글. 고경희 tbs 교통방송 기술국 기술감독

GPI controller (Auto scheduler) 개발 - 2

Flask와 Pandas 라이브러리를 이용한 프로그램

지난 10월호에 이어서 편의를 위해 추가한 web application part에 대해 자세한 설명을 하겠다.

Python(파이썬)만으로 구성된 스케줄러는 콘솔 창에서 동작하기 때문에 사용자 접근성이 낮다. 그래픽 사용자 인터페이스(GUI)가 필요한데 어디서나 접근이 가능하고 유지보수가 용이한 점 때문에 웹 인터페이스를 선택하게 되었다. 파이썬 기반의 Web Framework는 Django(장고), Flask(플라스크), Pyramid(피라미드), Tornado(토네이도), Bottle(보틀) 등 이외에도 수많은 Framework들이 있었지만 초보자에게는 문서화가 잘 되어있는지. 배우기가 쉬운지가 제일 중요한 선택요소였다. 우리는 파이썬과의 연계, 확장이 가능한 파이썬 기반의 Framework인 플라스크로 개발을 진행하였다.

플라스크는 장고와 함께 파이썬 웹 개발을 대표하는 프로젝트 중 하나이다. 최소한의 기능만을 제공하여 유연하게 애플리케이션 작성이 가능하며 구인/구직 사이트로 유명한 Linkedin(링크드인)이나 이미지 포털인 Pinterest(핀터레스트) 등이 플라스크를 이용해서 개발되었다. 좀 더 대중적인 장고를 사용하지 않은 이유는 작성해야 하는 코드가 더 적고 규모가 작은 애플리케이션을 만드는데 적합하다고 생각했기 때문이다.

웹에서 구현될 기능은 크게 세 가지로 구분했다.

1. 파일 업로드 (Flask) / 2. 파일 확인 (Pandas) / 3. 스케줄러 실행 및 정지 (Subprocess)

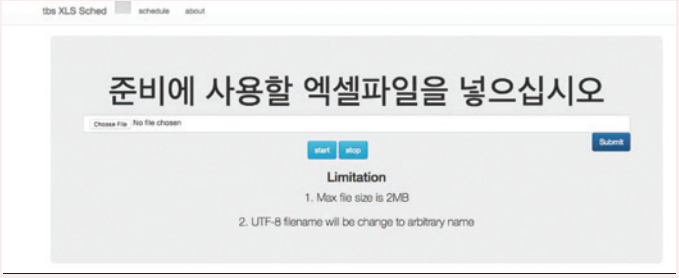

그림 1. Web Application 화면

파일 업로드

파일 업로드에 대한 부분은 일반적인 HTML 업로드 방식과 같지만 파일에 대한 검증(Validation)을 위해서 Flask-WTF 모듈을 사용했다. 적용된 제한사항으로는 파일 확장자는 xlsx, xls이며 2Mb 이하 용량의 파일만 업로드하여 엉뚱한 파일의 입력으로 인해 스케줄러가 멈출 상황을 차단하였다. 시안성을 좋게 만들기 위해 Bootstrap(부트스트랩)을 이용하여 가독성 및 시각적 친숙도를 높였다.

APC에서 추출한 xls 파일은 바로 읽을 경우엔 손상된 파일이라는 메시지가 떴다. 인코딩 방식이 맞지 않는 오래된 버전의 엑셀 파일이므로 raw 파일 형태로 열어 각 열별로 다시 저장한 후, xls로 다시 저장했다. 이 부분에 대한 코드는 다음과 같다.

```
import xlrd
import xlwt
from xlwt import Workbook
import io

filename  = 'FM180406.xls'
file = io.open(filename, 'r', encoding = 'euc-kr')
data = file.readlines()

xldoc = Workbook()
sheet = xldoc.add_sheet("Sheet1", cell_overwrite_ok=True)

for i, row in enumerate(data):
    # filtering
    for j, val in enumerate(row.replace('₩n', '').split('₩t')):
        sheet.write(i, j, val)

xldoc.save('myexcel.xls')
```

파일 확인 (Pandas)

불러온 엑셀 파일은 시간 정보와 실행 장소, 제목, 출연자 등의 많은 정보를 담고 있다. Pandas는 이 중 필요한 정보만으로 필터링하기 위해 사용하였다. Pandas는 파이썬에서 사용하는 데이터구조 라이브러리로 행과 열로 이루어진 데이터를 분석, 가공하기 위해 사용한다. 이 정도의 작은 리스트를 다룰 수 있는 라이브러리는 Pandas 외에도 많지만 최근 데이터 사이언스 등에 많이 쓰이고 있으며 직관적으로 데이터를 다룰 수 있는 점이 장점이다.

