Manual para varias instancias de Octo-Print

David Zotes González
ABADÍA TECNOLÓGICA

Contenido

1. Maquina

Lo primero que tenemos que tener es instalada una maquina con un Linux. En nuestro caso hemos instalado un Debian.

También existe una imagen reparada para Raspberry Pi pero en este manual lo vamos a hacer sobre un pc con Debian.

No nos vamos a detener en este aspecto ya que asumimos que este paso no debería dar problemas.

2. Usuarios

A continuación, procederemos a añadir un usuario por cada instancia de OctoPrint que queremos instalar. Añadirems un usuario por cada instancia de OctoPrint que queramos instalar. Esto lo haremos con el siguiente comando:

```
1.  adduser impresor1
2.  adduser impresor2
3.  adduser impresor3
4.  adduser impresor4
5.  adduser impresor5
6.  adduser impresor6
7.  adduser impresor7
```

Importante:

Como en nuestro caso nos conectaremos a nuestra maquina por SSH NO deberemos utilizar el comando *"useradd"* ya que si usamos este comando en la consola no saldrá el prompt y dificultará enormemente nuestra labor.

3. Grupos de trabajo

Una vez hemos creado los usuarios (en nuestro caso 7) deberemos añadir todos nuestros usuarios a los grupos de trabajo "dialout" y "tty" para ello usaremos el comando "usermod" de la siguiente manera:

```
1.  usermod -a -G dialout impresor1
2.  usermod -a -G tty impresor1
3.
4.  usermod -a -G dialout impresor2
5.  usermod -a -G tty impresor2
6.          .
7.          .
8.          .
```

4. Usuario cámara

A continuación, hemos creado un usuario nuevo para controlar todas las webcams:

```
useradd -G video -M -r webcam
```

5. Clonar repositorio

Debemos iniciar sesión con todos los usuarios que hemos creado anteriormente y luego procederemos a clonar todas las instancias de OctoPrint dentro de cada usuario (uno por uno).

Ahora procederemos a clonar nuestro repositorio de OctoPrint en cada usuario que hemos creado. En nuestro caso lo hemos clonado desde aquí: https://github.com/foosel/OctoPrint
Pero también se puede clonar desde el repositorio oficial de OctoPrint. Además de clonar el repositorio deberemos arrancar el virtualenv y ejecutar el setup de python. Esto lo haremos de la siguiente manera:

```
1.  git clone https://github.com/foosel/OctoPrint
2.  cd OctoPrint
3.  virtualenv venv
4.  ./venv/bin/pip install pip --upgrade
5.  ./venv/bin/python setup.py install
6.  mkdir ~/.octoprint
```

Para lanzar el servidor de manera manual utilizaremos el siguiente comando:

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

Tan solo lo haremos para probar ya que más adelante veremos cómo configurar los scripts para que se arranquen de manera automática desde una misma consola.

6. Scripts OctoPrint

Octoprint cuenta con unos scripts para lanzar los OctoPrint con un solo comando y nos permite poderlos lanzar todos desde la misma consola.
Los scripts de encuentran en los directorios */etc/init.d* y *etc/default* y deberemos cambiarlos para adecuarlos a nuestras necesidades.

