# I2C LCD 모듈을 사용하기 위한 드라이버 및 기타 라이브러리 임포트

import I2C\_LCD\_driver

import time

# 운영 체제 관련 기능을 사용하기 위한 라이브러리 임포트

import os

# Raspberry Pi GPIO 라이브러리 임포트

import RPi.GPIO as GPIO

# I2C 통신을 위한 라이브러리 임포트

import smbus

# DS18B20 온도 센서를 사용하기 위한 라이브러리 설정

os.system('modprobe w1-gpio')

os.system('modprobe w1-therm')

# 스레딩을 위한 라이브러리 임포트

import threading

# Flask 웹 애플리케이션을 위한 라이브러리 임포트

from flask import Flask, request, jsonify, render\_template, Response

# 웹 페이지 리다이렉션을 위한 라이브러리 임포트

from werkzeug.utils import redirect

# Adafruit\_DHT 라이브러리를 사용하여 DHT 센서를 위한 라이브러리 임포트

import Adafruit\_DHT

# 이미지 및 텍스트 관련 작업을 위한 라이브러리 임포트

from PIL import ImageFont, ImageDraw, Image

# 현재 시간 및 날짜를 처리하기 위한 라이브러리 임포트

import datetime

# Picamera2 모듈을 사용하기 위한 라이브러리 임포트

from picamera2 import Picamera2

# 이미지 처리를 위한 라이브러리 임포트

import numpy as np

import cv2

# 시간 지연을 위한 sleep 함수 임포트

from time import sleep

# Flask 웹 애플리케이션 인스턴스 생성

app = Flask(\_\_name\_\_)

# GPIO 설정 및 초기화

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

# 물순환모터 및 물공급(수중)모터 및 LED를 제어하는 GPIO 핀 설정

GPIO.setup(5, GPIO.OUT) # 물순환모터

GPIO.setup(6, GPIO.OUT) # 물순환모터

GPIO.setup(13, GPIO.OUT) # 물공급(수중)모터

GPIO.setup(19, GPIO.OUT) # 물공급(수중)모터

GPIO.setup(26, GPIO.OUT) # LED

# 서보 모터를 제어하는 GPIO 핀 설정

servo\_pin = 21

GPIO.setup(servo\_pin, GPIO.OUT)

# 서보 모터 PWM 설정 및 시작

servo\_pwm = GPIO.PWM(servo\_pin, 50)

servo\_pwm.start(6.0)

# 물순환모터 및 물공급모터의 PWM 설정

motor\_pwm\_pin\_2 = 5 # 물순환모터 회전속도 조절

GPIO.setup(motor\_pwm\_pin\_2, GPIO.OUT)

motor\_pwm\_2 = GPIO.PWM(motor\_pwm\_pin\_2, 1000)

motor\_pwm\_pin\_3 = 13 # 물공급모터 회전속도 조절

GPIO.setup(motor\_pwm\_pin\_3, GPIO.OUT)

motor\_pwm\_3 = GPIO.PWM(motor\_pwm\_pin\_3, 1000)

def set\_motor\_speed\_2(speed):

motor\_pwm\_2.start(speed)

def set\_motor\_speed\_3(speed):

motor\_pwm\_3.start(speed)

# 모터 속도 제어 함수

# DHT11 온습도 센서 설정

DHT\_SENSOR = Adafruit\_DHT.DHT11

DHT\_PIN = 25 # DHT 센서가 연결된 GPIO 핀 번호

# 수온 센서 설정

temp\_sensor = '/sys/bus/w1/devices/28-fdd4451f64ff/w1\_slave'

# PCF8591 아날로그-디지털 컨버터 주소 및 변수 설정

address = 0x48

AIN2 = 0x42 # 수위센서

AIN3 = 0x43 # 탁도센서

AIN0 = 0x40 # 조도센서

# 초기 변수 설정

value = 0 # 수위

value1 = 0 # 수온

value2 = 0 # 조도

value3 = 0 # 탁도

value4 = 0 # 온습도

# DHT11 센서에서 습도 및 온도 읽기

humidity, temperature = Adafruit\_DHT.read\_retry(DHT\_SENSOR, DHT\_PIN)

# PCF8591 모듈 사용을 위한 smbus 주소 설정

bus = smbus.SMBus(1)

# I2C LCD 모듈 인스턴스 생성

mylcd = I2C\_LCD\_driver.lcd()

