DAFTAR PUSTAKA

A. G. Bakirtzis, O. Erdinc, N. G. Paterakis, I. N. Pappi, J. P. S. Catalão, “a new perspective for sizing of distributed generation and energy storage for smart households under demand response,” Applied Energy, vol. 143, pp. 26-37, 2015.

BMW model i3 specifications <https://www.bmw.com>

C. Haksever, J. Moussourakis, “A model for optimizing multi-product inventory systems with multiple constraints”, International Journal of Production Economics, vol. 97, pp. 18-30, 2005.

C. Liu, K. T. Chau, D. Wu, S. Gao, "Opportunities and Challenges of Vehicle-to-Home, Vehicle to Vehicle, and Vehicle to Grid Technologies," Proceedings of the IEEE, vol. 101, no. 11, pp. 2409-2427, 2013.

C. P. Vineetha, C. A. Babu, “Smart grid Challenges, Issues and Solutions,” IGBSG 2014 - International Conference on Intelligent Green Building and Smart Grid, 2014.

Craig Smith, Kelly “Parmenter. Energy Management Principles 2nd Edition”. 2016

Chenxi Li, Fengji Luo, Yingying Chen, Zhao Xu, Yinan An, Xiao Li. “Smart home energy management with vehicle-to-home technology”, 13th IEEE International Conference on Control & Automation (ICCA), 2017.

Chris Mi, M. Abul Masrur, “Hybrid Electric Vehicles: Principles and Applications with Practical Perspectives”. 2018

Dewan Energi Nasional, *“Laporan Neraca Energi Nasional”,* 2019.

D. Thomas, O. Deblecker, C. S. Ioakimidis. “Optimal operation of an energy management system for a grid- connected smart building considering photovoltaics” uncertainty and stochastic electric vehicles' driving schedule", Applied Energy, vol. 210, pp. 1188-1206, 2018.

E. Crisostomi, R. Shorten, S.Stüdli, F. Wirth, "Electric and Plug-in Hybrid Vehicle Networks: Optimization and Control," 2018.

F. Y. Melhem, O. Grunder, Z. Hammoudan, N. Moubayed, "Optimization and Energy Management in Smart Home Considering Photovoltaic, Wind, and Battery Storage System With Integration of Electric Vehicles," Canadian Journal of Electrical and Computer Engineering, vol. 40, no. 2, pp. 128-138, 2017.

Gowrishankar S. Veena A. , "Introduction to Python Programming", 2019.

Global Atlas <https://globalsolaratlas.info>

Gurobi Solver <https://www.gurobi.com>

H. S. V. S. Kumar Nunna, Swathi Battula, Suryanarayana Doolla, Dipti Srinivasan, "Energy Management in Smart Distribution Systems with Vehicle-to-Grid Integrated Microgrids", IEEE Transactions on Smart Grid, vol. 9, pp. 4004 - 4016, 2018.

J. Axsen, A. Burke, and K. Kurani, "Batteries for plug-in hybrid electric vehicles (PHEVs): Goals and the state of technology circa 2008". Report UCD-ITS-RR-08-14, Institute of Transportation Studies, University of California, Davis, CA, USA, 2008.

J. Lunden, S. Warner, V. Koivunen, "Distributed demand-side optimization with load uncertainty," 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 5229-5232, 2013, Jond v. guttag, “Introduction to computation and programming using python”, 2013,

John K. Karlof, “Integer Programming: Theory and Practice”, 2006.

K. C. Sou, J. Weimer, H. Sandberg, K. H. Johansson. “Scheduling Smart Home Appliances Using Mixed Integer Linear Programming”, 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC). 2011

Kemenetrian Energi Dan Sumber Daya Mineral, *“Handbook of Energy & Economic Statistics of Indonesia”,* 2019.

M. Beaudin , H. Zareipour, “Home energy management systems: A review of modelling and complexity,” Renewable and Sustainable Energy Reviews, vol. 45, pp. 318-355, 2014.

