컴퓨터전자공학과
A반 5조
201707530 신창섭

전국지역 온도출력 프로젝트

목적

프로젝트를 만들 때 저희의 첫 목적은 지역 이름을 입력 후 서버에 요청하고 온도정보를 받아서 지속적으로 세그먼트에 업데이트를 시켜주고 싶었지만, 서버에 요청할 수 있는 횟수가 한정적이라 일정 횟수를 초과하게 되면 금전적인 지출을 해야 하는 상황이 생겨서 지역 이름을 입력받으면 그 지역의 온도를 업데이트 없이 세그먼트에 한 번 출력하는 목표로 변경하였습니다.

활용도

프로젝트에 적용한 날씨 데이터를 제공해주는 웹서버에 들어가 보면 전 세계 모든 나라의 날씨 데이터를 제공하고 있습니다. 저희는 우리나라 날씨 데이터를 가져왔지만, 궁금한 나라의 온도를 알고 싶다면 언제든지 도시 이름값과 서버에서 가입하면 주는 key 값으로 서버에 요청하여 원하는 날씨 데이터를 얻을 수 있습니다. 이 프로젝트를 사회에 활용하게 된다면 거울 디스플레이(Smart Mirror)처럼 생활에서 주로 사용하는 거울이나, 유리, 창문, 인테리어 등 사물에 날씨 데이터를 결합하여 사용자에게 더 편리한 일상생활을 제공해줄 수 있습니다.

동작 전체 구성

1. 터미널에서 'python A_5.py'를 입력하여 프로그램을 실행합니다.

2. 터미널에 검색하고 싶은 지역 이름을 입력합니다.

3. Open Weather 서버에 요청하여 전국 지역 정보 데이터를 가져옵니다.

4. 최저온도, 현재 온도, 최고온도, 기압, 습도를 전국 지역별로 순서대로 정리하여 차트 형식으로 화면상에 출력합니다.

5. 4번에 출력하여 나온 차트의 다음 줄에 입력한 지역의 최저온도, 현재 온도, 최고온도를 순서대로 화면상에 출력합니다.

6. 4번에 나온 최저온도, 현재 온도, 최고온도를 세그먼트 두 개에 온도 양식에 맞추어 출력합니다.

7. 만약 입력한 지역의 온도를 계속 보기를 원하면 '계속 출력하시겠습니까(y/n)'가 출력이 되는데 이때 'y'를 입력하게 되면 계속 온도를 세그먼트에 출력하게 되며, 'n' 또는 'y'가 아닌 문자를 입력하게 되면 '다른 지역을 검색하시겠습니까? (y/n)'가 출력됩니다.

8. '다른 지역을 검색하시겠습니까? (y/n)'에서 'y'를 입력하게 되면 2번으로 돌아가서 자신이 검색하고 싶은 지역의 이름을 다시 검색할 수 있게 되고, 'n' 또는 'y'가 아닌 문자를 입력하게 되면 반복문을 빠져나오게 되어 프로그램이 정상적으로 종료하게 됩니다.

9. 위의 7번에서 프로그램을 종료하는 방법 이외에 또 다른 방법은 'Ctrl + C'를 입력하게 되면 "^C GoodBye"라는 문구가 화면상에 출력되면서 프로그램이 종료하게 되게끔 동작을 구성하였습니다.

동작 세부 설명

1. 터미널에서 파일이 있는 디렉터리로 이동 후 명령어를 입력하여 프로그램을 실행시킵니다.

```
pi@raspberrypi:~/report $ python A_5.py
지역 이름을 입력하십시오.
```

2. 검색하고 싶은 지역 이름을 입력합니다. (예 : 서울)

```
지역 이름을 입력하십시오.
서울
```

3. 서버에서 제공하는 날씨 데이터를 가져온 후 절대 온도의 값을 섭씨온도로 변환하고 전국 지역의 최저온도, 현재 온도, 최고온도, 기압, 습도를 순서대로 정리하여 차트 형식으로 화면상에 출력합니다.