A continuación, veremos el script asociado al octoprint1, pero en nuestro caso tenemos 7 máquinas así que cambiaremos la variable “INSTANCENUMBER” para configurar los demás scripts.
Script */etc/init.d/octoprint1*:

```
1.  #!/bin/sh
2.
3.  ### BEGIN INIT INFO
4.  # Provides:          octoprint
5.  # Required-Start:    $local_fs networking
6.  # Required-Stop:
```

```
7. # Should-Start:
8. # Should-Stop:
9. # Default-Start:     2 3 4 5
10.      # Default-Stop:      0 1 6
11.      # Short-Description: OctoPrint daemon
12.      # Description:       Starts the OctoPrint daemon with the user spe-
   cified in
13.      #                    /etc/default/octoprint.
14.      ### END INIT INFO
15.
16.      # Author: Sami Olmari & Gina Häußge
17.      INSTANCENUMBER=1
18.
19.      PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
20.      DESC="OctoPrint $INSTANCENUMBER Daemon"
21.      NAME="OctoPrint $INSTANCENUMBER"
22.      PKGNAME=octoprint$INSTANCENUMBER
23.      PIDFILE=/var/run/$PKGNAME.pid
24.      SCRIPTNAME=/etc/init.d/$PKGNAME
25.      DEFAULTS=/etc/default/$PKGNAME
26.
27.      # Read configuration variable file if it is present
28.      [ -r $DEFAULTS ] && . $DEFAULTS
29.
30.      # Define LSB log_* functions.
31.      # Depend on lsb-base (>= 3.0-6) to ensure that this file is present.
32.      . /lib/lsb/init-functions
33.
34.      # Exit if the DAEMON is not set
35.      if [ -z "$DAEMON" ]
36.      then
37.          log_warning_msg "Not starting $PKGNAME, DAEMON not set in
   /etc/default/$PKGNAME."
38.          exit 0
39.      fi
40.
41.      # Exit if the DAEMON is not installed
42.      [ -x "$DAEMON" ] || exit 0
43.
44.      # Load the VERBOSE setting and other rcS variables
45.      [ -f /etc/default/rcS ] && . /etc/default/rcS
46.
47.      if [ -z "$START" -o "$START" != "yes" ]
48.      then
49.         log_warning_msg "Not starting $PKGNAME, edit /etc/de-
   fault/$PKGNAME to start it."
50.         exit 0
51.      fi
52.
53.      if [ -z "$OCTOPRINT_USER" ]
54.      then
55.          log_warning_msg "Not starting $PKGNAME, OCTOPRINT_USER not set
   in /etc/default/$PKGNAME."
56.          exit 0
57.      fi
58.
59.      COMMAND_ARGS=
60.      if [ -n "$BASEDIR" ]
61.      then
62.          COMMAND_ARGS="--basedir $BASEDIR $COMMAND_ARGS"
63.      fi
64.
65.      if [ -n "$CONFIGFILE" ]
66.      then
```

```
67.           COMMAND_ARGS="--config $CONFIGFILE $COMMAND_ARGS"
68.       fi
69.
70.       #
71.       # Function to verify if a pid is alive
72.       #
73.       is_alive()
74.       {
75.          pid=`cat $1` > /dev/null 2>&1
76.          kill -0 $pid > /dev/null 2>&1
77.          return $?
78.       }
79.
80.       #
81.       # Function that starts the daemon/service
82.       #
83.       do_start()
84.       {
85.          # Return
86.          #   0 if daemon has been started
87.          #   1 if daemon was already running
88.          #   2 if daemon could not be started
89.
90.          is_alive $PIDFILE
91.          RETVAL="$?"
92.
93.          if [ $RETVAL != 0 ]; then
94.              start-stop-daemon --start --background --quiet --pid-
   file $PIDFILE --make-pidfile \
95.              --exec $DAEMON --chuid $OCTOPRINT_USER --user $OCTO-
   PRINT_USER --umask $UMASK --nicelevel=$NICELEVEL \
96.              -- serve $COMMAND_ARGS $DAEMON_ARGS
97.              RETVAL="$?"
98.          fi
99.       }
100.
101.      #
102.      # Function that stops the daemon/service
103.      #
104.      do_stop()
105.      {
106.         # Return
107.         #   0 if daemon has been stopped
108.         #   1 if daemon was already stopped
109.         #   2 if daemon could not be stopped
110.         #   other if a failure occurred
111.
112.         start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --
   user $OCTOPRINT_USER --pidfile $PIDFILE
113.         RETVAL="$?"
114.         [ "$RETVAL" = "2" ] && return 2
115.
116.         rm -f $PIDFILE
117.
118.         [ "$RETVAL" = "0"  ] && return 0 || return 1
119.      }
120.
121.      case "$1" in
122.        start)
123.         [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
124.         do_start
125.         case "$?" in
126.            0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
127.            2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
```

```
        esac
        ;;
      stop)
        [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
        do_stop
        case "$?" in
           0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
           2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
        esac
        ;;
      status)
        status_of_proc -p $PIDFILE $DAEMON $NAME && exit 0 || exit $?
        ;;
      restart)
        log_daemon_msg "Restarting $DESC" "$NAME"
        do_stop
        case "$?" in
          0|1)
           do_start
           case "$?" in
              0) log_end_msg 0 ;;
              1) log_end_msg 1 ;; # Old process is still running
              *) log_end_msg 1 ;; # Failed to start
           esac
           ;;
          *)
             # Failed to stop
           log_end_msg 1
           ;;
        esac
        ;;
      *)
        echo "Usage: $SCRIPTNAME {start|stop|status|restart}" >&2
        exit 3
        ;;
    esac

```

Ahora haremos lo propio para los que se encuentran en la ruta /etc/default/

Script /etc/default/octoprint1:

```
# Configuration for /etc/init.d/octoprint

INSTANCENUMBER=1

# The init.d script will only run if this variable non-empty.
OCTOPRINT_USER=impresor$INSTANCENUMBER

# base directory to use
BASEDIR=/home/impresor${INSTANCENUMBER}/.octoprint

      # configuration file to use
      CONFIGFILE=/home/impresor${INSTANCENUMBER}/.octoprint/config.yaml

      # On what port to run daemon, default is 5000
      PORT=500$INSTANCENUMBER

      # Path to the OctoPrint executable, you need to set this to match
   your installation!
```

```
18.       DAEMON=/home/impresor${INSTANCENUMBER}/OctoPrint/venv/bin/octoprint
19.
20.       # What arguments to pass to octoprint, usually no need to touch this
21.       DAEMON_ARGS="--port=$PORT"
22.
23.       # Umask of files octoprint generates, Change this to 000 if running
    octoprint as its own, separate user
24.       UMASK=022
25.
26.       # Process priority, 0 here will result in a priority 20 process.
27.       # -2 ensures Octoprint has a slight priority over user processes.
28.       NICELEVEL=-2
29.
30.       # Should we run at startup?
31.       START=yes
```

