CCTV

http://www.kkorack.com

sudo apt install python3-opencv

- Start Stop
- Main Program
- USB Camera program
- Camera Movement program
- Speech program
- Postgresql program
- Python 로그인 설명
- 기타 연계 프로그램

Start Stop

Start

```
#!/bin/bash

echo
echo '##### START KKo KKo Rack #####'
echo

wpa_cli -i wlan0 status
echo
echo

FILENAME=/home/pi/stepperMotor/nohup.out
if [ -f "$FILENAME" ] ; then
        echo "nohup.out delete !"
        rm /home/pi/stepperMotor/nohup.out
else
        echo "file not exist"
fi

echo '##### START CCTV #####'
cd /home/pi/stepperMotor
nohup python3 /home/pi/stepperMotor/stepperMoter.py &
sleep 3

echo '##### START BROD SOUND #####'
nohup cvlc -vvv alsa://plughw:1 --sout
'#transcode{acodec=mp3,ab=64,channels=1}:standard{access=http,dst=0.0.0.0:8080/
out.mp3}' 1> /dev/null 2>&1 &
sleep 3

echo
echo '##### STARTed KKo KKo Rack #####'
echo
```

Stop

```
#!/bin/bash

echo
echo '##### STOP  KKo KKo Rack #####'
echo


ps -ef | grep stepperMoter.py | grep -v grep
KILLPID=`ps -ef | grep stepperMoter.py | grep -v grep | awk '{print($2)}'`
echo "Stop CCTV Process = " $KILLPID
kill -9 $KILLPID
sleep 3

ps -ef | grep vlc | grep -v grep
KILLPID=`ps -ef | grep vlc | grep -v grep | awk '{print($2)}'`
echo "Stop Brod Sound Process = " $KILLPID
kill -9 $KILLPID
sleep 3


echo
echo '##### STOPed KKo KKo Rack #####'
echo
```