M. Elkazaz , M. Sumner, R. Davies, S. Pholboon, D. Thomas, “Optimization based Real-Time Home Energy Management in the Presence of Renewable Energy and Battery Energy Storage,” 2019 International Conference on Smart Energy Systems and Technologies (SEST), 2019.

M. E. V. Segatto, H. R. O. Rocha, J. A. L. Silva, M. H. M. Paiva, M. A. R. S. Cruz, “Telecommunication Technologies for Smart Grids: Total Cost Optimization,” Biomass, Fuel Cells, Geothermal Energies, and Smart Grids, vol. 2, pp. 451-478, 2018.

M. L. Lee, O. Aslam, B. Foster, D. Kathan, C. Young, “Assessment of demand response and advanced metering,”. Staff report. Federal Energy Regulatory Commission, 2014.

O. Erdinç, N. G. Paterakis, I. N. Pappi, A. G. Bakirtzis, J. P. S. Catalão, “Coordinated Operation of a Neighborhood of Smart Households Comprising Electric Vehicles, Energy Storage and Distributed Generation," EEE Transactions on Smart Grid, vol. 7, no. 6, pp. 2736-2747, 2016.

O. Erdinç, N. G. Paterakis, T. D. P. Mendes, A. G. Bakirtzis, J. P. S. Catalão, "Smart Household Operation Considering Bi-Directional EV and ESS Utilization by Real-Time Pricing-Based DR," IEEE Transactions On Smart Grid, vol. 6, pp. 1281-1291, 2014.

Peraturan Presiden Republik Indonesia nomor 22, 2017.

Peraturan Presiden Republik Indonesia nomor 55, 2019.

Peraturan Menteri ESDM-RI nomor 49, 2018.

Python <https://www.python.org>

Sándor F. Tóth, "Basic Linear Programming Concepts", 2012.

Spyder <https://www.spyder-ide.org>

T. Kobashi, M. Yarime. "Techno-economic assessment of the residential photovoltaic systems integrated with electric vehicles: A case study of Japanese households towards 2030," Energy Procedia, vol. 158, pp. 3802-3807, 2019.

Worighi, J. V. Mierlo, A. Maach, A. Hafid, Omar H. "Integrating Renewable Energi In Smart Grid System: Architecture, Virtualization And Analysis," Sustainable Energi, Grids And Networks, vol. 18, pp. 100-226. 2019.

Xiaohua Wu, Xiaosong Hu, Scott Moura, Xiaofeng Yin, Volker Pickert, "Stochastic control of smart home energy management with plug-in electric vehicle battery energy storage and photovoltaic array", Journal of Power Sources, vol. 333, pp. 203-212, 2016.

Young-Min Wi, Jong-Uk Lee, and Sung-Kwan Joo. "Electric vehicle charging method for smart homes/buildings with a photovoltaic system", IEEE Transactions on Consumer Electronics 2013, vol. 59, no. 2, pp. 323-328. 2013.

Z. Zhu, J. Tang, S. Lambotharan, W. H. Chin, Z. Fan, "An Integer Linear Programming Based Optimization for Home Demand-side Management in Smart Grid", IEEE PES Innovative Smart Grid Technologies (ISGT), 2012.

LAMPIRAN

1. Program

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed sep 21 14:50:27 2020

@author: ianadrian
"""
```

```
from pyomo.environ import *
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.cbook as cbook
import os
import pandas
import numpy
import datetime as dt
from dateutil.relativedelta import *
import xlsxwriter

#Export data dari excel
ketersediaan_kendaraan = pandas.read_excel('ev.xlsx')
pv_data = pandas.read_excel('pv.xlsx')
beban_data = pandas.read_excel('ph.xlsx')
c_pembelian_data = pandas.read_excel('c_pembelian.xlsx')
c_penjualan_data = pandas.read_excel('c_penjualan.xlsx')
harga = "Variable"

opsi_v2h = 'v2h'
#
objective_function = 'pemasukan'
#
#####################################################################
#################

