result":{
        "counters": {
            "nozzle_2_up_mins":128,
            "motor_x1_up_mins":128,
            "motor_x2_up_mins":128,
            "motor_y2_up_mins":128,
            "nozzle_1_up_mins":128,
            "motor_z2_up_mins":128,
            "motor_z1_up_mins":128,
            "printer_up_mins":128,
            "printer_print_mins":46,
            "bed_up_mins":128,
            "motor_y1_up_mins":128,
            "printer_up_mins_session":32
        }
    }
}
```

4.3.24 get_filesystem_utilization

`MessageHandler.`**`get_filesystem_utilization`**`()`

Get current filesystem utilization.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 3,
    "method": "get_filesystem_utilization",
    "params": {}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 3,
    "result": {
        '/': {
            'free_bytes': 183485939712,
            'total_bytes': 243515678720,
            'utilized_bytes': 47636201472,
            'utilized_percent': 19.6
        },
        '/boot': {
            'free_bytes': 110014464,
            'total_bytes': 246755328,
            'utilized_bytes': 124001280,
            'utilized_percent': 50.3
        }
    }
}
```

4.3.25 get_status

`MessageHandler.`**`get_status`**`()`

Get current printer status.

Note: This method should only be used to determine baseline printer status. To maintain a view of current printer status listen for the various *jsonrpc* events.

Example request:

```
{
    "jsonrpc": "2.0",
    "id": 4,
    "method": "get_status",
    "params": {}
}
```

Example response:

```
{
    "jsonrpc": "2.0",
    "id": 4,
    "result": {
        "status": {
            "progress": {
                "current": 0,
                "total": 0
            },
            "state": 20,
            "position": {
                "z": 0,
                "y": 0,
                "x": 0
            },
            "temp": {
                "bed_current": 100,
                "nozzle2_target": 0,
                "bed_target": 0,
                "nozzle1_target": 0,
                "nozzle2_current": 209,
                "nozzle1_current": 205
            },
            "steppers": {
                "enabled": true
            },
            "extrude_override": {
                "percent": 100
            },
            "speed_override": {
                "percent": 120
            },
            "fan_speed": {
                0: 100,
                1: 75,
                2: 0
            }
        }
    }
}
```

4.4 Events

4.4.1 state_change

Sent when the printer status changes. Valid states are:

1. DISCONNECTED
2. READY
3. EXECUTING
4. PAUSED
5. FILAMENT_SWAP
6. ERROR

Example event:

```
{
    "jsonrpc": "2.0",
    "event": "state_change",
    "params": {
        "old": "READY",
        "new": "EXECUTING"
    }
}
```

4.4.2 extrude_override_change

Sent when the extrude override percentage changes.

Example event:

```
{
    "jsonrpc": "2.0",
    "event": "extrude_override_change",
    "params": {
        "percent": 120
    }
}
```

4.4.3 speed_override_change

Sent when the movement speed override percentage changes.

Example event:

```
{
    "jsonrpc": "2.0",
    "event": "speed_override_change",
    "params": {
        "percent": 120
    }
}
```

4.4.4 fan_speed_change

Sent when the speed of a fan changes.

Example event:

```
{
    "jsonrpc": "2.0",
    "event": "fan_speed_change",
    "params": {
        "fan": 1,
        "percent": 75
    }
}
```

4.4.5 temp_update

Sent periodically to provide current and target temperatures of printer components.

Note: Not all parameters of every *temp_update* will necessarily contain values. If a parameter's value is *null* then the update does not contain any new data for that parameter and its value should be considered *unchanged*.

Example event:

```
{
    "jsonrpc": "2.0",
    "event": "temp_update",
    "params": {
        "bed_current": 205,
        "bed_target": 0,
        "nozzle1_current": 106,
        "nozzle1_target": 0,
        "nozzle2_current": 101,
        "nozzle2_target": 0
    }
}
```

4.4.6 position_update

Sent on print head movement to provide current print head position.

Example event:

```
{
    "jsonrpc": "2.0",
    "event": "position_update",
    "params": {
        "x": 100,
        "y": 80,
        "z": 20
    }
}
```

4.4.7 progress_update

Sent periodically while printer is executing a gcode sequence.

Example event:

```
{
    "jsonrpc": "2.0",
    "event": "progress_update",
    "params": {
        "current_line": 327,
        "total_lines": 4393
    }
}
```

4.4.8 steppers_update

Sent when stepper motors are enabled/disabled.

Example event:

```
{
    "jsonrpc": "2.0",
    "event": "steppers_update",
    "params": {
        "enabled": true
    }
}
```

CHAPTER 5

Contributing

OpenGB is an Open Source project and all contributions are gratefully appreciated.

By donating your time and skills you are helping to build a platform used by 3D printing enthusiasts around the globe!

5.1 Repositories

5.1.1 OpenGB

OpenGB is written in Python and hosted in the Github repository re-3D/opengb.

5.1.2 OpenGB Web

The OpenGB web frontend is written in Vue.js, a Javascript framework. It is hosted separately in the Github repository re-3D/opengb-web.

Changes to the frontend trigger a build which outputs to re-3D/opengb-web/dist. This directory is periodically pulled into the OpenGB repo using `git read-tree` as described in the OpenGB README.

5.2 Bug Reports

Report bugs as OpenGB Github issues using the label `bug`.

Be sure to check existing issues for duplicates.

5.3 Feature Requests

Request features as OpenGB Github issues using the label `enhancement`.

Be sure to check existing issues for duplicates.

5.4 Submitting Code

1. Create an OpenGB Github issue for the bug/feature in question (if one does not already exist).

2. Fork the OpenGB repository.
3. Create a topic branch off `develop`.
4. Make changes, being aware of:
   - Logical commits
   - Whitespace
   - PEP8
5. Update the CHANGELOG following the instructions therein.
6. Push changes to your topic branch.
7. Submit a PR.
8. Wait for feedback from a project maintainer.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Index