Main Program

stepperMoter.py

메인 프로그램

```
from flask import Flask, render_template, send_from_directory, Response, send_file,
request, redirect, url_for
from flask_socketio import SocketIO
import argparse, logging, logging.config, conf
import os
from gpioControl import GpioControl
from time import sleep
from usbwebcamera import UsbWebCamera
from power import PowerStatus
from listenSpeech import ListenSpeech
from capture import capture_and_save
from postgresql import KKMsgDao
import json
from flask_sqlalchemy import SQLAlchemy
from models import db
from models import User
from flask import session
from flask_wtf.csrf import CSRFProtect #csrf
from form import RegisterForm, LoginForm



app = Flask(__name__)
app.config['SECRET_KEY'] = '17xxxx21q'

socketio = SocketIO(app)
archive_path = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'archive')

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

gpio = GpioControl()
power = PowerStatus()
listenspeech = ListenSpeech()

usbcamera = UsbWebCamera(video_source=0)
usbcamera.start()

postgresql = KKMsgDao()
```

```
@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('/', methods=['GET', 'POST'])
def mainpage():
    userid = session.get('userid',None)
    form = LoginForm() #로그인폼
    if userid is None:
        return render_template('login.html', form=form)
    else:
        msgList = postgresql.getMsg()
        return render_template("index.html", msgList=msgList, userid=userid)


@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm() #로그인폼
    try:
        if form.validate_on_submit(): #유효성 검사
            logger.debug('{}가 로그인 했습니다'.format(form.data.get('userid')))
            session['userid']=form.data.get('userid') #form에서 가져온 userid를 세션에 저장
            msgList = postgresql.getMsg()
            return render_template("index.html", msgList=msgList,
userid=form.data.get('userid')) #성공하면 main.html로
    except ValueError as ve:
        logger.debug('로그인 실패 ' + str(ve))
        return render_template('login.html', form=form, message=str(ve))

@app.route('/register', methods=['GET','POST'])  #겟, 포스트 메소드 둘다 사용
def register():   #get 요청 단순히 페이지 표시 post요청 회원가입-등록을 눌렀을때 정보 가져오는것
    form = RegisterForm()
    if form.validate_on_submit(): # POST검사의 유효성검사가 정상적으로 되었는지 확인할 수 있다. 입력 안 한것들이 있는지 확인됨.
    #비밀번호 = 비밀번호 확인 -> EqulaTo

        user = User()  #models.py에 있는 user
        user.userid = form.data.get('userid')
        user.username = form.data.get('username')
        user.email = form.data.get('email')
        user.password = form.data.get('password')
```

```
        print(user.userid,user.password)   #회원가입 요청시 콘솔창에 ID만 출력 (확인용, 딱히 필요없음)
        db.session.add(user)   # id, name 변수에 넣은 회원정보 DB에 저장
        db.session.commit()   #커밋

        return "가입 완료" #post요청일시는 '/'주소로 이동. (회원가입 완료시 화면이동)

    return render_template('register.html', form=form)


@app.route("/index.html")
def index():
    logger.debug("Requested /")

    userid = session.get('userid',None)
    if userid is None:
        return render_template('login.html', form=form)

    msgList = postgresql.getMsg()
    return render_template("index.html", msgList=msgList)

@app.route("/video_usb_feed")
def video_usb_feed():
    userid = session.get('userid',None)
    if userid is None:
        return Response(None,
            mimetype="multipart/x-mixed-replace; boundary=frame")
    return Response(genusb(usbcamera),
        mimetype="multipart/x-mixed-replace; boundary=frame")

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("/temperature")
def temperature():
    content = os.popen("vcgencmd measure_temp").readline()
    content = content.replace("temp=", "")
    powerstatus = power.getPowerStatus()
    return Response(content+"["+powerstatus+"]", mimetype='text/xml')


''' ##### Control Camera Section ##### '''
@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("/up", methods=['GET', 'POST'])
def up():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.moveUp(int(moveOrder))
        print("Move Up ..." + moveOrder)
    return Response(reString, mimetype='text/html')

@app.route("/down", methods=['GET', 'POST'])
def down():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.moveDown(int(moveOrder))
        print("Move Down ..." + moveOrder)
    return Response(reString, mimetype='text/html')

@app.route("/left", methods=['GET', 'POST'])
def left():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.moveLeft(int(moveOrder))
        print("Move Left ..." + moveOrder)
    return Response(reString, mimetype='text/html')

@app.route("/right", methods=['GET', 'POST'])
def right():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.moveRight(int(moveOrder))
        print("Move Right ..." + moveOrder)
    return Response(reString, mimetype='text/html')
```

```
''' ##### Achive File Section ##### '''
@app.route("/usbcapture")
def captureusb():
    logger.debug("Requested USBCAM capture")
    im = usbcamera.get_captureFrame()
    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)
```

```
''' ##### Achive File Section ##### '''
def messageReceived(methods=['GET', 'POST']):
    logger.debug('message was received!!!')

@socketio.on('my event')
def handle_my_custom_event(jsonMsg, methods=['GET', 'POST']):
    jsonObj = json.loads(str(jsonMsg).replace("'", "\""))
    #logger.debug('User name =' + str(jsonObj.get("user_name")))
    username = jsonObj.get("user_name")
    message = jsonObj.get("message")
    clientIP = request.environ.get('HTTP_X_REAL_IP', request.remote_addr)
    if username and message and clientIP:
        #logger.debug(">>> INSERT")
        postgresql.insMsg(username, message, clientIP)
        socketio.emit('my response', jsonMsg, callback=messageReceived)
    else:
        logger.debug("NONE")

''' ##### Firing File Section ##### '''
@app.route('/firing', methods=['GET', 'POST'])
def kkr_fire():
    gpio.fire()
    logger.debug('message KK Fired !!!')
    return Response('Fired !', mimetype='text/html')


@app.route("/favorit.ico")
def favorit_ico():
    logger.debug("Requested favorit.ico image")
    filename = "favorit.ico"
    return send_file(filename)


if __name__=="__main__":

  #데이터베이스---------
    basedir = os.path.abspath(os.path.dirname(__file__)) #현재 파일이 있는 디렉토리 절대 경로
    dbfile = os.path.join(basedir, 'db.sqlite') #데이터베이스 파일을 만든다

    app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///' + dbfile
    app.config['SQLALCHEMY_COMMIT_ON_TEARDOWN'] = True #사용자에게 정보 전달완료하면 teadown.
그 때마다 커밋=DB반영
    app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False #추가 메모리를 사용하므로 꺼둔다

#    db = SQLAlchemy() #SQLAlchemy를 사용해 데이터베이스 저장
    db.init_app(app) #app설정값 초기화
    db.app = app #Models.py에서 db를 가져와서 db.app에 app을 명시적으로 넣는다
```

```
    with app.app_context():
        db.create_all() #DB생성
    logger.debug("Starting VisionK server")
    socketio.run(app, log_output=True, host='0.0.0.0', port=8081, debug=True,
use_reloader=False)
```

