mjpeg stream server from webcam

https://gist.github.com/n3wtron/4624820

Requires:

```
pip3 install opencv-python pillow
```

Usage:

```
Define camera in line 72 cv2.VideoCapture(2) refers to /dev/video2
Define resolution in lines 73/74
Define port in line 88

Browser with embedded live feed: http://ip:1339/index.html
Direct live feed: http://ip:1339/cam.mjpg
Snapshot image: http://ip:1339/still.jpg
```

```
#!/usr/bin/python3
'''
    Author: Igor Maculan - n3wtron@gmail.com
    A Simple mjpg stream http server
    Converted to python3 and additional fixes
'''
import cv2
from PIL import Image
import threading
from http.server import BaseHTTPRequestHandler,HTTPServer
from socketserver import ThreadingMixIn
from io import StringIO,BytesIO
from urllib.parse import urljoin, urlparse
import time
capture=None

class CamHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        #remove query string from url
        self.path=urljoin(self.path, urlparse(self.path).path)

        if self.path.endswith('.jpg'):
            self.send_response(200)
            self.send_header('Content-type','image/jpeg')
            self.send_header('Cache-Control','no-cache')
            self.send_header('Pragma','no-cache')
            self.end_headers()
            try:
                rc,img = capture.read()
                if not rc:
                    return
```

```
                imgRGB=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
                jpg = Image.fromarray(imgRGB)
                jpg.save(self.wfile,'JPEG')
            except:
                return
        if self.path.endswith('.mjpg'):
            self.send_response(200)
            self.send_header('Content-type','multipart/x-mixed-replace;
boundary=jpgboundary')
            self.end_headers()
            while True:
                try:
                    rc,img = capture.read()
                    if not rc:
                        continue
                    imgRGB=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
                    jpg = Image.fromarray(imgRGB)
                    tmpFile = BytesIO()
                    jpg.save(tmpFile,'JPEG')
                    self.wfile.write("\r\n--jpgboundary\r\n".encode())
                    self.send_header('Content-type','image/jpeg')
                    self.send_header('Content-
length',str(tmpFile.getbuffer().nbytes))
                    self.end_headers()
                    jpg.save(self.wfile,'JPEG')
                    time.sleep(0.05)
                except KeyboardInterrupt:
                    self.wfile.write("\r\n--jpgboundary--\r\n").encode()
                    break
                except BrokenPipeError:
                    continue
            return
        if self.path.endswith('.html'):
            self.send_response(200)
            self.send_header('Content-type','text/html')
            self.end_headers()
            self.wfile.write('<html><head></head><body>'.encode())
            self.wfile.write('<img src="cam.mjpg"/>'.encode())
            self.wfile.write('</body></html>'.encode())
            return


class ThreadedHTTPServer(ThreadingMixIn, HTTPServer):
    """Handle requests in a separate thread."""

def main():
    global capture
    capture = cv2.VideoCapture(2)
    capture.set(cv2.CAP_PROP_FRAME_WIDTH, 640);
    capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 480);
```

```
    capture.set(cv2.CAP_PROP_SATURATION,0.2);
    capture.set(cv2.CAP_PROP_BRIGHTNESS, .8);
#    capture.set(cv2.CAP_PROP_AUTO_EXPOSURE, 0.25) # 0.25 is turn OFF auto
exposure (Logitech); 0.75 is ON
#    time.sleep(.5) # wait for auto exposure change to be set
#    capture.set(cv2.CAP_PROP_EXPOSURE, .01) # fairly dark - low exposure
# a few other properties that can be set - not a complete list
#    capture.set(cv2.CAP_PROP_BRIGHTNESS, .4); #1 is bright 0 or -1 is dark
.4 is fairly dark default Brightness  0.5019607843137255
#    capture.set(cv2.CAP_PROP_CONTRAST, 1);
#    capture.set(cv2.CAP_PROP_FRAME_WIDTH, 320);
#    capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 240);
#    capture.set(cv2.CAP_PROP_SATURATION,0.2);
    global img
    try:
        server = ThreadedHTTPServer(('0.0.0.0', 1339), CamHandler)
        print( "server started")
        server.serve_forever()
    except KeyboardInterrupt:
        capture.release()
        server.socket.close()

if __name__ == '__main__':
    main()
```