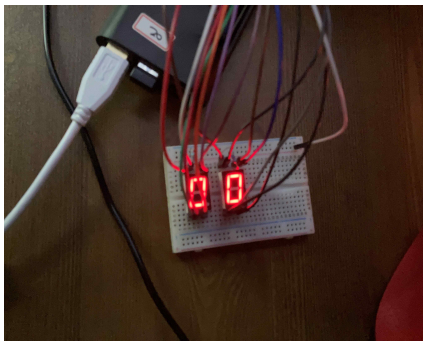
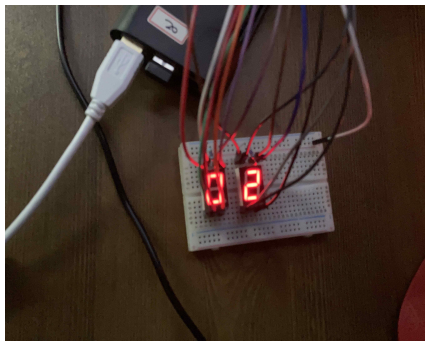
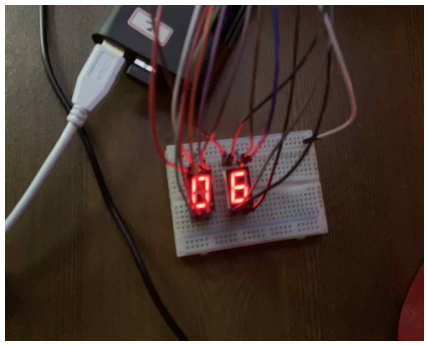
(저희는 서울~제주까지 대표적인 몇몇 지역을 따로 선택하여 데이터를 가져왔습니다.)

	city_id	city_name	temp_min	temp	temp_max	pressure	humidity
0	1835847	서울	-0.18	2.15	5.80	1027	48
1	1838524	부산	3.08	3.08	3.08	1025	71
2	1835329	대구	2.81	2.81	3.84	1026	80
3	1843564	인천	-0.10	4.65	5.88	1027	45
4	1841811	광주	1.76	1.76	1.76	1026	87
5	1835235	대전	3.67	3.67	3.67	1027	76
6	1833747	울산	8.62	8.62	8.62	1025	78
7	1835235	세종	3.67	3.67	3.67	1027	76
8	1841610	경기	-0.49	1.29	1.45	1027	62
9	1843125	강원	0.49	0.49	0.49	1028	63
10	1845106	충북	1.91	1.91	1.91	1027	73
11	1845105	충남	-0.70	1.31	1.31	1027	65
12	1845789	전북	-0.36	-0.36	-0.36	1027	90
13	1845788	전남	0.90	0.90	0.90	1026	92
14	1841597	경북	2.88	2.88	2.88	1026	87
15	1902028	경남	2.75	2.75	2.75	1026	84
16	1846266	제주	9.89	9.89	9.89	1026	67

4. 입력한 지역의 최저온도, 현재 온도, 최고온도의 수치를 반올림하여 문자열로 화면상에 출력합니다.

```
현재 입력한 '서울'지역의 최저온도는 영상 0도, 현재온도는 영상 2도, 최고 온도는 영상 6도입니다
```

5. 입력한 지역의 최저온도, 현재 온도, 최고온도를 세그먼트 두 개에 0.7초 간격으로 출력합니다.

<최저온도> <현재온도> <최고온도>

6. 처음 입력한 지역의 온도를 세그먼트에 계속 출력할 것인지 아니면 다른 지역의 온도를 검색할 것인지에 대해 각각의 입력을 받는 명령문을 출력합니다.

```
계속 출력하시겠습니까? (y/n)
```

```
다른 지역을 검색하시겠습니까? (y/n)
```

'y'를 입력하게 되면 세그먼트에 온도를 계속 출력, 'n'을 입력하게 되면 반복문을 나가게 되고, 위와 같은 사진처럼 출력이 됩니다. 'y'를 입력하게 되면 2번으로 가게 되어 지역 이름을 다시 입력할 수 있고, 'n'을 입력하게 되면 반복문을 빠져나가게 되어 정상적으로 종료가 됩니다.

```
pi@raspberrypi:~/report $ python A_5.py
지역 이름을 입력하십시오.
^CGoodBye
```

갑작스러운 종료에 대처하기 위해 'Ctrl + C'로 나가게 된다면 "GoodBye"라는 문구와 함께 프로그램을 정상적으로 종료시킬 수 있습니다.

설치 방법

라즈베리파이 터미널에서 각 명령어를 입력하여 설치합니다.

1. sudo apt-get update : 라즈베리 파이 시스템의 패키지 목록을 업데이트합니다.

2. sudo apt-get upgrade : 라즈베리 파이 설치된 패키지들을 최신으로 업그레이드합니다.

3. pip install requests : 서버에 요청하고 가져올 때 필요한 라이브러리를 설치합니다.

4. pip install pandas : 자료형의 있는 값을 깔끔하게 카테고리에 맞게 출력하게 하는 라이브러리를 설치합니다.