USB Camera program

USB 카메라 구동 프로그램

```
import os
import sys
import time
import math
import getopt
import numpy as np
import cv2
import threading
import subprocess
from collections import deque
from lock_manager import Lock_Manager
from util import Util

class UsbWebCamera(threading.Thread):
    def __init__(self, fps=10, video_source=0, source=None):

        threading.Thread.__init__(self)

        self.name = self.__class__.__name__
        self.archive = os.path.join(os.path.dirname(os.path.realpath(__file__)),
'archive')

        self.current_frame = None

        self.codec = cv2.VideoWriter_fourcc('M','J', 'P', 'G')
        self.OBSERVER_LENGTH = 5 # Time in seconds to be observed for motion
        self.threshold = 15

        self.CAMERA_SOURCE = video_source
        self.REMAIN_RECORDING_FILES = 10 # 10이상 부터 삭제 후 저장

        self.source = cv2.VideoCapture(self.CAMERA_SOURCE) if source is not None else
self.init_camera()

        #self.fps = self.find_fps(self.source)
        self.fps = fps
        Util.log(self.name, "Setting FPS to " + str(self.fps))
        self.source.set(cv2.CAP_PROP_FPS,self.fps)
        #self.height, self.width = self.get_dimensions(self.source)
        Util.log(self.name, "Initializing pi camera class with video_source=" +
str(self.CAMERA_SOURCE))
        #Util.log(self.name,"width: {"+str(self.width)+"}, height :
```

```
{"+str(self.height)+"}")
        self.lock_manager = Lock_Manager("motion")

    def __del__(self):
        self.source.release()
        cv2.destroyAllWindows()
        self.lock_manager.remove()

    def get_captureFrame(self):
        return self.current_frame if self.current_frame is not None else None

    def get_frame(self):
        return self.frame_to_jpg(self.current_frame) if self.current_frame is not None
else None

    def frame_to_jpg(self, frame):
        ret, jpeg = cv2.imencode('.jpg', self.current_frame)
        return jpeg.tobytes()

    def get_dimensions(self, source):
        frame = cv2.cvtColor(source.read()[1],cv2.COLOR_RGB2GRAY)
        return frame.shape[0: 2]

    def find_fps(self, source):
        Util.log(self.name, "Determining FPS...")
        num_frames = 120
        start = time.time()
        for i in range(0, num_frames):
            ret, frame = source.read()
        end = time.time()
        fps = int(math.floor(num_frames / (end - start)))
        Util.log(self.name, "Setting FPS to " + str(fps))
        return fps

    def init_camera(self):
        camera = cv2.VideoCapture(self.CAMERA_SOURCE)
        time.sleep(0.5)

        return camera

    def run(self):
        while True:
            (grabbed, self.current_frame) = self.source.read()
            # End of feed
            if not grabbed:
                break
```