Una vez tenemos tantos scripts como instancias de OctoPrint procederemos a lanzar los scripts de la siguiente manera. En nuestro caso los lanzamos desde el usuario maestro de nuestra maquina (es el único que tiene permisos sudo).

```
1. sudo /etc/init.d/octoprint1 start
2. sudo /etc/init.d/octoprint2 start
3. sudo /etc/init.d/octoprint3 start
4. sudo /etc/init.d/octoprint4 start
5. sudo /etc/init.d/octoprint5 start
6. sudo /etc/init.d/octoprint6 start
```

7. Scripts cámaras

Una vez tengamos nuestras instancias de OctoPrint funcionando deberemos hacer lo propio con las cámaras para ello deberemos modificar los scripts que se encuentran en */usr/share/scripts* y en */etc/init.d/webcam1*

Puede que la carpeta “scripts” anteriormente citada no exista por defecto, por lo que la deberemos crear y colocar en ella los scripts correspondientes.

Al igual que pasaba con los scripts de los OctoPrint deberemos cambiar la variable “INSTANCENUMBER” por el número de cámara adecuado.
En este caso deberemos asegurarlo de cambiar también el “device” y “DispositivoVideo” por el nombre de la cámara que queramos modificar.

Por defecto las cámaras operan el puerto 8080, en este script lo hemos modificado para que la primera cámara opere en la 8081, la segunda en el 8082 y así sucesivamente.

Script /usr/share/scripts/webcam1:

```
1. #!/bin/bash
2.
3. ########################################################################
4. ### DO NOT EDIT THIS FILE TO CHANGE THE CONFIG!!!                    ###
```

```
### ------------------------------------------------------------------ ###
### There is no need to edit this file for changing resolution,       ###
### frame rates or any other mjpg-streamer parameters. Please edit    ###
### /boot/octopi.txt instead - that's what it's there for! You can    ###
### even do this with your Pi powered down by directly accessing the ###
    ### file when using the SD card as thumb drive in your regu-
lar        ###
    ### compu-
ter.                                                              ###
    ######################################################################
####

    MJPGSTREAMER_HOME=/usr/local/lib/mjpg-streamer
    MJPGSTREAMER_INPUT_USB="input_uvc.so"

    INSTANCENUMBER=1
    device="/dev/videoEYE"  #/dev/video
    DispositivoVideo="/dev/videoEYE"
    camera="auto"
    # camera_usb_options="-d /dev/video$INSTANCENUMBER -r 640x480 -f 10
-y"
    camera_usb_options="-d $DispositivoVideo -r 640x480 -f 10 -y"
    camera_raspi_options="-fps 10"
    camera_http_webroot="/usr/local/share/mjpg-streamer/www-octopi"
    camera_http_options="-n -p 808$INSTANCENUMBER"
    #camera_http_options="-p 808$INSTANCENUMBER"
    additional_brokenfps_usb_devices=()


    brokenfps_usb_devices=("046d:082b" "${additional_brokenfps_usb_devi-
ces[@]}")

    # cleans up when the script receives a SIGINT or SIGTERM
```

```
        # make sure that

     child processed die when we die
        local pids=$(jobs -pr)
        [ -n "$pids" ] && kill $pids
        exit 0
    }

    # says goodbye when the script shuts down
    function goodbye() {
        # say goodbye
        echo ""
        echo "Goodbye..."
        echo ""
    }

    # runs MJPG Streamer, using the provided input plugin + configura-
tion
    function runMjpgStreamer {
        input=$1
        pushd $MJPGSTREAMER_HOME > /dev/null 2>&1
            echo Running ./mjpg_streamer -o "output_http.so -w /usr/lo-
cal/share/mjpg-streamer/www-octopi  $camera_http_options" -i "$input"
```

```
54.             LD_LIBRARY_PATH=. mjpg_streamer -o "output_http.so -w
    /usr/local/share/mjpg-streamer/www-octopi  $camera_http_options" -i "$in-
    put" &
55.             wait
56.         popd > /dev/null 2>&1
57.     }
58.
59.     # starts up the USB webcam
60.     function startUsb {
61.         options="$camera_usb_options"
62.         device="video$INSTANCENUMBER"
63.         extracted_device=`echo $options | sed 's@.*-d /dev/\(video[0-
    9]+\).*@\1@'`
64.         if [ "$extracted_device" != "$options" ]
65.         then
66.             # the camera options refer to another device, use that for
    determining product
67.             device=$extracted_device
68.         fi
69.
70.         uevent_file="/sys/class/video4linux/$device/device/uevent"
71.         if [ -e $uevent_file ]; then
72.             # let's see what kind of webcam we have here, fetch vid and
    pid...
73.             product=`cat $uevent_file | grep PRODUCT | cut -d"=" -f2`
74.             vid=`echo $product | cut -d"/" -f1`
75.             pid=`echo $product | cut -d"/" -f2`
76.             vidpid=`printf "%04x:%04x" "0x$vid" "0x$pid"`
77.
78.             # ... then look if it is in our list of known broken-fps-de-
    vices and if so remove
79.             # the -f parameter from the options (if it's in there, else
    that's just a no-op)
80.             for identifier in ${brokenfps_usb_devices[@]};
81.             do
82.                 if [ "$vidpid" = "$identifier" ]; then
83.                     echo
84.                     echo "Camera model $vidpid is known to not work with
    -f parameter, stripping it out"
85.                     echo
86.                     options=`echo $options | sed -e "s/\(\s\+\|^\)-
    f\s\+[0-9]\+//g"`
87.                 fi
88.             done
89.         fi
90.
91.         logger -s "Starting USB webcam"
92.         runMjpgStreamer "$MJPGSTREAMER_INPUT_USB $options"
93.     }
94.
95.     # make sure our cleanup function gets called when we receive SIGINT,
    SIGTERM
96.     trap "cleanup" SIGINT SIGTERM
97.     # say goodbye when we EXIT
98.     trap "goodbye" EXIT
99.
100.    # echo configuration
101.    echo "Starting up webcamDaemon..."
102.    echo ""
103.    echo "--- Configuration: ----------------------------"
104.    echo "camera:         $camera"
105.    echo "usb options:    $camera_usb_options"
106.    echo "raspi options: $camera_raspi_options"
107.    echo "http options:   -w $camera_http_webroot $camera_http_options"
```

```
108.     echo "-----------------------------------------------"
109.     echo ""
110.
111.     # we need this to prevent the later calls to vcgencmd from blocking
112.     # I have no idea why, but that's how it is...
113.     vcgencmd version > /dev/null 2>&1
114.
115.     # keep mjpg streamer running if some camera is attached
116.     while true; do
117.         if [ -e "$DispositivoVideo" ] && { [ "$ca-
   mera" = "auto" ] || [ "$camera" = "usb" ] ; }; then
118.             startUsb
119.             sleep 30 &
120.             wait
121.         else
122.             echo "No camera detected, trying again in two minutes"
123.             sleep 120 &
124.             wait
125.         fi
126.     done
```

