Table of Contents

BashInLove with OctoPrint

Configuracions i utilitats necessaries

Instalació d'Octoprint a la Raspberry

Dependències necessàries:

Octoprint està programat en python, per tant necessitem un entorn amb llibreries bàsiques i amb possibilitat de compilació amb GCC.

```
# aptitude install python-pip python-dev python-setuptools python-virtualenv git
libyaml-dev build-essential
```

Instalació d'octoprint

```
$ mkdir OctoPrint && cd OctoPrint
$ virtualenv venv
$ source venv/bin/activate
$ pip install https://get.octoprint.org/latest
$ sudo usermod -a -G tty pi
$ sudo usermod -a -G dialout pi
$ ~/OctoPrint/venv/bin/octoprint serve → Això inicia el server OctoPrint al port 5000
```

Ara cal connectar via web al port 5000 de la raspberry i executar una configuració mínima.

Un cop connectats també cal anar a la secció Settings → API i habilitar CORS (Cross Origin Resource Sharing) i obtenir l'**ApiKey**

Instalació de Headless Cura3D

Deixo el procés d'instalació tot i que no l'he pogut fer servir, ja que no he trobat els perfils adeqüats per aquesta versió i la BQ Hephestos 2.

```
# cd /opt/
# git clone -b legacy https://github.com/Ultimaker/CuraEngine.git
# cd CuraEngine/
# wget http://bit.ly/curaengine_makefile_patch -O CuraEngine.patch
# patch < CuraEngine.patch
# make
# ln /opt/CuraEngine /bin/CuraEngine
$ ./CuraEngine → verifiquem que funciona
```

Habilitar servei SystemD per executar el servidor a l'inici del sistema.

vim /usr/lib/systemd/system/octoprint.service

```
[Unit]
Description=Octoprint server a port 5000
After=network.target
[Service]
Type=simple
User=pi
ExecStart=/home/pi/OctoPrint/venv/bin/octoprint serve
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=octoprint
KillMode=process
[Install]
WantedBy=multi-user.target
Alias=octoprint.service
```

systemctl enable octoprint.service

Instalació de dependècies de sistema.

Curl

Curl és un client del protocol HTTP per consola, suporta tots els mètodes (GET, POST, PUT, DELETE) que necessitarem per treballar amb l'API d'octoprint.

```
# aptitude install curl
```

FzF

És un selector Ncurses. Permet triar entre una llista de files i retornar-ne només una seleccionada.

```
$ cd
$ git clone https://github.com/junegunn/fzf.git .fzf
$ .fzf/install
```

Amb la instal·lació podrem tirar si volem agefir un control optatiu a bash que millora funcionalitats de History.

JQ

És una utilitat de parseig de documents JSON. Amb força funcionalitats de filtratge o fins i tot per la creació de coleccions a partir de dades RAW o CSV. És util per l'acolorit i per realitzar consultes ràpides sobre les respostes de servidor.

/etc/hosts

Afegirem una línia alias a hosts per identificar la raspberry. Per tal que el codi sigui agnòstic de la IP.

```
10.199.160.152 rpi
```

Ententent l'API

Se'n pot obtenir informació completissima a:

http://docs.octoprint.org/en/master/api/

Mètodes GET

És el mètode fet servir per obtenir informació. És el mètode que es fa servir per veure planes web.

En el cas de les API RESTful el retorn no és un HTML sinó un JSON.

Mètodes POST

És el mètode utilitzat per enviar informació al servidor. El símil pot ser quan omplim i enviem un formulari d'una web.

En aquest cas en API seríem nosaltres qui enviem un JSON, bé sol o acompanyat d'un «attachment»

Mètodes DELETE i PUT

Serveixen per realitzar operacions sobre objectes HTTP, crear o esborrar arxius, etc ...

Endpoints

Equivalen als noms de les funcions en programació. Generalment controlats per un objecte. Canvia la seva programació en funció del mètode utilitzat.

Descripció del projecte

Objectius

1. Documentar els Endpoints més útils pel mètode GET. Obtenir informació i status operacional de la impressora des de l'API.
2. Aconseguir pujar un arxiu GCODE i imprimir-lo.
3. Monitorar un estat d'impressió, % completat, temperatura de l'extrussor...
4. Evitar intentar imprimir si l'estat de la impressora no és «preparat per imprimir».

Col·lecció de GETs

La primera fase consisteix a identificar els endpoints més útils que em permetin obtenir la informació operacional de la impressora.

A tal efecte vaig crear una col·lecció simple de noms d'endpoints i veure quin era l'output que produien.

La clau de la primera fase és passar un «here document» a **fzf** que modificara l'script de connexió.

```
#!/usr/bin/env bash

CURL='curl -s -H Content-Type:application/json -H X-Api-
Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 '
BURL='rpi:5000/api/'

get_utils(){

fzf << EOF
connection
files
files?recursive=true
languages
job
slicing
printer
timelapse
version
EOF

}
${CURL}${BURL}$(get_utils) |jq .
```

Pujar un Gcode

L'endpoint triat és un que funciona amb mètode GET. Permet el pas de variables a URL i requereix ben poques opcions per poder inicialitzar una impressió.

```
curl -i -H X-Api-Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 -F select=true -F print=true -F
file=@$1 rpi:5000/api/files/local
```

Les variables inicialitzades son:

- select=true → Posa l'arxiu a la cua d'impressió.
- print=true → Ordena que s'inicii directament la impressió
- file=@$1 → @especifica un arxiu local, passat a l'script per paràmetres «$1»

L'endpoint és «files» i «local» és un paràmetre. Es podria especificar també «sdcard».

