


3차년도 주요 결과물

(과제명) 대규모 분산 에너지 저장장치 인프라의 안전한 자율운영
및 성능 평가를 위한 지능형 SW 프레임워크 개발
(과제번호) 2021-0-00077

- 결과물명 : ESS 데이터 수집 환경 별 지표화 처리(SW)
- 작성일자 : 2023년 11월 14일

과학기술정보통신부 SW컴퓨팅산업원천기술개발사업
“3차년도 주요 결과물” 로 제출합니다.

수행기관	성명/직위	확인
충남대학교	김종훈/교수	

정보통신기획평가원장 귀하

사 용 권 한

본 문서에 대한 서명은 한국전자기술연구원 내부에서 본 문서에 대하여
수행 및 유지관리의 책임이 있음을 인정하는 것임.

본 문서는 작성, 검토, 승인하여 승인된 원본을 보관한다.

작성자 :	이미영	일자 :	2023. 11. 13
-------	-----	------	--------------

검토자 :	한동호	일자 :	2023. 11. 14
-------	-----	------	--------------

승인자 :	김중훈	일자 :	2023. 11. 14
-------	-----	------	--------------

제 · 개정 이력

버전	변경일자	제.개정 내용	작성자
1.0	2023-11-13	최초 등록	이미영

목 차

1. 개요	-----1
2. ESS 데이터 수집 환경 별 지표화 처리 모듈	-----2

1. 개요

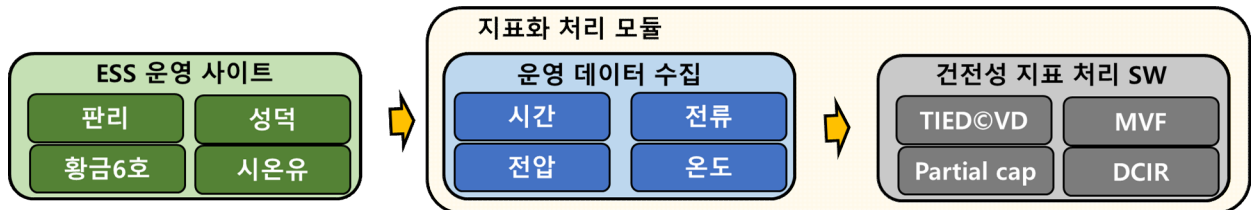
□ 목적

- 본 명세서의 목적은 대규모 분산에너지 저장장치 인프라의 지능형 안전SW 프레임워크에 적용하기 위한 전처리 모듈로서 배터리 내부 상태를 대변할 수 있는 지표를 추출하고자 함.

□ 범위

- 연구실 보유 노화 데이터 기반 배터리 수명과 연관성이 높은 건전성 지표를 실제 운영 ESS를 기반으로 추출하는 코드를 설명하고자 함.

□ 시스템 개요



운영 ESS 데이터 기반 데이터 취득 및 전처리 모듈 흐름도

2. ESS 배터리 데이터 수집 환경 특징

□ ESS 수집 환경 특징

- ESS는 신재생 연계형, 주파수 조정용, 부하 저감용 등 다양한 용도로 사용되며, 외부에 설치되어 날씨 등에 영향을 받을 가능성이 다분함.
- 일반적으로 ESS는 충전 및 방전을 정적으로 수행하기 때문에 운영 데이터 확보 시 정적인 조건 하에서 건전성 지표를 도출 할 수 있음.
- 하루단위 운영 데이터를 다운로드하여 파이썬 기반 건전성 지표 추출 모듈을 설계하여 유의미한 정보를 추출하고자 함.

3. ESS 수집 환경 별 지표화 처리 모듈

- o 운영 사이트 별 ESS에서 취득 가능한 데이터를 수집하는 코드를 아래 표에 나타내었음.