Haremos los mismo para los scripts que se encuentran en el directorio */etc/init.d/*

Script /etc/init.d/webcamd1:

```
1. #!/bin/sh
2.
3. ### BEGIN INIT INFO
4. # Provides:          webcamd1
5. # Required-Start:    $local_fs networking
6. # Required-Stop:
7. # Should-Start:
8. # Should-Stop:
9. # Default-Start:     2 3 4 5
10.      # Default-Stop:      0 1 6
11.      # Short-Description: webcam daemon
12.      # Description:       Starts the OctoPi webcam daemon with the user
   specified config in
13.      #                    /etc/default/webcamd.
14.      ### END INIT INFO
15.
16.      # Author: Gina Haeussge
17.
18.      INSTANCENUMBER=1
19.      PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
20.      DESC="Webcam $INSTANCENUMBER Daemon"
21.      NAME="webcamd$INSTANCENUMBER"
22.      DAEMON=/usr/share/scripts/webcamd$INSTANCENUMBER
23.      USER=maestro
24.      PIDFILE=/var/run/$NAME$INSTANCENUMBER.pid
25.      PKGNAME=webcamd$INSTANCENUMBER
26.      SCRIPTNAME=/etc/init.d/$PKGNAME
27.      LOG=/var/log/webcamd$INSTANCENUMBER.log
28.
29.      # Read configuration variable file if it is present
30.      [ -r /etc/default/$PKGNAME ] && . /etc/default/$PKGNAME
31.
32.      # Exit if the octoprint is not installed
33.      [ -x "$DAEMON" ] || exit 0
34.
```

```
35.      # Load the VERBOSE setting and other rcS variables
36.      [ -f /etc/default/rcS ] && . /etc/default/rcS
37.
38.      # Define LSB log_* functions.
39.      # Depend on lsb-base (>= 3.0-6) to ensure that this file is present.
40.      . /lib/lsb/init-functions
41.
42.      if [ -z "$ENABLED" -o "$ENABLED" != "1" ]
43.      then
44.         log_warning_msg "Not starting $PKGNAME, edit /etc/de-
   fault/$PKGNAME to start it."
45.         exit 0
46.      fi
47.
48.      #
49.      # Function to verify if a pid is alive
50.      #
51.      is_alive()
52.      {
53.         pid=`cat $1` > /dev/null 2>&1
54.         kill -0 $pid > /dev/null 2>&1
55.         return $?
56.      }
57.
58.      #
59.      # Function that starts the daemon/service
60.      #
61.      do_start()
62.      {
63.         # Return
64.         #   0 if daemon has been started
65.         #   1 if daemon was already running
66.         #   2 if daemon could not be started
67.
68.         is_alive $PIDFILE
69.         RETVAL="$?"
70.
71.         if [ $RETVAL != 0 ]; then
72.             start-stop-daemon --start --background --no-close --quiet --
   pidfile $PIDFILE --make-pidfile \
73.             --startas /bin/bash --chuid $USER --user $USER -- -
   c "exec $DAEMON" >> $LOG 2>&1
74.             RETVAL="$?"
75.         fi
76.      }
77.
78.      #
79.      # Function that stops the daemon/service
80.      #
81.      do_stop()
82.      {
83.         # Return
84.         #   0 if daemon has been stopped
85.         #   1 if daemon was already stopped
86.         #   2 if daemon could not be stopped
87.         #   other if a failure occurred
88.
89.         start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --
   user $USER --pidfile $PIDFILE
90.         RETVAL="$?"
91.         [ "$RETVAL" = "2" ] && return 2
92.
93.         rm -f $PIDFILE
94.
```

```
95.          [ "$RETVAL" = "0"  ] && return 0 || return 1
96.      }
97.
98.      case "$1" in
99.        start)
100.        [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
101.        do_start
102.        case "$?" in
103.           0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
104.           2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
105.        esac
106.        ;;
107.      stop)
108.        [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
109.        do_stop
110.        case "$?" in
111.           0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
112.           2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
113.        esac
114.        ;;
115.      restart)
116.        log_daemon_msg "Restarting $DESC" "$NAME"
117.        do_stop
118.        case "$?" in
119.          0|1)
120.           do_start
121.           case "$?" in
122.              0) log_end_msg 0 ;;
123.              1) log_end_msg 1 ;; # Old process is still running
124.              *) log_end_msg 1 ;; # Failed to start
125.           esac
126.           ;;
127.          *)
128.             # Failed to stop
129.           log_end_msg 1
130.           ;;
131.        esac
132.        ;;
133.      *)
134.        echo "Usage: $SCRIPTNAME {start|stop|restart}" >&2
135.        exit 3
136.        ;;
137.     esac
```