VCC +5, 한 개의 세그먼트마다 하나의 저항을 각각 연결한 후, 라즈베리파이에 GPIO 선 연결을 합니다.

첫 번째 세그먼트에는 (a : 17) (b : 27) (c : 22) (d : 10) (e : 9) (f : 11) (g : 13) 핀 번호로 연결합니다

두 번째 세그먼트에는 (a: 12) (b : 16) (c : 18) (d : 23) (e : 24) (f : 25) (g : 20) 핀 번호로 연결합니다.

프로그램을 실행시킬 때는 터미널에서 파일이 있는 디렉터리로 이동 후 python A_5(파일명).py를 입력합니다.

소스 설명

```
import pandas as pd # 리스트(배열)의 있는 값을 카테고리에 맞게 출력하게 하는 라이브러리를 가져옵니다.
import requests # weather 서버에 요청할 때 필요한 라이브러리를 가져옵니다.
import RPi.GPIO as GPIO # GPIO 모듈을 가져옵니다.
from time import sleep # sleep(delay) 딜레이 함수를 time 라이브러리에서 가져옵니다.

GPIO.setmode(GPIO.BCM) # GPIO(BCM) 핀 번호로 설정합니다.

def reset() : # 세그먼트 초기화 함수입니다. (세그먼트에 불이 모두 꺼진 상태)
    # 반복문을 이용하여 세그먼트1을 a ~ g까지 OFF 시킵니다.
    for segment in segments1 :
        GPIO.setup(segment, GPIO.OUT)
        GPIO.output(segment, 1)
    # 반복문을 이용하여 세그먼트2를 a ~ g까지 OFF 시킵니다.
    for segment in segments2 :
        GPIO.setup(segment, GPIO.OUT)
        GPIO.output(segment, 1)

def converte_kelvin_to_celsius(k): # 절대온도를 섭씨온도로 바꾸고 반환하는 함수입니다.
    return (k-273.15)

def Temperature1(temp) : # 섭씨온도를 영상 / 영하 문자열로 바꾸고 반환하는 함수입니다.
    if temp >= 0 :
        return "영상 " + str(temp) + "도"
    else :
        return "영하 " + str(temp + temp * -2) + "도"

def Temperature2(temp) : # 섭씨온도가 0도 이상, 10도 미만일 경우 문자열로 바꾸고 0을 더하고 반환하는 함수입니다.
    if temp >= 0 and temp < 10 :
        return '0' + str(temp)
    else : # 섭씨온도가 영하(-) 이거나 10도 이상일 경우 그대로 반환합니다.
        return str(temp)

#튜플과 일반 배열(리스트)의 차이점은 안에 있는 값을 변화시킬 수 있는가의 여부입니다. 세그먼트의 핀 번호는 변경되지 않아야 하므로 튜플 자료형으로 설정하였습니다.

# 첫 번째 세그먼트 연결된 핀 번호 : a부터 g까지 튜플 자료형으로 선언
segments1 = (17, 27, 22, 10, 9, 11, 13)

# 두 번째 세그먼트 연결된 핀 번호 : a부터 g까지 튜플 자료형으로 선언
segments2 = (12, 16, 18, 23, 24, 25, 20)
```

```
# 딕셔너리 자료형은 리스트나 튜플처럼 순차적으로 해당 요솟값을 구하지 않고 Key를 통해 Value(값)를 얻습니다.
# 즉 반복문을 통하여 순차적으로 접근할 필요가 없습니다. key 값만 알고 있으면 value 값을 바로 얻을 수 있습니다.
# 딕셔너리 자료형으로 Key(문자), Index(세그먼트 출력) 배열 선언
# (1 : 불이 꺼진상태, 0 : 불이 들어오는 상태, '-' : 영하온도 표시)
num = { '-' : (1,1,1,1,1,1,0), '0' : (0,0,0,0,0,0,1), '1' : (1,0,0,1,1,1,1), '2' : (0,0,1,0,0,1,0), '3' : (0,0,0,0,1,1,0),
        '4' : (1,0,0,1,1,0,0), '5' : (0,1,0,0,1,0,0), '6' : (0,1,0,0,0,0,0), '7' : (0,0,0,1,1,0,1), '8' : (0,0,0,0,0,0,0), '9' :
(0,0,0,0,1,0,0) }

prov_list = [ # API 서버에 요청할 때 필요한 지역 이름과, id를 가진 리스트(배열) 선언
    {'name':'서울','city_id':'1835847'},
    {'name':'부산','city_id':'1838524'},
    {'name':'대구','city_id':'1835329'},
    {'name':'인천','city_id':'1843564'},
    {'name':'광주','city_id':'1841811'},
    {'name':'대전','city_id':'1835235'},
    {'name':'울산','city_id':'1833747'},
    {'name':'세종','city_id':'1835235'},
    {'name':'경기','city_id':'1841610'},
    {'name':'강원','city_id':'1843125'},
    {'name':'충북','city_id':'1845106'},
    {'name':'충남','city_id':'1845105'},
    {'name':'전북','city_id':'1845789'},
    {'name':'전남','city_id':'1845788'},
    {'name':'경북','city_id':'1841597'},
    {'name':'경남','city_id':'1902028'},
    {'name':'제주','city_id':'1846266'},
]

weather_info_list = [] # weather 정보를 가지는 리스트(배열) 선언 및 초기화
url = 'http://api.openweathermap.org/data/2.5/weather' # weather 서버 API URL에 대입

try: # KeyboardInterrup exception 잡기 위한 try 문 (Ctrl + C)
    while True : # 계속 실행하는 반복문
        weather_info_list.clear() # 리스트(배열) 초기화 함수 호출
        reset() # 세그먼트 초기화 함수 호출
        print("지역 이름을 입력하십시오.")
        inputname = input() # 찾고 싶은 지역의 이름을 입력받아 inputname 변수에 대입
        for i in range(len(prov_list)): # prov_list의 크기만큼 증가하는 for문
            city_id = prov_list[i]['city_id'] # prov_list 리스트의 i번째 'city_id'(key) 안에 있는 값을 city_id에 대입
            city_name = prov_list[i]['name'] # prov_list 리스트의 i번째 'name'(key) 안에 있는 값을 city_name에 대입
            params = dict( # 도시 id와 자신의 key를 딕셔너리 자료형(사전)으로 변환 후 params에 대입
                id=city_id, # 도시 id의 정보를 id에 대입
                APPID='1cbebe0837da91d9ce1a6a16c747d8da', # 자신의 Key의 값을 APPID에 대입
            )