일일 운영 데이터 기반 건전성 지표 추출 모듈

```
import csv
import pandas as pd
import psycopg2
from sqlalchemy import create_engine
from sqlalchemy.sql.expression import false
import numpy as np
import os
import google_storage_class
import matplotlib.pyplot as plt
import matplotlib

for k in range(1, 2):
    if __name__ == "__main__":

        local_file_path = "./"

        _PROJECT_NAME = "KETI-IISRC"
        _BUCKET_NAME = "ess-bucket-1"
        _CREDENTIAL_JSON_FILE_PATH = "keti-iisrc-272da94e55c9.json"

        GCP = google_storage_class.google_cloud_storage(
            _PROJECT_NAME, _BUCKET_NAME, _CREDENTIAL_JSON_FILE_PATH
        )

        result1, result2 = GCP.list_blobs_with_prefix(
            prefix="2022/04/0"+str(k),
            delimiter="csv",
        )

        path_list = list(result2)
        path_list.sort()
```

```

for gcp_file_path in path_list:
    file_name = gcp_file_path.split("/")[-1]
    GCP.download_blob(gcp_file_path, local_file_path + file_name)

file_name1 = "2022040" #다운로드 하고자 하는 날짜
file_name2 = str(k)
file_name3 = "_rack.csv" #rack data 의미
file_name_final = file_name1 + file_name2 + file_name3
print(file_name_final)
ess_data = np.loadtxt(file_name_final, delimiter = ',', skiprows = 1,
usecols=[3,5,6,7,8,9,10,18]) #다운로드 하고자 하는 날짜의 ESS rack data를 불러옴.
n = []
for i in range(len(ess_data)):
    if ess_data[i, 0] == 1:
        n.append(i)
data = ess_data[n, 1:]
soc = data[:,0] #다운로드 받은 데이터의 SOC 정보
voltage = data[:,1] #다운로드 받은 데이터의 전압 정보
current = data[:,2] #다운로드 받은 데이터의 전류 정보
maxvoltage = data[:,3] #다운로드 받은 데이터의 최대 전압 정보
maxnumber = data[:,4] #다운로드 받은 데이터의 최대 전압 셀 번호 정보
minvoltage = data[:,5] #다운로드 받은 데이터의 최소 전압 정보
minnumber = data[:,6] #다운로드 받은 데이터의 최소 전압 셀 번호 정보
temperature = data[:, 7] #다운로드 받은 데이터의 온도 정보
print(len(data))

# 원하는열
# 건전성지표(tiecvd/tiedvd, 등충전/방전 전압 도달 시간, 충전/방전 구간에서 최대/
최소 전압 도달에 걸리는 시간)
# 추후 전압 구간 제한 예정 (예 700-800)
ch = [] #충전 구간 전압 저장 공간 나누기
dis = [] #방전 구간 전압 저장 공간 나누기
chtem = [] #충전 구간 온도 저장 공간
distem = [] #방전 구간 온도 저장 공간

for i in range(1, len(data)):
    if current[i] > 2: #충전

```

```

ch.append(voltage[i]) #충전 전압
chtem.append(temperature[i])
elif current[i] < -2: #방전
    dis.append(voltage[i]) #방전 전압
    distem.append(temperature[i])
chmax = ch.index(max(ch)) #충전 중 최대 전압 지점
#print(chmax)
chmin = ch.index(min(ch)) #충전 중 최소 전압 지점
tiecvd = abs(chmax-chmin) #충전 중 최대 전압-최소전압 도달 시 시간
#print(tiecvd)
tiedvd = abs(dis.index(max(dis))-dis.index(min(dis)))
#방전중 최소/최대 전압 도달 시 시간
#print(tiedvd)
TIEVD = [tiecvd, tiedvd]
print(TIEVD)

#MVF(Mean voltage falloff)
mvf = []
for i in range(40000, 50000, 200): #일정 시간 범위 설정(선형적인 구간), 간격
    설정
    mvf.append(round(max(dis)) - dis[i]) #rated 전압(여기서는 max 전압) 에서 일
    정 시간 간격 후 전압을 빼값
    MVF = [sum(mvf)/100, 1] #100개의 전압 구간을 최대 전압에서 빼 값 합산
    및 평균 (1은 배열 맞춰주기 위함)
    print(MVF, 1)
#VIEDTD
for i in range(1,len(soc)) :
    if soc[i] == 10 and current[i] > 0 :
        #SOC 10% 및 충전 구간일때 등시간간격 내 전압량
        viect1 = abs(data[i, 1] - data[1+1000, 1]) #1000s기준으로 진행
    elif soc[i] == 20 and current[i] > 0 :
        viect2 = abs(data[i,1] - data[i+1000, 1])
    elif soc[i] == 30 and current[i] > 0 :
        viect3 = abs(data[i, 1]- data[i+1000, 1])
    elif soc[i] == 40 and current[i] > 0 :
        viect4 = abs(data[i, 1]- data[i+1000, 1])