Cuando tenemos todas las cámaras configuradas procederemos a lanzarlas de la siguiente manera (método recomendado):

```
1. sudo /etc/init.d/webcamd1 start
2. sudo /etc/init.d/webcamd2 start
3. sudo /etc/init.d/webcamd3 start
```

El comando start se puede sustituir por stop o restart dependiendo de nuestra necesidad en cada momento.

Si queremos lanzar las cámaras de manera manual (nos ocupara una terminal cada cámara que lancemos, por lo que solo se suele usar con la finalidad de pruebas) deberemos hacerlo con el

siguiente método:

Para añadir más cámaras deberemos copiar los scripts de las rutas */usr/share/scripts/webcamd1* y */etc/default/webcamd1* y sustituir el "INSTANCENUMBER" que se encuentra dentro de dichos archivos.

8. Monitor OctoPrint

Hemos creado un monitor de OctoPrint para poder visualizar el estado de todas las impresoras en una misma página web.
A continuación, copiado nuestro monitor de OctoPrint en la ruta */home/maestro/FlaskApp*
Debemos hacer algunos cambios en nuestro programa para que funcione el script. Deberemos incluir la siguiente línea en la primera línea de monitorOcto.py (incluido en la carpeta FlaskApp):

```
#!/usr/bin/python
```

Ademas, para que no tengamos problemas con las rutas deberemos indicar la ruta completa de los CSV donde se encuentran los datos de las impresoras.
Además, deberemos añadir:

```
1. import sys
2. import os
```

Y añadir las siguientes líneas para solucionar los problemas de ejecutar la aplicación fuera de su ruta predeterminada:

```
1. pathname = os.path.dirname(sys.argv[0])
2. os.chdir(os.path.abspath(pathname))
```

Y a continuación el resto de "imports" que nos faltan, de manera que los "imports" al completo quedarán de la siguiente manera:

```
1. #!/usr/bin/python
2.
3. import json as simplejson
4. import requests
5. import threading
6. import collections
7. import sys
8. import os
9. import csv
10.
11.      pathname = os.path.dirname(sys.argv[0])
12.      os.chdir(os.path.abspath(pathname))
13.
14.      from flask import Flask, render_template, redirect, url_for, re-
   quest, session, abort
15.      from flask import jsonify
16.      from flask import request
17.      from datetime import datetime,timedelta
```

```
18.       from sqlalchemy.orm import sessionmaker
19.       from tabledef import *
```

IMPORTANTE:
Nos debemos asegurar de que el final de línea del archivo esta en formato Linux, ya que si tiene el final de línea de tipo Windows tendremos problemas a la hora de ejecutar los scripts.