# 수온 센서 파일의 내용을 읽어오는 함수

def temp\_raw():

f = open(temp\_sensor, 'r')

lines = f.readlines()

f.close()

return lines

# 온습도 값을 읽어오는 함수

def read\_dht\_sensor():

global humidity, temperature

while True:

humidity, temperature = Adafruit\_DHT.read\_retry(DHT\_SENSOR, DHT\_PIN)

time.sleep(10) # 10초마다 온습도 값을 업데이트

# 온습도 값을 업데이트하는 스레드 생성 및 실행

dht\_thread = threading.Thread(target=read\_dht\_sensor)

dht\_thread.daemon = True # 메인 스레드가 종료되면 함께 종료

dht\_thread.start()

# 수온 센서 파일에서 온도를 읽어오는 함수

def read\_temp():

lines = temp\_raw()

while lines[0].strip()[-3:] != 'YES':

time.sleep(0.2)

lines = temp\_raw()

temp\_output = lines[1].find('t=')

if temp\_output != -1:

temp\_string = lines[1].strip()[temp\_output + 2:]

temp\_c = float(temp\_string) / 1000.0

return temp\_c # 수온 센서 읽어온 파일의 구문을 분석해 온도 부분을 반환하는 함수

# Picamera2 객체 생성

piCam = Picamera2()

# 미리보기 설정: 크기 640x480

piCam.preview\_configuration.main.size = (640, 480)

piCam.preview\_configuration.main.format = "RGB888"

piCam.preview\_configuration.align()

# 카메라 시작

piCam.start()

# 한글 폰트 설정

font = ImageFont.truetype('fonts/SCDream6.otf', 20)

# 영상 프레임을 생성하는 함수

def gen\_frames():

while True:

now = datetime.datetime.now()

nowDatetime = now.strftime('%Y-%m-%d %H:%M:%S')

frame = piCam.capture\_array() # 현재 프레임을 캡처

frame = Image.fromarray(frame)

draw = ImageDraw.Draw(frame)

draw.text(xy=(10, 15), text="스마트어항 " + nowDatetime, font=font, fill=(255, 255, 255))

frame = np.array(frame)

ref, buffer = cv2.imencode('.jpg', frame)

frame = buffer.tobytes()

yield (b'--frame\r\n'

b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n') # 그림파일들을 쌓아두고 호출을 기다림

# 홈 페이지

@app.route('/')

def index():

global value, value1, value2, value3, temperature, humidity, jodo, takdo, suwui

# 센서 데이터를 필요한 대로 읽고 수집합니다.

messagetosend = ''.join([str(x) for x in ['{0}mA\n\n{1}dark\n\n{2:.1f}°C\n\n{3}\n\n{4}°C, {5}%'.format(value, value1, value2, value3, temperature, humidity)]])

return messagetosend

# 자동 모드 실행 시 수행되는 함수

@app.route('/jadong', methods=['GET'])

def jadong():

global value, value1, value2, value3, temperature, humidity, jodo, takdo, suwui

# 어플에서 자동모드 실행 시

mylcd.clear() # LCD에 남는 잔상 정리

bus.write\_byte(address, AIN2)

bus.read\_byte(address)

lvwater = bus.read\_byte(address)

print("value: %d" % lvwater) # 수위센서 값 받아와서 화면에 출력

bus.write\_byte(address, AIN0)

bus.read\_byte(address)

llux = bus.read\_byte(address)

print(":{0}".format(llux)) # 조도센서 값 받아와서 화면에 출력

watertemp = read\_temp() # 수온센서 변수 지정

bus.write\_byte(address, AIN3)

bus.read\_byte(address)

pol = bus.read\_byte(address)

print(":{0}".format(pol)) # 탁도센서 값 받아와서 화면에 출력

if humidity is not None and temperature is not None:

print("온도={0:0.1f}°C 습도={1:0.1f}%".format(temperature, humidity)) # 온습도 센서 값 출력

# read\_dht\_sensor()

mylcd.lcd\_display\_string("{0:.1f}C / {1:.1f}mA".format(watertemp, lvwater), 1)

mylcd.lcd\_display\_string("{0}Da / NTU={1:.1f}".format(llux, pol), 2) # LCD에 수온, 수위, 조도, 탁도 센서값 출력

# 탁도, 조도, 수위센서값은 환경에 따라 조절해서 사용하세요.

# 탁도 범위에 따른 물순환 모터 동작

if pol < 60:

GPIO.output(5, 1)

set\_motor\_speed\_2(40)

elif pol > 80:

GPIO.output(5, 0)

motor\_pwm\_2.stop()

# 탁도센서 범위를 임의로 지정하여 물순환 모터 구동 함수

# 조도 범위에 따른 LED 동작

if llux > 228:

GPIO.output(26, 1)

elif llux < 224:

GPIO.output(26, 0)

# 조도센서 범위를 임의로 지정하여 LED 구동 함수

# 수위센서 범위에 따른 물순환 모터 동작

if lvwater < 10:

GPIO.output(13, 1)

set\_motor\_speed\_3(50)

elif lvwater > 50:

GPIO.output(13, 0)

motor\_pwm\_3.stop()

# 수위센서 범위를 임의로 지정하여 물순환 모터 구동 함수

return redirect('/')

# 모든 GPIO에 연결된 모터 및 LED 정지

@app.route('/stop', methods=['GET'])

def stop():

GPIO.output(5, 0)

GPIO.output(6, 0)

GPIO.output(13, 0)

GPIO.output(19, 0)

GPIO.output(26, 0)

motor\_pwm\_2.stop()

motor\_pwm\_3.stop()

# 자동 모드에서 나왔을 시 다시 자동모드가 작동 할 수 있게 초기화

global value, value1, value2, value3, temperature, humidity

value = 0

value1 = 0

value2 = 0

value3 = 0

temperature = 0

humidity = 0

return redirect('/')

# 실시간 센서 데이터 표시 페이지

@app.route('/re', methods=['GET'])

def re():

global value, value1, value2, value3, temperature, humidity

mylcd.clear() # LCD에 남는 잔상 정리

bus.write\_byte(address, AIN2)

bus.read\_byte(address)

value = bus.read\_byte(address)

print("value: %d" % value) # 수위센서 값 받아와서 화면에 출력

bus.write\_byte(address, AIN0)

bus.read\_byte(address)

value1 = bus.read\_byte(address)

print(":{0}".format(value1)) # 조도센서 값 받아와서 화면에 출력

value2 = read\_temp()

print(value2) # 수온센서 값 받아와서 화면에 출력

bus.write\_byte(address, AIN3)

bus.read\_byte(address)

value3 = abs((bus.read\_byte(address)))

print("value: {0}".format(value3)) # 탁도센서 값 받아와서 화면에 출력

if humidity is not None and temperature is not None:

print("온도={0:0.1f}°C 습도={1:0.1f}%".format(temperature, humidity)) # 온습도 센서 값 출력

mylcd.lcd\_display\_string("{0:.1f}C / {1:.1f}mA".format(value2, value), 1)

mylcd.lcd\_display\_string("{0}Da / NTU={1:.1f}".format(value1, value3), 2) # LCD에 수온, 수위, 조도, 탁도 센서값 출력

return redirect('/')

# LED를 켜는 함수

@app.route('/ledon', methods=['GET'])

def ledon():

GPIO.output(26, 1)

return redirect('/')

# LED를 끄는 함수

@app.route('/ledoff', methods=['GET'])

def ledoff():

GPIO.output(26, 0)

return redirect('/')

# 사료 공급 모터를 동작시키는 함수

@app.route('/saryoon', methods=['GET'])

def saryon():

servo\_pwm.ChangeDutyCycle(3.5)

time.sleep(1.0)

servo\_pwm.ChangeDutyCycle(6.0)

sleep(1)

return redirect('/')

# 물순환 모터를 동작시키는 함수

@app.route('/takon', methods=['GET'])

def takon():

GPIO.output(5, 1)

GPIO.output(6, 0) # 모터 동작

set\_motor\_speed\_2(40)

return redirect('/')

# 물순환 모터를 정지시키는 함수

@app.route('/takoff', methods=['GET'])

def takoff():

GPIO.output(5, 0)

GPIO.output(6, 0) # 모터 정지

motor\_pwm\_2.stop()

return redirect('/')

# 물공급 모터를 동작시키는 함수

@app.route('/wtron', methods=['GET'])

def wtron():

GPIO.output(13, 1)

GPIO.output(19, 0) # 모터 동작

set\_motor\_speed\_3(70)

return redirect('/')

# 물공급 모터를 정지시키는 함수

@app.route('/wtroff', methods=['GET'])

def wtroff():

GPIO.output(13, 0)

GPIO.output(19, 0) # 모터 정지

motor\_pwm\_3.stop()

return redirect('/')

# CCTV 영상 스트리밍 페이지

@app.route('/video')

def index2():

return render\_template('index4#2.html') # index4#2.html 템플릿을 렌더링하여 웹 페이지 반환

# CCTV 영상 스트리밍을 위한 URL

@app.route('/video\_feed')

def video\_feed():

return Response(gen\_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

# Flask 애플리케이션 실행

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host="0.0.0.0", port="8000")