```



```

elif soc[i] == 50 and current[i] > 0 :
    viect5 = abs(data[i, 1]- data[i+1000, 1])
elif soc[i] == 60 and current[i] > 0 :
    viect6 = abs(data[i, 1]- data[i+1000, 1])
elif soc[i] == 70 and current[i] > 0 :
    viect7 = abs(data[i, 1]- data[i+1000, 1])
elif soc[i] == 80 and current[i] > 0 :
    viect8 = abs(data[i, 1]- data[i+1000, 1])
if soc[i] == 10 and current[i] < 0 :
#SOC 10% 및 방전 구간일때 등시간간격 내 전압량
    viedt1 = abs(data[i, 1] - data[i+1000, 1])
elif soc[i] == 20 and current[i] < 0 :
    viedt2 = abs(data[i,1] - data[i+1000, 1])
elif soc[i] == 30 and current[i] < 0 :
    viedt3 = abs(data[i, 1]- data[i+1000, 1])
elif soc[i] == 40 and current[i] < 0 :
    viedt4 = abs(data[i, 1]- data[i+1000, 1])
elif soc[i] == 50 and current[i] < 0 :
    viedt5 = abs(data[i, 1]- data[i+1000, 1])
elif soc[i] == 60 and current[i] < 0 :
    viedt6 = abs(data[i, 1]- data[i+1000, 1])
elif soc[i] == 70 and current[i] < 0 :
    viedt7 = abs(data[i, 1]- data[i+1000, 1])
elif soc[i] == 80 and current[i] < 0 :
    viedt8 = abs(data[i, 1]- data[i+1000, 1])
viect = np.array([viect1, viect2, viect3, viect4, viect5, viect6, viect7, viect8])
viedt = np.array([viedt1, viedt2, viedt3, viedt4, viedt5, viedt6, viedt7, viedt8])
VIET = np.stack([viect, viedt], 1)
print(VIET)
#DCIR
cs = []
for i in range(1, len(current)-1):
    if abs(voltage[i]-voltage[i+1]) > 3 and current[i+1] > 0:#충전 구간 전압 변화
    량 3이상
        cs.append(i+1)
#print(cs)

```

```

dcir = abs((data[cs[0]-1, 1]-data[cs[0], 1])/data[cs[0], 2])
#DCIR (전류 인가 후 전압 변화량 의미)
DCIR = [dcir, 1]

#DTV  $T(i)-T(i-1)/V(i)-V(i-1)$ 
dtv = []
disvol = []
#voldiff = []
for i in range(1, round(len(dis)/10) - 1):
    if dis[10*(i-1)]-dis[10*i] != 0:
        dtv.append((distem[10*(i-1)] - distem[10*i+1])/(dis[10*(i-1)]-dis[10*i]))
        disvol.append(dis[10*i])
        #voldiff.append(dis[10*(i-1)]-dis[10*i])
    #plt.plot(disvol, dtv)
#voldiff.append(dis[10*(i-1)]-dis[10*i])
DTV = [disvol, dtv]
#이거는 PARTIAL CAPACITY 저장
out_name1 = "2022040"
out_name2 = str(k)
out_name3 = "_partialcap.csv"
out_name4 = "_partialcap_sum.csv"
out_name5 = "_TIEVD" #건전성 지표1
out_name6 = "_MVF" #건전성 지표2
out_name7 = "_VIET" #건전성 지표3
out_name8 = "_DCIR" #건전성 지표4
out_name9 = "_DTV" #건전성 지표5
out_name_final1 = out_name1 + out_name2 + out_name3
out_name_final2 = out_name1 + out_name2 + out_name4
out_name_final3 = out_name1 + out_name2 + out_name5
out_name_final4 = out_name1 + out_name2 + out_name6
out_name_final5 = out_name1 + out_name2 + out_name7
out_name_final6 = out_name1 + out_name2 + out_name8
out_name_final7 = out_name1 + out_name2 + out_name9
#np.savetxt(out_name_final1,partialcap,delimiter=",")
#np.savetxt(out_name_final2,partialcap_sum,delimiter=",")
np.savetxt(out_name_final3, TIEVD, delimiter=",") #건전성 지표1 저장

```

```

np.savetxt(out_name_final4, MVF, delimiter=",") #건전성 지표2 저장
np.savetxt(out_name_final5, VIET, delimiter=",") #건전성 지표3 저장
np.savetxt(out_name_final6, DCIR, delimiter=",") #건전성 지표4 저장
np.savetxt(out_name_final7, DTV, delimiter=",") #건전성 지표5 저장
#plt.figure(figsize=(14, 5))
#plt.plot(dis)
#plt.plot(current)
#plt.show()
plt.plot(soc)
plt.plot(current)
plt.show()

```