            resp = requests.get(url=url, params=params) # url과 params를 인자로 갖고 (request) 요청하여 서버에
.json파일을 (get) 가져옵니다.
            data = resp.json() # json()함수를 이용하여 .json파일을 읽고 구문 분석한 값의 리스트를 data에 대입
            data_main = data['main'] # data 리스트의 key 값이 main인 데이터를 data_main에 대입
```

```
    if inputname == prov_list[i]['name'] : # 입력한 도시 이름이 맞는지 판별하는 if 문
        info = [ # 입력한 도시 이름의 데이터를 info 배열에 저장
            city_id,
            city_name,
            converte_kelvin_to_celsius(data_main['temp_min']), \
            # data_main 리스트의 key 값이 temp_min인 데이터를 섭씨온도로 변환
            converte_kelvin_to_celsius(data_main['temp']), \
            # data_main 리스트의 key 값이 temp인 데이터를 섭씨온도로 변환
            converte_kelvin_to_celsius(data_main['temp_max']), \
            # data_main 리스트의 key 값이 temp_max인 데이터를 섭씨온도로 변환
            data_main['pressure'], \
            # data_main 리스트의 key 값이 pressure인 기압 데이터
            data_main['humidity']]
            # data_main 리스트의 key 값이 humidity인 습도 데이터
        weather_info_list.append(info) # weather_info 리스트(배열)에 info를 추가(append)

        # 입력한 도시 날씨 데이터를 따로 변수에 저장
        outputcity = city_name
        # 도시 이름을 따로 outputcity 변수에 대입
        outputtempmin = round(converte_kelvin_to_celsius(data_main['temp_min']))
        # 최저온도 : 절대 온도 값을 섭씨온도로 변환 후 반올림하여 outputtempmin 변수에 대입
        outputtemp = round(converte_kelvin_to_celsius(data_main['temp']))
        # 현재 온도 : 절대온도 값을 섭씨온도로 변환 후 반올림하여 outputtemp 변수에 대입
        outputtempmax = round(converte_kelvin_to_celsius(data_main['temp_max']))
        # 최고온도 : 절대온도 값을 섭씨온도로 변환 후 반올림하여 outputtempmax 변수에 대입
        continue
        # 밑에는 실행되지 않고 다음 반복문으로 넘어가는 continue

    info = [ # 다른 도시 날씨 데이터를 info에 저장
        city_id, \
        city_name, \
        converte_kelvin_to_celsius(data_main['temp_min']), \
        converte_kelvin_to_celsius(data_main['temp']), \
        converte_kelvin_to_celsius(data_main['temp_max']), \
        data_main['pressure'], \
        data_main['humidity']]
    weather_info_list.append(info) # weather_info 리스트(배열)에 info를 추가(append)

# 출력문
# pandas 라이브러리를 이용하여 weather_info_list의 정렬 배치 후 데이터 출력
df = pd.DataFrame(weather_info_list, columns=['city_id', 'city_name', \
                                              'temp_min', 'temp', 'temp_max',\
                                              'pressure', 'humidity'])
print(df)

print(" ") # 한 칸 띄어쓰기
print("현재 입력한 " + outputcity + "지역의 최저온도는 " + Temperature1(outputtempmin) + ", 현재온도는 " +
          Temperature1(outputtemp) + ", 최고 온도는 " + Temperature1(outputtempmax) + "입니다.\n")
print(" ") # 한 칸 띄어쓰기
```

```
        length = len(segments1) # 한 세그먼트당 핀 개수 길이를 length 변수에 저장
        array = [Temperature2(outputtempmin), Temperature2(outputtemp), Temperature2(outputtempmax)] # 최저, 현재, 최고온도를 Temperature2함수를 이용하여 문자열로 반환 후 배열로 선언
        while True : # 반복문
            for i in range(len(array)) : # 최저온도, 현재 온도, 최고온도를 순서대로 반복하는 반복문
                for j in range(length) : # 세그먼트의 a부터 g까지 반복하는 반복문
                    GPIO.output(segments1[j], num[array[i][0]][j]) # 2개의 문자를 가진 문자열을 반복하여 첫 번째 세그먼트에 하나씩 출력 (0번지)
                    GPIO.output(segments2[j], num[array[i][1]][j]) # 2개의 문자를 가진 문자열을 반복하여 세그먼트에 하나씩 출력 (1번지)
                sleep(0.7) # 딜레이 0.7초간 프로세스 중지

            print("계속 출력하시겠습니까? (y/n)")
            hello1 = input() # y를 입력받으면 계속 출력 할 수 있습니다.
            if hello1 != 'y': # 입력한 값이 y가 아닐 경우
                break # 반복문을 빠져나옵니다.

        print("다른 지역을 검색하시겠습니까? (y/n)")
        hello2 = input() # y를 입력받으면 계속 검색 할 수 있습니다.
        if hello2 != 'y': # 입력한 값이 y가 아닐 경우
            GPIO.cleanup() # GPIO가 해제되면서 세그먼트가 초기화가 되어 불빛이 모두 꺼집니다.
            break # 반복문을 빠져나옵니다.

except KeyboardInterrupt: # Ctrl + C를 입력하면 프로그램 중단(exception 방지)
    print("GoodBye")
    GPIO.cleanup() # GPIO가 해제되면서 세그먼트가 초기화가 되어 불빛이 모두 꺼집니다.
```