Camera Movement program

Camera Movement program

```
import RPi.GPIO as GPIO
from time import sleep

out1 = 13
out2 = 11
out3 = 15
out4 = 12

#out5 = 33
#out6 = 31
#out7 = 35
#out8 = 32

out5 = 32
out6 = 35
out7 = 31
out8 = 33

fire = 16

#GPIO.setmode(GPIO.BOARD)
#GPIO.setup(out1,GPIO.OUT)
#GPIO.setup(out2,GPIO.OUT)
#GPIO.setup(out3,GPIO.OUT)
#GPIO.setup(out4,GPIO.OUT)

#GPIO.setup(out5,GPIO.OUT)
#GPIO.setup(out6,GPIO.OUT)
#GPIO.setup(out7,GPIO.OUT)
#GPIO.setup(out8,GPIO.OUT)


class GpioControl(object):

    def __init__(self):
        print("> Init GpioControl !")

    def __del__(self):
        print(" > GPIO.__del__() ")
        GPIO.cleanup()
```

```
    def startUp(self):
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(out1,GPIO.OUT)
        GPIO.setup(out2,GPIO.OUT)
        GPIO.setup(out3,GPIO.OUT)
        GPIO.setup(out4,GPIO.OUT)
        GPIO.setup(out5,GPIO.OUT)
        GPIO.setup(out6,GPIO.OUT)
        GPIO.setup(out7,GPIO.OUT)
        GPIO.setup(out8,GPIO.OUT)

    def cleanUp(self):
        print(" > GPIO.cleanUp() ")
        GPIO.cleanup()
        sleep(2)

    def moveUp(self, movestep):
        print(" > GPIO.moveUp() " + str(movestep))
        self.upMove(movestep)
        return "GPIO.moveUp()"

    def moveDown(self, movestep):
        print(" > GPIO.moveDown() " + str(movestep))
        self.downMove(movestep)
        return "GPIO.moveDown()"

    def moveRight(self, movestep):
        print(" > GPIO.moveRight() " + str(movestep))
        self.rightMove(movestep)
        return "GPIO.moveRight()"

    def moveLeft(self, movestep):
        print(" > GPIO.moveLeft() " + str(movestep))
        self.leftMove(movestep)
        return "GPIO.moveLeft()"

    def fire(self):
        print(" > Fired ! ")
        self.fire()
        return "Fired !"

# Function #

    def rightMove(self, stepCnt):
        i=0
        y=0
        #self.initGpio()
        self.startUp()
        for y in range(stepCnt,0,-1):
```

```
            if i==0:
                i=7
            else:
                i=i-1
            #print(str((stepCnt)-y) + " , " + str(y) + " , " + str(i))
            self.MoveLeftRight(i)
        GPIO.cleanup()

    def leftMove(self, stepCnt):
        i=0
        y=0
        #self.initGpio()
        self.startUp()
        for y in range(stepCnt,0,-1):
            if i==7:
                i=0
            else:
                i=i+1
            #print(str((stepCnt)-y) + " , " + str(y) + " , " + str(i))
            self.MoveLeftRight(i)
        GPIO.cleanup()

    def upMove(self, stepCnt):
        i=0
        y=0
        #self.initGpio()
        self.startUp()
        for y in range(stepCnt,0,-1):
            if i==0:
                i=7
            else:
                i=i-1
            #print(str((stepCnt)-y) + " , " + str(y) + " , " + str(i))
            self.MoveUpDown(i)
        GPIO.cleanup()

    def downMove(self, stepCnt):
        i=0
        y=0
        #self.initGpio()
        self.startUp()
        for y in range(stepCnt,0,-1):
            if i==7:
                i=0
            else:
                i=i+1
            #print(str((stepCnt)-y) + " , " + str(y) + " , " + str(i))
            self.MoveUpDown(i)
        GPIO.cleanup()
```

```
    def initGpio(self):
        GPIO.output(out1,GPIO.LOW)
        GPIO.output(out2,GPIO.LOW)
        GPIO.output(out3,GPIO.LOW)
        GPIO.output(out4,GPIO.LOW)
        GPIO.output(out5,GPIO.LOW)
        GPIO.output(out6,GPIO.LOW)
        GPIO.output(out7,GPIO.LOW)
        GPIO.output(out8,GPIO.LOW)

    def MoveLeftRight(self, position):
        if position==0:
          GPIO.output(out1,GPIO.HIGH)
          GPIO.output(out2,GPIO.LOW)