#memasukkan nilai
ev_hari = ketersediaan_kendaraan['ev'].values
pv_p = pv_data['pv'].values*1000
permintaan = beban_data['ph'].values*1000
global memulai_Constraint
global waktu_pengisian_ev
memulai_Constraint = []

#pengaturan waktu
hari = 365
satu_hari = 24*1
satu_minggu = 24*7
nperiods = 24*hari#96*hari
shift = 0
nsteps = 8748
T_overlap = 12 #48
T_step = 1 #81
```

```
T_rh = T_overlap + T_step
step_finalizer = T_rh
shift = 0
shift_SatuMinggu = 0
delta_t = 1.0/1.0;  #time coefficient represents the 1 hour

#penggunaan NumPy untuk matriks di python.
#menggunakan pack Numpy dgn tipe data object, float dan int
l_ev_e_mtrs = numpy.zeros((nperiods), dtype = float) #EV energy level
soc_ev_mtrs = numpy.zeros((nperiods), dtype = float) #SOC EV
e_ev_ch_mtrs = numpy.zeros((nperiods), dtype = float) #EV charge
power
e_ev_dis_mtrs = numpy.zeros((nperiods), dtype = float) #EV discharge
power
e_utiliti_mtrs = numpy.zeros((nperiods), dtype = float) #Power drawn
fromthe grid
e_jual_mtrs = numpy.zeros((nperiods), dtype = float) #Powerinjected
to grid
ctrl_ev_ch_mtrs = numpy.zeros((nperiods), dtype = int) #EV charge
controller
ctrl_ev_dis_mtrs = numpy.zeros((nperiods), dtype = int) #EV discharge
controller
solution = numpy.empty((nperiods), dtype = object)
ketersediaan_kendaraan_mtrs = numpy.zeros((nperiods), dtype = int)

rh_range = range(T_rh)
model.RH = Set(initialize=rh_range, ordered = True) # Modelset
initialization

#biaya energi (konstan)
if harga == "Constant":
    c_pembelian = 0.0002899; #Rp/Wh
    c_penjualan = 0.00015; #Rp/Wh
else:
    c_pembelian_full = c_pembelian_data['c_pembelian'].values
    c_penjualan_full = c_penjualan_data['c_penjualan'].values


#EV(BMW i3) Battery kontrol
#https://www.bmw.com.tr/tr/all-models/bmw-i/i3/2017/range-charging-
efficiency.html
e_ev_ch_max = 4e3 #energi maksimal untuk pengisian
E_EV_min = 6e3 #minimum energi kendaraan listrik (w) #https://ev-
database.uk/car/1104/BMW-i3 W
E_EV_max = 33e3 #energi maksimal pada ev (w)
e_ev_awal = 15e3 #energi awal kendaraan
eff_ev_ch = 0.9 #efficiensi pengisian ev
eff_ev_dis = eff_ev_ch
n_EV_drive = 0.7
E_EV_goal = 26.4e3
P_EV_drive = 1.2e3 #https://www.holmgrensbil.se/globalassets/nya-
bilar/bmw/modellsidor/i3/dokument/i3-psl-eal_web.pdf
SOC_EV_goal = 0.8

#optimasi variabel kendaraan
#memberikan index/nilai awal (pack : pyomo)
```

```
#referensi ==>
https://pyomo.readthedocs.io/en/stable/pyomo_modeling_components/Vari
ables.html
model.e_ev_ch = Var(model.RH, within = NonNegativeReals)
model.e_ev_dis = Var(model.RH, within = NonNegativeReals)
model.ctrl_ev_ch = Var(model.RH, within = Binary)
model.ctrl_ev_dis = Var(model.RH, within = Binary)

#Energi EV
model.E_EV = Expression(model.RH)