다른자료 참조 및 해결과정

이번 프로젝트를 진행하면서 처음에는 C언어로 웹 API를 가져오는 방법을 찾다가 어려움에 부딪혀 웹 검색을 하여 여러 방법을 찾는 와중에 파이썬 언어로, 서버에 요청하여 정보를 받을 수 있는 라이브러리가 있고, 예제들이 있으며, 쉽고 효율적이라고 판단하게 되어 파이썬 언어로 시작하게 되었습니다. 파이썬 언어는 처음이었지만 웹 검색을 통해 파이썬 언어에 대해 공부를 하고 프로젝트를 만들게 되었습니다.

프로젝트 진행 과정에 있어서 파이썬 언어를 이용하여 웹에서 날씨 정보를 가져오기 위해 웹에 검색하여 참조하였습니다. 참조한 사이트의 주소는 https://m.blog.naver.com/wideeyed/221335980042 이며, 파이썬 언어를 공부하고 하나씩 주석처리 해나가면서 해석하였습니다.

참조한 사이트의 코드를 해석하는 과정에서 자료형의 종류, 함수의 종류가 많아서 프로젝트 기간에 파이썬 언어를 공부했었던 사이트의 주소입니다. https://wikidocs.net/14 "점프 투 파이썬"

라즈베리파이에서 파이썬 언어를 이용하여 GPIO 연결을 하기 위해 웹에 검색하여 라이브러리와 사용 방법이 적힌 사이트를 참고하였습니다. https://junolefou.tistory.com/5

감사합니다!