          GPIO.output(out3,GPIO.LOW)
          GPIO.output(out4,GPIO.LOW)
          sleep(0.03)
          #time.sleep(1)
        elif position==1:
          GPIO.output(out1,GPIO.HIGH)
          GPIO.output(out2,GPIO.HIGH)
          GPIO.output(out3,GPIO.LOW)
          GPIO.output(out4,GPIO.LOW)
          sleep(0.03)
          #time.sleep(1)
        elif position==2:
          GPIO.output(out1,GPIO.LOW)
          GPIO.output(out2,GPIO.HIGH)
          GPIO.output(out3,GPIO.LOW)
          GPIO.output(out4,GPIO.LOW)
          sleep(0.03)
          #time.sleep(1)
        elif position==3:
          GPIO.output(out1,GPIO.LOW)
          GPIO.output(out2,GPIO.HIGH)
          GPIO.output(out3,GPIO.HIGH)
          GPIO.output(out4,GPIO.LOW)
          sleep(0.03)
          #time.sleep(1)
        elif position==4:
          GPIO.output(out1,GPIO.LOW)
          GPIO.output(out2,GPIO.LOW)
          GPIO.output(out3,GPIO.HIGH)
          GPIO.output(out4,GPIO.LOW)
          sleep(0.03)
          #time.sleep(1)
        elif position==5:
          GPIO.output(out1,GPIO.LOW)
          GPIO.output(out2,GPIO.LOW)
          GPIO.output(out3,GPIO.HIGH)
```

```
        GPIO.output(out4,GPIO.HIGH)
        sleep(0.03)
        #time.sleep(1)
    elif position==6:
        GPIO.output(out1,GPIO.LOW)
        GPIO.output(out2,GPIO.LOW)
        GPIO.output(out3,GPIO.LOW)
        GPIO.output(out4,GPIO.HIGH)
        sleep(0.03)
        #time.sleep(1)
    elif position==7:
        GPIO.output(out1,GPIO.HIGH)
        GPIO.output(out2,GPIO.LOW)
        GPIO.output(out3,GPIO.LOW)
        GPIO.output(out4,GPIO.HIGH)
        sleep(0.03)
        #time.sleep(1)

def MoveUpDown(self, position):
    if position==0:
        GPIO.output(out5,GPIO.HIGH)
        GPIO.output(out6,GPIO.LOW)
        GPIO.output(out7,GPIO.LOW)
        GPIO.output(out8,GPIO.LOW)
        sleep(0.03)
        #time.sleep(1)
    elif position==1:
        GPIO.output(out5,GPIO.HIGH)
        GPIO.output(out6,GPIO.HIGH)
        GPIO.output(out7,GPIO.LOW)
        GPIO.output(out8,GPIO.LOW)
        sleep(0.03)
        #time.sleep(1)
    elif position==2:
        GPIO.output(out5,GPIO.LOW)
        GPIO.output(out6,GPIO.HIGH)
        GPIO.output(out7,GPIO.LOW)
        GPIO.output(out8,GPIO.LOW)
        sleep(0.03)
        #time.sleep(1)
    elif position==3:
        GPIO.output(out5,GPIO.LOW)
        GPIO.output(out6,GPIO.HIGH)
        GPIO.output(out7,GPIO.HIGH)
        GPIO.output(out8,GPIO.LOW)
        sleep(0.03)
        #time.sleep(1)
    elif position==4:
        GPIO.output(out5,GPIO.LOW)
        GPIO.output(out6,GPIO.LOW)
```

```
            GPIO.output(out7,GPIO.HIGH)
            GPIO.output(out8,GPIO.LOW)
            sleep(0.03)
            #time.sleep(1)
        elif position==5:
            GPIO.output(out5,GPIO.LOW)
            GPIO.output(out6,GPIO.LOW)
            GPIO.output(out7,GPIO.HIGH)
            GPIO.output(out8,GPIO.HIGH)
            sleep(0.03)
            #time.sleep(1)
        elif position==6:
            GPIO.output(out5,GPIO.LOW)
            GPIO.output(out6,GPIO.LOW)
            GPIO.output(out7,GPIO.LOW)
            GPIO.output(out8,GPIO.HIGH)
            sleep(0.03)
            #time.sleep(1)
        elif position==7:
            GPIO.output(out5,GPIO.HIGH)
            GPIO.output(out6,GPIO.LOW)
            GPIO.output(out7,GPIO.LOW)
            GPIO.output(out8,GPIO.HIGH)
            sleep(0.03)
            #time.sleep(1)

    def fire(self):
         self.startUp()
         GPIO.setup(fire,GPIO.OUT)
         GPIO.output(fire,GPIO.HIGH)
         sleep(0.5)
         GPIO.output(fire,GPIO.LOW)
         GPIO.cleanup()

# Function #
```

