Octoprint

https://community.octoprint.org/t/setting-up-octoprint-on-a-raspberry-pi-running-raspbian-or-raspberry-pi-os/2337

```
cd ~
sudo apt update
sudo apt install python3-pip python3-dev python3-setuptools python3-venv git
libyaml-dev build-essential
mkdir OctoPrint && cd OctoPrint
python3 -m venv venv
source venv/bin/activate
```

In the virtual environment do:

```
pip install pip --upgrade
pip install octoprint
```

Add user to tty and dialout group to allow access to serial port:

```
sudo usermod -a -G tty $USER
sudo usermod -a -G dialout $USER

#reboot required for the new group to take effect. For temporary access:
sudo chown $USER /dev/ttyUSB0
sudo chmod a+rw /dev/ttyUSB0
```

Allow octoprint to restart itself, replace USER with username:

/etc/sudoers.d/octoprint

```
USER ALL=NOPASSWD: /usr/bin/systemctl start
octoprint,/usr/bin/systemctl stop octoprint,/usr/bin/systemctl restart
octoprint
USER ALL=NOPASSWD: /usr/sbin/service octoprint
restart,/usr/sbin/service octoprint stop,/usr/sbin/service octoprint
stop
```

Start service:

```
~/OctoPrint/venv/bin/octoprint serve
```

Access it via http://localhost:5000

Autostart service:

```
wget
```

```
https://github.com/OctoPrint/OctoPrint/raw/master/scripts/octoprint.service
sudo mv octoprint.service /etc/systemd/system/octoprint.service
sudo sed -i 's/pi/'"$USER"'/g' /etc/systemd/system/octoprint.service
sudo systemctl enable octoprint.service
```

Update:

```
cd ~/OctoPrint
python3 -m venv venv
source venv/bin/activate
pip install pip --upgrade
pip install setuptools --upgrade
pip install octoprint --upgrade
```

Nexus AI

Free local plugin to detect print failures from webcam images
https://plugins.octoprint.org/plugins/nexus_ai/

OctoLapse

Better TimeLapse, independent of the built in timelapse.

Auto config:
https://github.com/FormerLurker/Octolapse/wiki/V0.4---Automatic-Slicer-Configuration#if-you-are-using-cura-follow-these-steps

Go to Settings → Preferences and click Machine Settings of printer. Paste at top of Start G-code:

```
; Script based on an original created by tjjfvi (https://github.com/tjjfvi)
; An up-to-date version of the tjjfvi's original script can be found
; here:  https://csi.t6.fyi/
; Note - This script will only work in Cura V4.2 and above!
; --- Global Settings
; layer_height = {layer_height}
; smooth_spiralized_contours = {smooth_spiralized_contours}
; magic_mesh_surface_mode = {magic_mesh_surface_mode}
; machine_extruder_count = {machine_extruder_count}
; --- Single Extruder Settings
; speed_z_hop = {speed_z_hop}
; retraction_amount = {retraction_amount}
; retraction_hop = {retraction_hop}
; retraction_hop_enabled = {retraction_hop_enabled}
; retraction_enable = {retraction_enable}
; retraction_speed = {retraction_speed}
; retraction_retract_speed = {retraction_retract_speed}
```

```
; retraction_prime_speed = {retraction_prime_speed}
; speed_travel = {speed_travel}
```

Taking photo before or at first layer: https://github.com/FormerLurker/Octolapse/issues/677

Add to the bottom of the very end of the Start G-code:

```
@OCTOLAPSE TAKE-SNAPSHOT
SNAP
```