Monitorant l'estat d'impressió

Els endpoints triats a aquest efecte són:

- Job → Per obtenir la informació de «temps estimat» i «progrés»
- Printer → Estatus (Offline, Printing,) i temperatura de l'extrusor.

```
$ curl -s -XGET -H content-Type:application/json -H X-Api-
Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 rpi:5000/api/printer
```


```
{
  "sd": {
    "ready": true
  },
  "state": {
    "flags": {
      "cancelling": false,
      "closedOrError": false,
      "error": false,
      "finishing": false,
      "operational": true,
      "paused": false,
      "pausing": false,
      "printing": true,
      "ready": false,
      "resuming": false,
      "sdReady": true
    },
    "text": "Printing"
  },
  "temperature": {
    "tool0": {
      "actual": 211,
      "offset": 0,
      "target": 210
    }
  }
}
```


```
$ curl -s -XGET -H content-Type:application/json -H X-Api-
Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 rpi:5000/api/job
```


```
{
  "job": {
    "averagePrintTime": null,
    "estimatedPrintTime": null,
    "filament": null,
    "file": {
      "date": 1539771267,
      "display": "BQH2_3_microsd.gcode",
      "name": "BQH2_3_microsd.gcode",
      "origin": "local",
      "path": "BQH2_3_microsd.gcode",
      "size": 247919
    },
    "lastPrintTime": null,
    "user": "_api"
  },
  "progress": {
    "completion": 16.65140630609191,
    "filepos": 41282,
    "printTime": 277,
    "printTimeLeft": null,
    "printTimeLeftOrigin": "linear"
  },
  "state": "Printing"
}
```


En la versió reduïda obtenint només la informació més crítica he deixat:

```
ESTAT: Printing
Arxiu: null
Progres: %Complert: 47.549804573267885 %Temps estimat restant: 523
Temperatura: 211 de 210
```

Connectar i desconnectar

Aquests endpoints van per mètode POST i per tant s'envia un JSON al servidor.

$ connect.sh

```
#!/usr/bin/env bash
echo $0
curl -i -s -XPOST -H content-Type:application/json -H X-Api-
Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 -d @connect.json  rpi:5000/api/connection
```

$ disconnect.sh

```
#!/usr/bin/env bash
echo $0
curl -i -s -XPOST -H content-Type:application/json -H X-Api-
Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 -d @disconnect.json  rpi:5000/api/connection
```

$ connect.json

```
{
       "command": "connect"
}
```

$ disconnect.json

```
{
       "command": "disconnect"
}
```

Cancel·lar el job actual

$ canceljob.sh

```
#!/usr/bin/env bash
echo $0
curl -i -s -XPOST -H content-Type:application/json -H X-Api-
Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 -d @cancel.json  rpi:5000/api/job
```

$ cancel.json

```
{
  "command": "cancel"
}
```

Cancel es pot subtutir per «pause» o «resume» si haguéssim de fer una operació al mig de la impressió i despres continuar-la

Tot unit

La comanda print.sh espera un argument que seria el nom de l'arxiu gcode que tenim en local a l'ordinador i que serà enviat.

Verifica si la impressora està preparada per imprimir i en cas afirmatiu llença el gcode i inicia el treball d'impressió.

Mostra l'estat de la impressió per pantalla.

Espera el trap ^C per si cal cancel·lar la impressió. En aquest cas executa «canceljob.sh»

```
#!/usr/bin/env bash

# trap ctrl-c and call ctrl_c()
trap ctrl_c INT

function ctrl_c() {
        echo "... Job stopped!"
               #echo $(./canceljob.sh)
               exit 0
}

# Argument Check
if [ $# -ge 1 ]; then
       if [ -f $1 ]; then
               echo ...Good args , ready to print: $1
       else
```

```
                # Even if a file is issued. Check if exists
                echo ...File $1 does not exist. ABORT
                exit 1
        fi

        # Temporary sources. Refreshed every 2 seconds.
        tmpstatus=/tmp/.prstatus.out
        tmpjob=/tmp/.jobstatus.out
        # Checks if printer is ready to print.
        startstatus=`curl -s -H X-Api-Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 -H content-
Type:application/json rpi:5000/api/printer | jq -r .state.text`
        # Checks if printer is connected.
        [[ $? -ne 0 ]] && exit 4
        # If all good. Start print
        if [ ${startstatus-fail} == "Operational" ]; then
                curl -i -H X-Api-Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 -F select=true -F
print=true -F file=@$1 rpi:5000/api/files/local
        else
                echo "... Check printer status"
        fi
        # file suddently missing?
        if [ $? -ne 0 ]; then echo "wrong filename"; exit 1; fi
        printstatus=`curl -s -H X-Api-Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 -H content-
Type:application/json rpi:5000/api/printer | jq -r .state.text`
        # Status monitoring loop
        while [ $printstatus == "Printing" ]; do
                clear
                curl -s -XGET -H content-Type:application/json -H X-Api-
Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 rpi:5000/api/job > $tmpjob
                curl -s -XGET -H content-Type:application/json -H X-Api-
Key:15BE7B04DE2B4DB68D7A7D750B6F8E42 rpi:5000/api/printer > $tmpstatus
                printstatus=`jq -r .state.text $tmpstatus`
                echo ESTAT: $printstatus
                echo Arxiu: `jq -r .file.name $tmpjob`
                echo Progres: `jq -r '.progress | "%Complert: \(.completion)\t%Temps
estimat restant: \(.printTimeLeft)"' $tmpjob`
                echo `jq -r '.temperature.tool0 | "Temperatura: \(.actual) de \
(.target)"' $tmpstatus`
                sleep 2
        done
else
        echo ... Must provide Gcode filename as Arg
        exit 1
fi
# Cleanup
jq -r . /tmp/.jobstatus.out
```