Speech program

Speech program

```
from google_trans_new import google_translator
from google_speech import Speech
from time import sleep

translator = google_translator()

class ListenSpeech(object):
    def speech(self,message):
        print(' -> Message : ', message)
        ko_result = translator.translate(message, lang_tgt='ko')
        print(' -> ', ko_result)
        speech = Speech(ko_result, 'ko')
        speech.play()
        en_result = translator.translate(message, lang_tgt='en')
        speech = Speech(en_result, 'en')
        speech.play()
        it_result = translator.translate(message, lang_tgt='it')
        speech = Speech(it_result, 'it')
        speech.play()
        ja_result = translator.translate(message, lang_tgt='ja')
        speech = Speech(ja_result, 'ja')
        speech.play()
        returnString = ko_result + " , EN=" + en_result + " , IT=" + it_result + ",
JA=" + ja_result
        print("Return message = " + returnString)
        return returnString

if __name__ == "__main__":
    print("START ...")
    listenspeech = ListenSpeech()
    listenspeech.speech("안녕하세요")
```

Postgresql program

Postgresql program

```
#!python

import psycopg2

class KKMsgDao:
     def __init__(self):
          pass

     def getMsg(self):
          ret = []
          db = psycopg2.connect(host='localhost',
dbname='kkrack',user='xxxxxxx',password='xxxxxxxx',port=5432)
          curs = db.cursor()
          sql = "SELECT * FROM KKRACK ORDER BY TIME DESC LIMIT 5;"
          curs.execute(sql)
          rows = curs.fetchall()
          for e in rows:
               temp = {'username':e[0],'message':e[1],'time':e[2],'etc':e[3] }
               ret.append(temp)
          db.commit()
          db.close()
          return ret


     def insMsg(self, username, message, clientIP):
          db = psycopg2.connect(host='localhost',
dbname='kkrack',user='xxxxxxxxx',password='xxxxxx',port=5432)
          curs = db.cursor()
          sql = '''insert into KKRACK (username, message, time, etc) values(%s,%s,
NOW(), %s)'''
          curs.execute(sql,(username, message, clientIP))
          db.commit()
          db.close()

if __name__ =='__main__':
     kkMsgDao = KKMsgDao()
     msg = kkMsgDao.getMsg()
     print(msg)
     #kkMsgDao.insEmp("hyunsu", "hahaha", "10.10.10.1")
```

Python 로그인 설명

Python 로그인 설명

메인 프로그램에서 "/" 유입

세션 여부를 판단해서 로그인 페이지나 메인 페이지로 이동

```
@app.route('/', methods=['GET', 'POST'])
def mainpage():
    userid = session.get('userid',None)
    form = LoginForm() #로그인폼
    if userid is None:
        return render_template('login.html', form=form)
    else:
        msgList = postgresql.getMsg()
        return render_template("index.html", msgList=msgList, userid=userid)
```

로그인 페이지

```
<html>
    <head>
        <meta charset='utf-8'/>
        <meta name='viewport' content="width=device-width, initial-scale=1, shrink-to-
fit=no"/>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/
css/bootstrap.min.css"
     integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/
iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
     integrity="sha384-q8i/
X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous">
     </script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/
popper.min.js"
     integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous">
     </script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/
bootstrap.min.js"
     integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/
nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous">
     </script>
    </head>
```

```
<body>

    <div class="container">
        <div class="row mt-5">
            <h1>로그인</h1>
        </div>
        <div class="row mt-5">
            <div class="col-12">
                <form method="POST" action="/login">
                    {{form.csrf_token}}
                    <div class="form-group">

                        {{form.userid.label("아이디")}}
                        {{form.userid(class="form-control", placeholder="아이디")}}
                    </div>

                    <div class="form-group">

                        {{form.password.label("비밀번호")}}
                        {%if form.password.errors%}
            <!-- 로그인 실패시 errors/ password필드의 모든 에러를 갖고있다. -->
                        {{form.password.errors.0}}
            <!--0 : 첫번째 에러메시지를 출력 하겠다 라는 뜻  -->
                        {%endif%}
                        {{form.password(class="form-control", placeholder="비밀번호")}}
                        <p>
                        {{message}}
                    </div>

                    <button type="submit" class="btn btn-primary">로그인</button>
                </form>
            </div>

        </div>
    </div>
</body>
</html>
```

메인 페이지 로그인 처리

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm() #로그인폼
    try:
        if form.validate_on_submit(): #유효성 검사
            logger.debug('{}가 로그인 했습니다'.format(form.data.get('userid')))
            session['userid']=form.data.get('userid') #form에서 가져온 userid를 세션에 저장
            msgList = postgresql.getMsg()
```

```
            return render_template("index.html", msgList=msgList,
userid=form.data.get('userid')) #성공하면 main.html로
    except ValueError as ve:
        logger.debug('로그인 실패 ' + str(ve))
        return render_template('login.html', form=form, message=str(ve))
```

Form