or

```
G4 P1
```

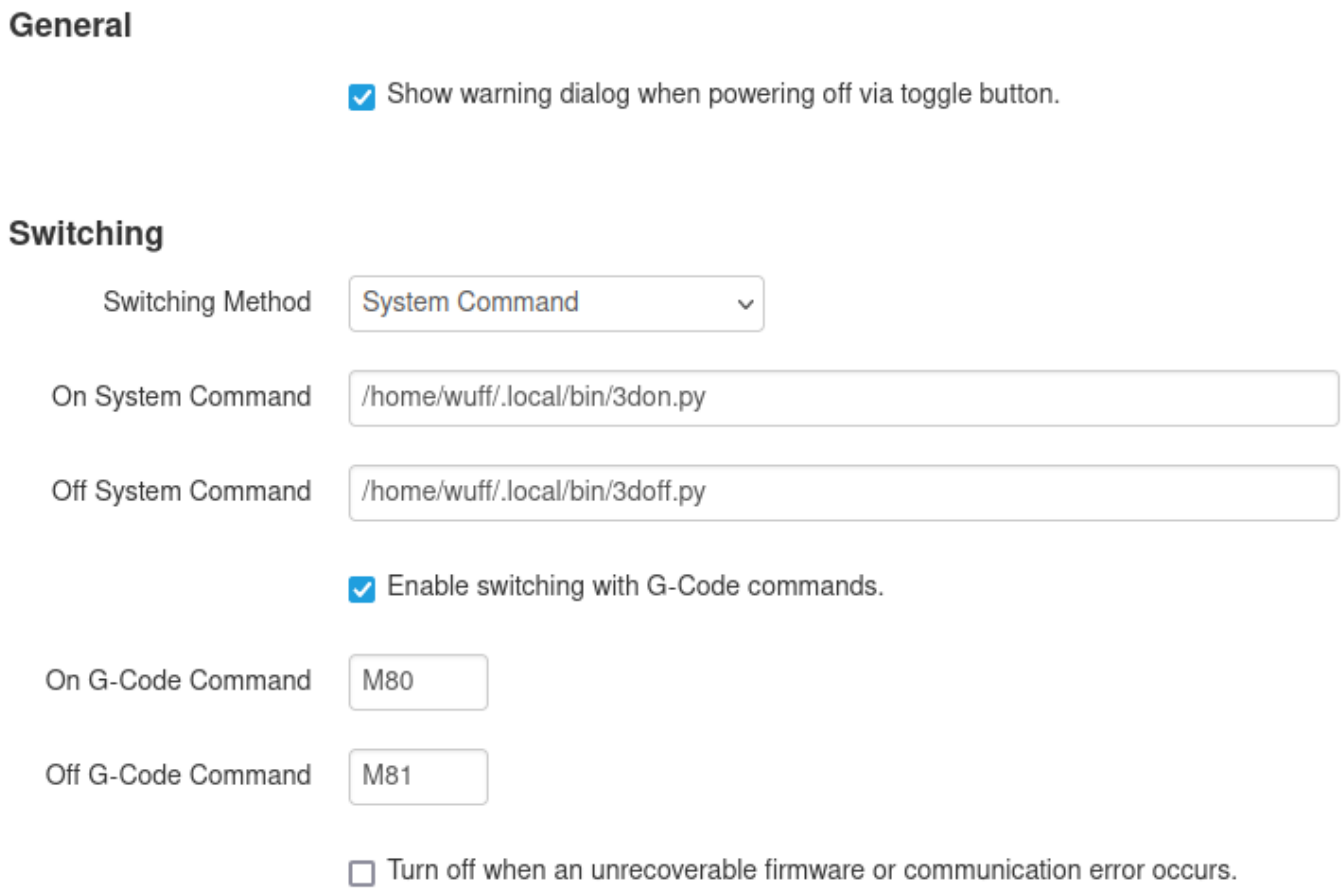
Add to very end of End G-Code:

```
G28 Z0 ;move Z to min endstops
```

PSUControl + Tuya Plug/LED On/Off

The following python scripts can be used to switch on/off a Tuya compatible smart plug in the local network and to trigger two flashes in red of a smart LED, then restoring the LED to the previous colour and state.

Using the PSUControl plugin, these scripts can be triggered automatically:
https://github.com/kantlivelong/OctoPrint-PSUControl



To locally control the Tuya devices, the tinytuya python module needs to be installed:

```
pip install tinytuya
```

The module provides local system scanning options and methods to obtain the local key required to control the devices. The devices may need to be connected to a Tuya cloud account first and an API account set up. However, this might only be required if the devices need to be controlled from tinytuya using the cloud option from outside the local network. More information provided here: https://pypi.org/project/tinytuya/

~/.local/bin/3don.py

```
#!/usr/bin/python
import tinytuya

# 3D Printer Plug
d = tinytuya.OutletDevice(
    dev_id='07870772cc50e3d2fcf2',
    address='192.168.1.23',
    local_key='d7382aa465d40908',
    version=3.3)

d.turn_on()
```

~/.local/bin/3doff.py

```python
#!/usr/bin/python
import tinytuya
import time

# Hall Light

# Connect to Device
d = tinytuya.BulbDevice(
    dev_id='722168502cf4320a9d1e',
    address='192.168.1.11',
    local_key='3209036606016f40',
    version=3.1)

# Optional: Keep socket open for multiple commands
d.set_socketPersistent(True)
d.set_socketNODELAY(True)
d.set_sendWait(0)

# Get Status as dictionary
olddata = d.status()
olddps = olddata['dps']

#Switch on
d.turn_on(nowait=True)

d.set_scene(3, nowait=True)

time.sleep(6)

# restoring old data
for key, value in olddps.items():
    # print('%s : %s' % (key, value))
    d.set_value(key, value)

# 'dps': {'1': True, '2': 'colour', '3': 135, '4': 255, '5':
'301f000027ff2f', '6': 'cf38000168ffff', '7': 'ffff500100ff00', '8':
'ffff8003ff000000ff000000ff00000000000000000', '9': 'ffff5001ff0000',
'10': 'ffff0505ff000000ff00ffff00ff00ff0000ff000000'}}
# Wulf Default:
d.set_value(1, olddps['1'])
d.set_value(2, "colour")
d.set_value(3, 135)
d.set_value(4, 255)
d.set_value(5, '301f000027ff2f')
d.set_value(6, 'cf38000168ffff')
d.set_value(7, 'ffff500100ff00')
d.set_value(8, 'ffff8003ff000000ff000000ff00000000000000000')
d.set_value(9, 'ffff5001ff0000')
d.set_value(10, 'ffff0505ff000000ff00ffff00ff00ff0000ff000000')

time.sleep(10)
```

```
# 3D Printer Plug
p = tinytuya.OutletDevice(
    dev_id='07870772cc50e3d2fcf2',
    address='192.168.1.23',
    local_key='d7382aa465d40908',
    version=3.3)

p.turn_off()
```

```
chmod 755 ~/.local/bin/3don.py
chmod 755 ~/.local/bin/3doff.py
```

From:
http://wuff.dyndns.org/ - **Wulf's Various Things**

Permanent link:
http://wuff.dyndns.org/doku.php?id=3dprinter:octoprint&rev=1682942807

Last update: **2023/05/29 11:53**