#Fungsi perhitungan energi kendaraan
def e_ev_level(rng, iteration_number, ev_data, E_EV_initial):
    for counter in range(0, rng):
        if iteration_number < 1:
            if counter == 0:
                if ev_data[counter] == 1:
                    q = 1
                    m = 0
                else:
                    q = 0
                    m = 1
                    model.ctrl_ev_ch[counter] = 0
                    model.ctrl_ev_dis[counter] = 0
                model.E_EV[0] = E_EV_initial +
q*(model.e_ev_ch[0]*eff_ev_ch*delta_t - model.e_ev_dis[0] *
delta_t/eff_ev_dis) -m*((P_EV_drive/n_EV_drive)*delta_t)
            else:
                if ev_data[counter] == 1:
                    q = 1
                    m = 0
                else:
                    q = 0
                    m = 1
                    model.ctrl_ev_ch[counter] = 0
                    model.ctrl_ev_dis[counter] = 0
                model.E_EV[counter] = model.E_EV[counter-1] +
q*(model.e_ev_ch[counter]*eff_ev_ch*delta_t - model.e_ev_dis[counter]
* delta_t/eff_ev_dis) -m*((P_EV_drive/n_EV_drive)*delta_t)
        elif iteration_number >= 1:
            E_EV_initial = value(model.E_EV[T_step-1])
            if counter == 0:
                if ev_data[counter] == 1:
                    q = 1
                    m = 0
                else:
                    q = 0
                    m = 1
                    model.ctrl_ev_ch[counter] = 0
                    model.ctrl_ev_dis[counter] = 0
                model.E_EV[0] = E_EV_initial +
q*(model.e_ev_ch[0]*eff_ev_ch*delta_t - model.e_ev_dis[0] *
delta_t/eff_ev_dis) -m*((P_EV_drive/n_EV_drive)*delta_t)
            else:
                if ev_data[counter] == 1:
                    q = 1
                    m = 0
                else:
                    q = 0
                    m = 1
                    model.ctrl_ev_ch[counter] = 0
                    model.ctrl_ev_dis[counter] = 0
                model.E_EV[counter] = model.E_EV[counter-1] +
q*(model.e_ev_ch[counter]*eff_ev_ch*delta_t - model.e_ev_dis[counter]
* delta_t/eff_ev_dis) -m*((P_EV_drive/n_EV_drive)*delta_t) #persamaan
6
    return model.E_EV

#Constraint transaksi jual beli energi
e_utiliti_maks = 33000.0 #W
e_jual_maks = 33000.0 #W

#optimasi variable menggunakan pack = pyomo
# sama dgn variable kendaraan
model.e_utiliti = Var(model.RH, within = NonNegativeReals)
model.e_jual = Var(model.RH, within = NonNegativeReals)


#pengguna kendaraan listrik
deadline = 0;
model.SOC_EV = Expression(model.RH)
model.Min_Slots = Expression(model.RH)
min_slots = numpy.empty((T_rh), dtype = object)
model.pengguna_kendaraan = Expression(model.RH)
pengguna_kendaraan_level = numpy.ones(T_rh, dtype = object)

#fungsi untuk mengatur energi EV menjadi Nilai awal
#referensi https://www.geeksforgeeks.org/enumerate-in-python/
def difference(list_name):
    return [x - list_name[i - 1] for i, x in
enumerate(list_name)][1:]

#model pengguna kendaraan
def pengguna_kendaraan_calc(stop):
    global memulai_Constraint
    global waktu_pengisian_ev
    waktu_pengisian_ev = 0
    memulai_Constraint = numpy.zeros((13), dtype = int)
    waktu_pengisian_ev = 0
    memulai_pengisian_ev = 0
    for counter in range(0,stop):
        model.pengguna_kendaraan[counter] = 0.1

    ketersediaan_ev = difference(availability)
    for counter in range(0, T_rh-1):
        if ketersediaan_ev[counter] == -1:
            waktu_pengisian_ev = counter
            for counter_2 in range(0, counter):
                if ketersediaan_ev[counter_2] == 1:
                    memulai_pengisian_ev = counter_2
                else:
                    memulai_pengisian_ev = 0
            break
    if waktu_pengisian_ev > 0:
        for counter in range(memulai_pengisian_ev,
waktu_pengisian_ev+1):
```

