

Main Python 프로그램 [serverUsb.py]

Main Python 프로그램

serverUsb.py

```
from flask import Flask, render_template, send_from_directory, Response, request, send_file
# from flask_socketio import SocketIO
from pathlib import Path
from capture import capture_and_save
from usbwebcamera import UsbWebCamera
import argparse, logging, logging.config, conf
import os
from gpioControl import GpioControl
from listenSpeech import ListenSpeech
from urllib.parse import parse_qs

gpio = GpioControl()
archive_path = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'archive')
listenspeech = ListenSpeech()

logging.config.dictConfig(conf.dictConfig)
logger = logging.getLogger(__name__)

usbcamera = UsbWebCamera(20,0)
usbcamera.run()

app = Flask(__name__)
#app.static_url_path = '/home/pi/Work/opencv/templates'
# app.config["SECRET_KEY"] = "secret!"
# socketio = SocketIO(app)

@app.after_request
def add_header(r):
    """
    Add headers to both force latest IE rendering or Chrome Frame,
    and also to cache the rendered page for 10 minutes
    """
    r.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    r.headers["Pragma"] = "no-cache"
    r.headers["Expires"] = "0"
```

```
r.headers["Cache-Control"] = "public, max-age=0"
return r
```

```
@app.route("/")
def entripoint1():
    logger.debug("Requested /")
    return render_template("index.html")
```

```
@app.route("/index.html")
def entripoint2():
    logger.debug("Requested /")
    return render_template("index.html")
```

```
@app.route("/msg", methods=['GET', 'POST'])
def msg():
    if request.method == 'POST':
        message = request.form['textinput']
        if not message :
            reString = "No received message"
        else :
            reString = listenspeech.speech(message)
        print("Get Message = " + reString)
    return reString
```

```
@app.route("/usbcapture")
def captureusb():
    logger.debug("Requested capture")
    im = usbcamera.get_frame(_bytes=False)
    capture_and_save(im)
    return render_template("send_to_init.html")
```

```
@app.route('/archive')
def archive():
    return render_template('archive.html')
```

```
def get_type(filename):
    name, extension = os.path.splitext(filename)
    return 'video' if extension == '.mp4' else 'audio' if extension == '.wav' else 'audio' if extension == '.mp3' else
'photo'
```

```
@app.route('/archive/<string:filename>')
def archive_item(filename):
    name, extension = os.path.splitext(filename)
    type = get_type(filename)
    return render_template('record.html', filename=filename, type=type)
```

```
@app.route('/archive/delete/<string:filename>')
def archive_delete(filename):
    os.remove(archive_path + "/" + filename)
```

```

    return redirect(url_for('archive'))

@app.route('/archive/play/<string:filename>')
def archive_play(filename):
    return send_file('archive/' + filename)

def get_records():
    records = []

    for filename in sorted(os.listdir(archive_path), reverse=True):
        if not filename.startswith('.'):
            type = get_type(filename)
            size = byte_to_mb(os.path.getsize(archive_path + "/" + filename))
            record = {"filename": filename, 'size': size, 'type': type}
            records.append(record)

    return records

def byte_to_mb(byte):
    mb = "{:.2f}".format(byte / 1024 / 1024)
    return str(mb) + " MB"

app.jinja_env.globals.update(get_records=get_records)

@app.route("/archive/last")
def last_image():
    logger.debug("Requested last image")
    p = Path("archive/last.png")
    if p.exists():
        r = "last.png"
    else:
        logger.debug("No last image")
        r = "not_found.jpeg"
    return send_from_directory("archive",r)

def genusb(usbcamera):
    logger.debug("Starting USB stream")
    while True:
        frame = usbcamera.get_frame()
        yield (b'--frame\r\n'
              b'Content-Type: image/png\r\n\r\n' + frame + b'\r\n')

@app.route("/stream")
def stream_page():
    logger.debug("Requested stream page")
    return render_template("stream.html")

@app.route("/video_usb_feed")

```

```
def video_usb_feed():
    return Response(genusb(usbcamera),
        mimetype="multipart/x-mixed-replace; boundary=frame")
```

```
@app.route("/temperature")
```

```
def temperature():
    content = os.popen("vcgencmd measure_temp").readline()
    content = content.replace("temp=", "")
    return Response(content, mimetype='text/xml')
```

```
''' ##### Control Camera Section ##### '''
```

```
@app.route("/shooting", methods=['GET', 'POST'])
```

```
def shooting():
    reString = "Shooting : " + gpio.shooting()
    return Response(reString, mimetype='text/html')
```

```
@app.route("/init")
```

```
def init():
    reString = "Camera position init : " + gpio.initMotorPosition()
    print("Move init ...")
    return Response(reString, mimetype='text/html')
```

```
@app.route("/up")
```

```
def up():
    reString = gpio.moveUp()
    print("Move Up ...")
    return Response(reString, mimetype='text/html')
```

```
@app.route("/down")
```

```
def down():
    reString = gpio.moveDown()
    print("Move Down ...")
    return Response(reString, mimetype='text/html')
```

```
@app.route("/left")
```

```
def left():
    reString = gpio.moveLeft()
    print("Move Left ...")
    return Response(reString, mimetype='text/html')
```

```
@app.route("/right")
```

```
def right():
    reString = gpio.moveRight()
    print("Move Right ...")
    return Response(reString, mimetype='text/html')
```

```
''' ##### Caterpillar Tracks ##### '''
```

```
@app.route("/forward", methods=['GET', 'POST'])
```

```

def forward():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.goForward(moveOrder)
            print("Go Forward ...("+moveOrder+")")
    return reString

@app.route("/backward", methods=['GET', 'POST'])
def backward():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.goBackward(moveOrder)
            print("Go Backward ...("+moveOrder+")")
    return reString

@app.route("/turnleftback", methods=['GET', 'POST'])
def turnleftback():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.goTurnleftback(moveOrder)
            print("Go Turn Left back...("+moveOrder+")")
    return reString

@app.route("/turnrightback", methods=['GET', 'POST'])
def turnrightback():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.goTurnrightback(moveOrder)
            print("Go Turn Right back...("+moveOrder+")")
    return reString

@app.route("/turnleftforward", methods=['GET', 'POST'])
def turnleftforward():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :

```

```
    reString = gpio.goTurnleftforward(moveOrder)
    print("Go Turn Left forward...("+moveOrder+")")
    return reString
```

```
@app.route("/turnrightforward", methods=['GET', 'POST'])
```

```
def turnrightforward():
```

```
    if request.method == 'POST':
```

```
        moveOrder = request.form['moveCnt']
```

```
        if not moveOrder :
```

```
            reString = "Do not move"
```

```
        else :
```

```
            reString = gpio.goTurnrightforward(moveOrder)
```

```
            print("Go Turn Right forward...("+moveOrder+")")
```

```
    return reString
```

```
if __name__=="__main__":
```

```
    # socketio.run(app,host="0.0.0.0",port="3005",threaded=True)
```

```
    parser = argparse.ArgumentParser()
```

```
    parser.add_argument('-p','--port',type=int,default=8081, help="Running port")
```

```
    parser.add_argument("-H","--host",type=str,default='0.0.0.0', help="Address to broadcast")
```

```
    args = parser.parse_args()
```

```
    logger.debug("Starting server")
```

```
    #app.run(host=args.host, port=args.port, threaded=True, debug=True)
```

```
    app.run(host=args.host, port=args.port, threaded=True)
```

🔄Revision #2

★Created 30 December 2023 13:49:41 by Hyeon Su Ryu

✎Updated 30 December 2023 14:23:53 by Hyeon Su Ryu