Para que funcione deberemos modificar el script que se encuentra en la ruta:
/etc/init.d/monitorOcto:

Hemos modificado el DESC, NAME y PKGNAME.

```
1.  #!/bin/sh
2.
3.  ### BEGIN INIT INFO
4.  # Provides:          octoprint
5.  # Required-Start:    $local_fs networking
6.  # Required-Stop:
7.  # Should-Start:
8.  # Should-Stop:
9.  # Default-Start:     2 3 4 5
10.       # Default-Stop:      0 1 6
11.       # Short-Description: OctoPrint daemon
12.       # Description:       Starts the OctoPrint daemon with the user spe-
    cified in
13.       #                    /etc/default/octoprint.
14.       ### END INIT INFO
15.
16.       # Author: Sami Olmari & Gina Häußge
17.
18.
19.       PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
20.       DESC="Monitor OctoPrint Daemon"
21.       NAME="Monitor OctoPrint"
22.       PKGNAME=monitorOcto
23.       PIDFILE=/var/run/$PKGNAME.pid
24.       SCRIPTNAME=/etc/init.d/$PKGNAME
25.       DEFAULTS=/etc/default/$PKGNAME
26.
27.       # Read configuration variable file if it is present
28.       [ -r $DEFAULTS ] && . $DEFAULTS
29.
30.       # Define LSB log_* functions.
31.       # Depend on lsb-base (>= 3.0-6) to ensure that this file is present.
32.       . /lib/lsb/init-functions
33.
34.       # Exit if the DAEMON is not set
35.       if [ -z "$DAEMON" ]
36.       then
37.           log_warning_msg "Not starting $PKGNAME, DAEMON not set in
    /etc/default/$PKGNAME."
38.           exit 0
39.       fi
40.
41.       # Exit if the DAEMON is not installed
42.       [ -x "$DAEMON" ] || exit 0
43.
44.       # Load the VERBOSE setting and other rcS variables
45.       [ -f /etc/default/rcS ] && . /etc/default/rcS
46.
47.       if [ -z "$START" -o "$START" != "yes" ]
```

```
48.      then
49.         log_warning_msg "Not starting $PKGNAME, edit /etc/de-
   fault/$PKGNAME to start it."
50.         exit 0
51.      fi
52.
53.      if [ -z "$OCTOPRINT_USER" ]
54.      then
55.          log_warning_msg "Not starting $PKGNAME, OCTOPRINT_USER not set
   in /etc/default/$PKGNAME."
56.          exit 0
57.      fi
58.
59.      COMMAND_ARGS=
60.      if [ -n "$BASEDIR" ]
61.      then
62.          COMMAND_ARGS="--basedir $BASEDIR $COMMAND_ARGS"
63.      fi
64.
65.      #if [ -n "$CONFIGFILE" ]
66.      #then
67.      #    COMMAND_ARGS="--config $CONFIGFILE $COMMAND_ARGS"
68.      #fi
69.
70.      #
71.      # Function to verify if a pid is alive
72.      #
73.      is_alive()
74.      {
75.         pid=`cat $1` > /dev/null 2>&1
76.         kill -0 $pid > /dev/null 2>&1
77.         return $?
78.      }
79.
80.      #
81.      # Function that starts the daemon/service
82.      #
83.      do_start()
84.      {
85.         # Return
86.         #   0 if daemon has been started
87.         #   1 if daemon was already running
88.         #   2 if daemon could not be started
89.
90.         is_alive $PIDFILE
91.         RETVAL="$?"
92.
93.         if [ $RETVAL != 0 ]; then
94.             echo "arrancando MonitorOctoprint"
95.             start-stop-daemon --start --background --quiet --pid-
   file $PIDFILE --make-pidfile \
96.             --exec $DAEMON --chuid $OCTOPRINT_USER --user $OCTO-
   PRINT_USER --umask $UMASK --nicelevel=$NICELEVEL #\
97.             #-- serve $COMMAND_ARGS $DAEMON_ARGS
98.             RETVAL="$?"
99.         fi
100.     }
101.
102.     #
103.     # Function that stops the daemon/service
104.     #
105.     do_stop()
106.     {
107.        # Return
```

```
108.         #   0 if daemon has been stopped
109.         #   1 if daemon was already stopped
110.         #   2 if daemon could not be stopped
111.         #   other if a failure occurred
112.
113.         start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --
   user $OCTOPRINT_USER --pidfile $PIDFILE
114.         RETVAL="$?"
115.         [ "$RETVAL" = "2" ] && return 2
116.
117.         rm -f $PIDFILE
118.
119.         [ "$RETVAL" = "0"  ] && return 0 || return 1
120.     }
121.
122.     case "$1" in
123.       start)
124.         [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
125.         do_start
126.         case "$?" in
127.             0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
128.             2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
129.         esac
130.         ;;
131.       stop)
132.         [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
133.         do_stop
134.         case "$?" in
135.             0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
136.             2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
137.         esac
138.         ;;
139.       status)
140.         status_of_proc -p $PIDFILE $DAEMON $NAME && exit 0 || exit $?
141.         ;;
142.       restart)
143.         log_daemon_msg "Restarting $DESC" "$NAME"
144.         do_stop
145.         case "$?" in
146.           0|1)
147.             do_start
148.             case "$?" in
149.                 0) log_end_msg 0 ;;
150.                 1) log_end_msg 1 ;; # Old process is still running
151.                 *) log_end_msg 1 ;; # Failed to start
152.             esac
153.             ;;
154.           *)
155.               # Failed to stop
156.             log_end_msg 1
157.             ;;
158.         esac
159.         ;;
160.       *)
161.         echo "Usage: $SCRIPTNAME {start|stop|status|restart}" >&2
162.         exit 3
163.         ;;
164.     esac
```