```
            memulai_Constraint[counter] = 1
            min_slots[counter] = (E_EV_goal - 
model.E_EV[counter])/e_ev_ch_max
            pengguna_kendaraan_level[counter] = (waktu_pengisian_ev * 
delta_t - counter * delta_t + 1)
#            if pengguna_kendaraan_level[counter] <= 0:
#                pengguna_kendaraan_level[counter] = 1
            model.pengguna_kendaraan[counter] = min_slots[counter] / 
pengguna_kendaraan_level[counter]

        for counter in range(waktu_pengisian_ev+1, len(model.RH)):
            model.pengguna_kendaraan[counter] = 0.1

    return model.pengguna_kendaraan

#looping
for rh in range(0, nsteps):
    availability = ev_hari[0+shift:T_rh+shift]
    ketersediaan_ev =  difference(ev_hari[0+shift:12+shift])
    P_pv = pv_p[0+shift:T_rh+shift]
    P_demand = permintaan[0+shift:T_rh+shift]
    if harga == "Variable":
        c_pembelian = c_pembelian_full[0+shift:T_rh+shift]
        c_penjualan = c_penjualan_full[0+shift:T_rh+shift]


    E_EV_exp = e_ev_level(T_rh, rh, availability, e_ev_awal)
    pengguna_kendaraan_exp = pengguna_kendaraan_calc(T_rh)


#Constraint kendarann listrik
    model.EV_cons1 = Constraint(model.RH, rule = lambda model,  j: 0 
<= model.ctrl_ev_ch[j] + model.ctrl_ev_dis[j] <= 1) #mencegah 
pengisan dan pemakaian secara bersamaan persamaan 3
    model.EV_cons2 = Constraint(model.RH, rule = lambda model,  j: 
model.e_ev_ch[j] <= e_ev_ch_max*model.ctrl_ev_ch[j]) #batasan 
maksimal charger EV (persamaan 4)
    model.EV_cons3 = Constraint(model.RH, rule = lambda model,  j: 
model.e_ev_dis[j] <= e_ev_ch_max*model.ctrl_ev_dis[j]) #batassan 
maksimum penggunaan EV (persamaan 5)
    model.EV_cons4 = Constraint(model.RH, rule = lambda model,  j: 
E_EV_min <= E_EV_exp[j] <= E_EV_max) #persamaan 7 dan 8

    def chargeRule(model, j):
        if availability[j] == 0:
            return model.e_ev_ch[j] ==0
        else:
            return Constraint.Skip

    model.EV_cons5 = Constraint(model.RH, rule = chargeRule)

    def dischargeRule(model, j):
        if availability[j] == 0:
            return model.e_ev_dis[j] == 0
        else:
            return Constraint.Skip

    model.EV_cons6 = Constraint(model.RH, rule = dischargeRule)
    def minSlotsRule(model, j):
        if memulai_Constraint[j] ==  1:
            return model.E_EV[waktu_pengisian_ev] >= E_EV_goal
        else:
            return Constraint.Skip

    model.EV_cons7 = Constraint(model.RH, rule = minSlotsRule)

    if opsi_v2h == 'no_v2h':
        model.EV_cons8 = Constraint(model.RH, rule = lambda model, j: 
model.e_ev_dis[j] == 0)
    else:
        Constraint.Skip

#   batasan jaringan
    model.Grid_cons1 = Constraint(model.RH, rule = lambda model, j: 
0.0 <= model.e_utiliti[j] <= e_utiliti_maks) #persamaan 12
    model.Grid_cons2 = Constraint(model.RH, rule = lambda model, j: 
0.0 <= model.e_jual[j] <= e_jual_maks) #persamaan 13

#   keseimbangan energi
    if opsi_v2h == 'no_v2h':
        model.Balance_cons1 = Constraint(model.RH, rule = lambda 
model, j: P_demand[j] + model.e_ev_ch[j] - model.e_utiliti[j] - 
P_pv[j] - model.P_B_dis[j] + model.e_jual[j]==0)
    else:
        model.Balance_cons1 = Constraint(model.RH, rule = lambda 
model, j: P_demand[j] + model.e_ev_ch[j] - model.e_utiliti[j] - 
P_pv[j] - model.e_ev_dis[j] + model.e_jual[j]==0) #Persamaan 2