다음은 기존의 시간추출 코드와 pandas를 활용해 단순화시킨 코드의 비교이다.

기존

```
wb = load_workbook('/home/pi/Documents/{}.xlsx'.
format(filename))
sheet = wb.get_sheet_by_name('{}'.format(filename))
d = []
t = []
c = []

for i in range(1, 127):
      cell_D = "{}{}".format("D", i) #sheet['Di']
      d.append(sheet[cell_D].value)
      if sheet[cell_D].value!='생':
         TimeData = "{}{}".format("A", i)
         t.append(sheet[TimeData].value)
         cell_C = "{}{}".format("C", i)
         c.append(sheet[cell_C].value)

for j in range(len(t)):
   t[j] = str(t[j]).split(":")
```

Pandas

```
From pandas as pd

xls = 'myexcel.xls'
data = pd.read_excel(xls, header=None)

def time(i):
   time = data.iloc[i, 0].replace(' '', ' ')
   hours, minutes, seconds = time.split(":")
   return hours, minutes, seconds
```

스케줄에 필요한 정보는 시간이므로 시간데이터만을 추출하여 스케줄에 활용하고 또한 웹에 표현될 내용을 따로 수정하여 스케줄을 확인할 수 있게 해준다.

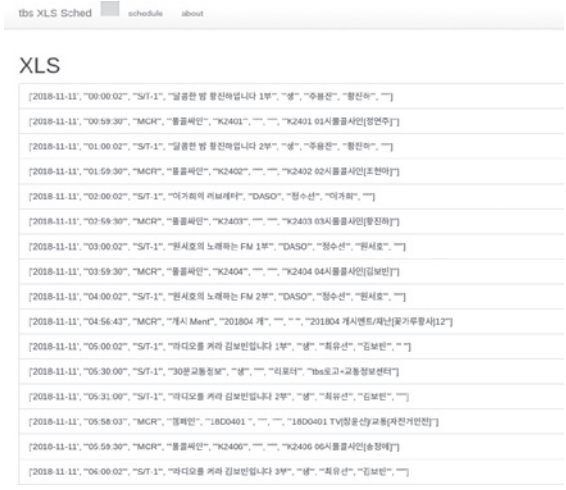

그림 2. 프로그램 실행 화면 (방송 정보 추출 목록)

스케줄러 실행 및 정지

스케줄러의 동작은 쉘 스크립트에서 직접 py 파일을 실행시키는 코드를 작성했다. subprocess는 파이썬 내에서 새로운 프로세스를 생성하고 여기에 입출력 파이프를 연결하며 코드를 획득할 수 있도록 하는 모듈로, 다른 언어로 만들어진 프로그램을 통합, 제어할 수 있도록 만드는 모듈이다.

subprocess로 GPIO 스케줄러 동작과 관련된 py 파일을 실행할 수 있도록 app.route를 설정하여 웹 인터페이스에서 웹 링크 형태로 연결되도록 했다. 코드는 다음과 같다.

```
@app.route('/start')
def start():
   subprocess.call ('gpi_controller.py', shell = True)
   return 'GPIO App is started'

@app.route('/stop')
def stop():
   subprocess.Popen("^D")
   return 'GPIO App stopped'
```

이번 호에서는 지난 10월호에 소개한 GPI controller 개발과정에 이어 접근성을 수월하게 하여 프로그램을 실행하고 모니터링 하는 web application 모듈 알고리즘에 대하여 살펴보았다. 앞으로 라디오 방송 업무에서 활용하여 자동으로 스케줄에 맞춰 녹음방송이 송출될 수 있도록 안정성을 좀 더 확보하고, 사용자의 편의와 요구에 맞게 프로그램을 수정 및 보완할 계획이다.