También deberemos hacer algún cambio en el siguiente script.
Hemos modificado el “OCTOPRINT_USER”, en “BASEDIR” le hemos indicado la ruta en la que se encuentra nuestra aplicación, el puerto sobre el que queremos que se ejecute (“PORT”) y por último

el “DAEMON” con el ejecutable de Python.

Script */etc/default/*:

```
# Configuration for /etc/init.d/monitorOcto


# The init.d script will only run if this variable non-empty.
OCTOPRINT_USER=maestro

# base directory to use
BASEDIR=/home/maestro/FlaskApp

      # configuration file to use
      #CONFIGFILE=/home/maestro/FlaskApp/config

      # On what port to run daemon, default is 5000
      PORT=8050

      # Path to the OctoPrint executable, you need to set this to match your installation!
      DAEMON=/home/maestro/FlaskApp/monitorOcto.py

      # What arguments to pass to octoprint, usually no need to touch this
      DAEMON_ARGS="--port=$PORT"

      # Umask of files octoprint generates, Change this to 000 if running octoprint as its own, separate user
      UMASK=022

      # Process priority, 0 here will result in a priority 20 process.
      # -2 ensures Octoprint has a slight priority over user processes.
      NICELEVEL=-2

      # Should we run at startup?
      START=yes
```

Una vez tengamos todo configurado utilizaremos el siguiente comando para ejecutar nuestro monitorOcto:

```
sudo /etc/init.d/monitorOcto start
```

Para que esto funcione hemos copiado los scripts de la ruta */etc/init.d/* y */etc/default/* y los hemos modificado con los nombre y las rutas de nuestra aplicación monitorOcto.

Por defecto el monitor de OctoPrint funcionará en la dirección: 192.168.1.200:8050. Pero si queremos que lo haga en otro puerto habrá que cambiar este parámetro tanto en los scripts anteriormente citados como en el método *run()* del propio monitorOcto.py

9. Controles OctoPrint

Para que funcionen las herramientas de apagar y reiniciar Octoprint desde su propia página hemos creado un grupo llamado impresores al que pertenecen todos los usuarios “impresor”. Para añadir los usuarios se puede hacer modificando el archivo “group” que se encuentra en la ruta sudo nano cd /etc/group. Ademas deberemos darle permisos sudo, ya que los comando de apagar y reiniciar lo necesitan. Para ello modificamos el archivo “sudoers” que se encuentra en la ruta /etc/. De forma que nuestro archivo quedaría de la siguiente manera:

```
sudo groupadd impresores
```

A continuación, mostraremos el archivo sudoers necesario para ejecutar todos los scripts y todas las ordenes desde OctoPrint sin tener ningún tipo de problema:

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
      Defaults        mail_badpass
      Defaults        secure_path="/usr/local/sbin:/usr/lo-
cal/bin:/usr/sbin:/usr/bin:/sbin:/bin"

      # Host alias specification

      # User alias specification

      # Cmnd alias specification

      # User privilege specification
      root    ALL=(ALL:ALL) ALL

      # Allow members of group sudo to execute any command
      %sudo   ALL=(ALL:ALL) ALL
      %maestro ALL=(ALL)ALL
      %maestro ALL=(ALL) NOPASSWD: /sbin/shutdown, /sbin/powe-
roff, /sbin/reboot
      %impresores ALL=(ALL) NOPASSWD: /sbin/shutdown, /sbin/powe-
roff, /sbin/reboot
      %impresores ALL=(ALL) NOPASSWD: /etc/init.d/octoprint?
      #%impresores ALL=(ALL) NOPASSWD: /etc/init.d/octoprint1
      #%impresores ALL=(ALL) NOPASSWD: /etc/init.d/octoprint2
      #%impresores ALL=(ALL) NOPASSWD: /etc/init.d/octoprint3
      #%impresores ALL=(ALL) NOPASSWD: /etc/init.d/octoprint4
      # See sudoers(5) for more information on "#include" directives:

      #includedir /etc/sudoers.d
```

10. Proxy Nginx

Vamos a configurar un proxy llamado nginx para poder acceder con el nombre de la máquina en lugar de una dirección y un puerto.

Lo primero que debemos hacer es instalar el proxy con el siguiente comando:

```
sudo apt-get install nginx
```

Una vez instalado procederemos a configurar el proxy. Para ello deberemos editar el archivo que mostrare a continuación:

```
sudo nano /etc/nginx/nginx.config
```

Para configurar cada máquina de OctoPrint deberemos añadir las siguientes líneas dentro del bloque HTML:

```
upstream "octoprint_1" {
     server 127.0.0.1:5001;
}
upstream "octoprint_2" {
    server 127.0.0.1:5002;
}
upstream "webcam1" {
    server 127.0.0.1:8081;
}
      upstream "webcam2" {
          server 127.0.0.1:8082;
      }
      upstream "monitorOcto" {
          server 127.0.0.1:8050;
      }
```

“octoprint_1” es el nombre que le hemos dado y server en la dirección que tiene cada máquina de OctoPrint

El resto de los archivos de configuración se encuentran en la ruta “/etc/nginx/sites-avaliable”. Dentro de esta carpeta deberemos editar el archivo “default” siguiendo la siguiente estructura:

El “server_name” es el nombre que debemos introducir en el navegador para que nos lleve a los OctoPrints o al monitor que hemos creado.

```
server_name linuxtipia.local www.linuxtipia.local;

location / {
                proxy_pass http://monitorOcto/;
                proxy_set_header Host $http_host;
                proxy_set_header Upgrade $http_upgrade;
```

```
                        proxy_set_header Connection "upgrade";
                        proxy_set_header X-Real-IP $remote_addr;
                        proxy_set_header X-Forwarded-For $proxy_add_x_for-
   warded_for;
                              proxy_set_header X-Scheme $scheme;
                              proxy_http_version 1.1;
                              proxy_set_header X-Script-Name /;

                              client_max_body_size 0;
                }

                location /3dp1/ {
                        proxy_pass http://octoprint_1/;
                        proxy_set_header Host $http_host;
                        proxy_set_header Upgrade $http_upgrade;
                        proxy_set_header Connection "upgrade";
                        proxy_set_header X-Real-IP $remote_addr;
                        proxy_set_header X-Forwarded-For $proxy_add_x_for-
   warded_for;
                        proxy_set_header X-Scheme $scheme;
                        proxy_http_version 1.1;
                        proxy_set_header X-Script-Name /3dp1;

                        client_max_body_size 0;
                }

```

Deberemos cambiar el texto del “location”, el “proxy_pass “y el “proxy_set_header X-Script-Name” para cada instancia de OctoPrint. En nuestro caso 3dp1, 3dp2, 3dp3…

Para configurar la página del monitor de Octoprint que hemos añadido el siguiente código en el archivo de configuración anteriormente citado

De esta manera cuando pongamos en el navegador “linuxtipia.local” nos cargará por defecto nuestro panel de control con todas las máquinas de OctoPrint.

Y si ponemos “linuxtipia.local/3dp1” nos cargará la página de OctoPrint con la impresora 1.

En este manual solo hemos visto la configuración para una impresora, esto lo deberemos replicar y modificar los apartados determinados para que funcionen las demás impresoras.

Una vez tengamos toda hayamos cambiado toda la información de las impresoras deberemos lanzar el servicio de la siguiente manera:

```
sudo service nginx start
```

Se podrá usar también el comando stop para parar dicho servicio.

11. Scripts al inicio

Una vez tenemos todos los scripts en la carpeta /etc/init.d y hemos comprobado que funcionan correctamente individualmente vamos a proceder configurar la máquina para que cuando se inicie

arranquen todos las instancias de OctoPrint, las cámaras y nuestro monitor de OctoPrint.

Deberemos ejecutar los siguientes comandos con todos los scripts que queremos que se ejecuten al inicio:

```
1. sudo update-rc.d octoprint1 defaults
2. sudo update-rc.d octoprint1 enable
3. sudo update-rc.d octoprint2 defaults
4. sudo update-rc.d octoprint2 enable
5. sudo update-rc.d octoprint3 defaults
6. sudo update-rc.d octoprint3 enable
7. sudo update-rc.d octoprint4 defaults
8. sudo update-rc.d octoprint4 enable
9. sudo update-rc.d octoprint5 defaults
10.      sudo update-rc.d octoprint5 enable
11.      sudo update-rc.d octoprint6 defaults
12.      sudo update-rc.d octoprint6 enable
13.      sudo update-rc.d octoprint7 defaults
14.      sudo update-rc.d octoprint7 enable
15.      sudo update-rc.d monitorOcto defaults
16.      sudo update-rc.d monitorOcto enable
17.      sudo update-rc.d webcamd1 defaults
18.      sudo update-rc.d webcamd1 enable
19.      sudo update-rc.d webcamd2 defaults
20.      sudo update-rc.d webcamd2 enable
21.      sudo update-rc.d webcamd3 defaults
22.      sudo update-rc.d webcamd3 enable
```

En nuestro caso hemos ejecutado las instancias de los siete OctoPrint, el monitor y las tres camaras.