```
from flask_wtf import FlaskForm
from wtforms import StringField
from wtforms import PasswordField
from wtforms.validators import DataRequired, EqualTo
from models import User #models.py 가져옴

class RegisterForm(FlaskForm):
    userid = StringField('userid', validators=[DataRequired()])
    username = StringField('username', validators=[DataRequired()])
    email = StringField('email', validators=[DataRequired()])
    password = PasswordField('password', validators=[DataRequired(),
EqualTo('re_password')]) #비밀번호 확인
    re_password = PasswordField('re_password', validators=[DataRequired()])

class LoginForm(FlaskForm):
    class UserPassword(object):
        def __init__(self, message=None):
            self.message = message
        def __call__(self, form, field):
            userid = form['userid'].data
            password = field.data
            usertable = User.query.filter_by(userid=userid).first()
            if usertable is None or usertable.password != password:
                raise ValueError('로그인 실패 비밀번호 틀림')

    userid = StringField('userid', validators=[DataRequired('ID를 입력하여 주세요 !')])
    password = PasswordField('password', validators=[DataRequired('패스워드를 입력해 주세
요'), UserPassword()])
```

객체

```
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy() #SQLAlchemy를 사용해 데이터베이스 저장

class User(db.Model): #데이터 모델을 나타내는 객체 선언
    __tablename__ = 'user_table' #테이블 이름

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(32), unique=True, nullable=False)
    email = db.Column(db.String(32), unique=True, nullable=False)
```

```
    userid = db.Column(db.String(32), unique=True, nullable=False)
    password = db.Column(db.String(8), nullable=False)

#    def __init__(self, email, password):
#        self.email = email
#        self.set_password(password)
#
    def set_password(self, password):
        self.password = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password, password)
```

사용자 등록 페이지

```
<html>
    <head>
        <meta charset='utf-8'/>
        <meta name='viewport' content="width=device-width, initial-scale=1, shrink-to-
fit=no"/>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/
css/bootstrap.min.css"
     integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/
iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
     integrity="sha384-q8i/
X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous">
     </script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/
popper.min.js"
     integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous">
     </script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/
bootstrap.min.js"
     integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/
nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous">
     </script>
    </head>

<body>
    <div class="container">
        <div class="row mt-5">
            <h1>회원가입</h1>
        </div>
        <div class="row mt-5">
            <div class="col-12">
```

```
                <form method="POST" >    <!--엑션 삭제해야됨-->
                    {{form.csrf_token}}
                    <div class="form-group">
                        <!-- <label for="userid">아이디</label>
                        <input type="text" class="form-control" id="userid"
placeholder="아이디" name="userid" /> -->
                        {{form.userid.label("아이디")}}
                        {{form.userid(class="form-control", placeholder="아이디")}}
                    </div>
                    <div class="form-group">
                        <!-- <label for="username">이름</label>
                        <input type="text" class="form-control" id="username"
placeholder="사용자이름" name="username" /> -->
                        {{form.username.label("사용자 이름")}}
                        {{form.username(class="form-control", placeholder="사용자 이
름")}}
                    </div>
                    <div class="form-group">
                        <!-- <label for="username">이메일</label>
                        <input type="text" class="form-control" id="email"
placeholder="사용자메일" name="email" /> -->
                        {{form.email.label("사용자 이메일")}}
                        {{form.email(class="form-control", placeholder="사용자 이메일")}}
                    </div>
                    <div class="form-group">
                        <!-- <label for="password">비밀번호</label>
                        <input type="password" class="form-control" id="password"
placeholder="비밀번호" name="password" /> -->
                        {{form.password.label("비밀번호")}}
                        {{form.password(class="form-control", placeholder="비밀번호")}}
                    </div>
                    <div class="form-group">
                        <!-- <label for="re-password">비밀번호 확인</label>
                        <input type="password" class="form-control" id="re_password"
placeholder="비밀번호 확인" name="re-password" /> -->
                        {{form.re_password.label("비밀번호 확인")}}
                        {{form.re_password(class="form-control", placeholder="비밀번호 확
인")}}
                    </div>
                    <button type="submit" class="btn btn-primary">제출</button>
                </form>
            </div>

        </div>
    </div>
</body>
</html>
```

메인 페이지 사용자 등록

```
@app.route('/register', methods=['GET','POST'])  #겟, 포스트 메소드 둘다 사용
def register():   #get 요청 단순히 페이지 표시 post요청 회원가입-등록을 눌렀을때 정보 가져오는것
    form = RegisterForm()
    if form.validate_on_submit(): # POST검사의 유효성검사가 정상적으로 되었는지 확인할 수 있다. 입력 안한것들이 있는지 확인됨.
    #비밀번호 = 비밀번호 확인 -> EqulaTo

        user = User()  #models.py에 있는 user
        user.userid = form.data.get('userid')
        user.username = form.data.get('username')
        user.email = form.data.get('email')
        user.password = form.data.get('password')

        print(user.userid,user.password)  #회원가입 요청시 콘솔창에 ID만 출력 (확인용, 딱히 필요없음)
        db.session.add(user)  # id, name 변수에 넣은 회원정보 DB에 저장
        db.session.commit()  #커밋

        return "가입 완료" #post요청일시는 '/'주소로 이동. (회원가입 완료시 화면이동)

    return render_template('register.html', form=form)
```

기타 연계 프로그램

기타 연계 프로그램

캡쳐

```
import os
import cv2
import datetime, time
from pathlib import Path

archive_path = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'archive')

def capture_and_save(im):
    s = im.shape
    # Add a timestamp
    font = cv2.FONT_HERSHEY_SIMPLEX
    bottomLeftCornerOfText = (10,s[0]-10)
    fontScale = 1
    fontColor = (20,20,20)
    lineType = 2

    cv2.putText(im,datetime.datetime.now().isoformat().split(".")
[0],bottomLeftCornerOfText,font,fontScale,fontColor, lineType)

    m = 0
    p = Path(archive_path)
    for imp in p.iterdir():
        if imp.suffix == ".png" and imp.stem != "last":
            num = imp.stem.split("_")[1]
            try:
                num = int(num)
                if num>m:
                    m = num
            except:
                print("Error reading image number for",str(imp))
    m +=1
    lp = Path(archive_path + "/last.png")
    if lp.exists() and lp.is_file():
        np = Path(archive_path + "/img_{}.png".format(m))
        np.write_bytes(lp.read_bytes())
    cv2.imwrite(archive_path + "/last.png",im)

if __name__=="__main__":
    capture_and_save()
    print("done")
```

환경 설정

```
from pathlib import Path

p = Path("logs")
if not p.exists():
    p.mkdir()

dictConfig = {
    'version': 1,
    'disable_existing_loggers': True,
    'formatters': {
        'standard': {
            'format': '%(asctime)s [%(levelname)s] %(name)s:: %(message)s',
        },
    },
    'handlers': {
        'default': {
            'level': 'DEBUG',
            'formatter': 'standard',
            'class': 'logging.StreamHandler',
            'stream': 'ext://sys.stdout',
        },
        'file': {
            'class': 'logging.handlers.RotatingFileHandler',
            'level': 'DEBUG',
            'formatter': 'standard',
            'filename': 'logs/logfile.log',
            'mode': 'a',
            'maxBytes': 5_242_880,
            'backupCount': 3,
            'encoding': 'utf-8',
        },
    },
    'loggers': {
        '__main__': {
            'handlers': ['default','file'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'camera': {
            'handlers': ['default', 'file'],
            'level': 'DEBUG',
            'propagate': False,
        },
    }
}
```

싱크로나이징

```
import os
import datetime

class Lock_Manager:
        def __init__(self, name):
                self.locks = os.path.dirname(os.path.realpath(__file__)) + "/locks"
                self.name = name + ".lock"
                self.path = os.path.join(self.locks, self.name)

                if not os.path.exists(self.locks):
                        os.makedirs(self.locks)

        def set(self):
                if not os.path.exists(self.path):
                        print("Setting {} lock".format(self.name))
                        timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")

                        with open(self.path, 'w+') as f:
                                f.write(timestamp)

        def remove(self):
                if os.path.isfile(self.path):
                        os.remove(self.path)

        def get_lock_time(self):
                for filename in os.listdir(self.locks):
                        with open(self.locks + "/" + filename) as f:
                                return f.read()

                return None
```

파워정보 조회

```
from vcgencmd import Vcgencmd

vcgm = Vcgencmd()

class PowerStatus(object):
    def __init__(self):
        print("> vcgm.version()=" + vcgm.version())

    def getPowerStatus(self):
        output=vcgm.get_throttled()
        print("raw_data=" + output['raw_data'])
        print("binary=" + output['binary'])
```

```
        return "raw_data=" + output['raw_data'] + ", binary=" + output['binary']
```

Data Util

```
from datetime import datetime

class Util:
        def log(source, msg):
                print(datetime.now().strftime("%Y%m%d_%H%M%S") + " | " + source + " |
" + str(msg))
```