#   Objective Functions

    if objective_function == 'energy':
        model.energy = Objective(expr = sum((model.e_jual[j]*delta_t 
+ P_demand[j]*delta_t + model.e_ev_ch[j]*delta_t + 
model.P_B_ch[j]*delta_t)  + pengguna_kendaraan_exp[j]  for j in 
model.RH), sense = minimize)
    elif objective_function == 'cost':
        model.cost = Objective(expr = 
sum(model.e_utiliti[j]*c_pembelian[j]*delta_t + 
pengguna_kendaraan_exp[j]  for j in model.RH), sense = minimize)
    elif objective_function == 'cost_2':
        model.cost = Objective(expr = sum(model.e_utiliti[j]*delta_t 
+ pengguna_kendaraan_exp[j]  for j in model.RH), sense = minimize)
    elif objective_function == 'pemasukan':
        model.pemasukan = Objective(expr = 
sum(model.e_utiliti[j]*c_pembelian[j]*delta_t-
model.e_jual[j]*c_penjualan[j]*delta_t + pengguna_kendaraan_exp[j] 
for j in model.RH), sense = minimize) #persamaan (1)
    elif objective_function == 'pemasukan_3':
        model.pemasukan = Objective(expr = 
sum(model.e_utiliti[j]*delta_t-model.e_jual[j]*delta_t + 
pengguna_kendaraan_exp[j]  for j in model.RH), sense = minimize)
    elif objective_function == 'pemasukan_2':
        model.pemasukan = Objective(expr = 
sum(model.e_jual[j]*c_penjualan[j]*delta_t + 
pengguna_kendaraan_exp[j]  for j in model.RH), sense = maximize)
```

```
#    elif objective_function == 'no_v2h':
#        model.no_v2h = Objective(expr = sum(-
model.e_utiliti[j]*c_pembelian[j]*delta_t + pengguna_kendaraan_exp[j]
for j in model.RH), sense = minimize)
    elif objective_function == 'penggunaan_sendiri':
        model.penggunaan_sendiri = Objective(expr =
sum((model.e_utiliti[j] + model.e_jual[j])*delta_t +
pengguna_kendaraan_exp[j]  for j in model.RH), sense = minimize)
    else:
        print("pilih objek di line 34")


    solver = SolverFactory("gurobi")

    #solution[0+shift:T_rh+shift] = solver.solve(model)
    solver.solve(model)

#hasil matrix
    for i in model.RH:
        l_ev_e_mtrs[i+shift] = value(model.E_EV[i])
        soc_ev_mtrs[i+shift] = value(model.E_EV[i])/E_EV_max
        e_ev_ch_mtrs[i+shift] = value(model.e_ev_ch[i])
        e_ev_dis_mtrs[i+shift] = value(model.e_ev_dis[i])
        e_utiliti_mtrs[i+shift] = value(model.e_utiliti[i])
        e_jual_mtrs[i+shift] = value(model.e_jual[i])
        ctrl_ev_ch_mtrs[i+shift] = value(model.ctrl_ev_ch[i])
        ctrl_ev_dis_mtrs[i+shift] = value(model.ctrl_ev_dis[i])
    for i in range(0, 11):
        ketersediaan_kendaraan_mtrs[i+shift] = ketersediaan_ev[i]

    shift = shift+T_step
    print(rh)
    print(shift)

#Total Results
energi_utiliti = sum(e_utiliti_mtrs)*delta_t/1000
energi_jual = sum(e_jual_mtrs)*delta_t/1000
total_EV_charge = sum(e_ev_ch_mtrs)*delta_t/1000
total_EV_discharge = sum(e_ev_dis_mtrs)*delta_t/1000
total_pemakaian_ev = sum(ev_hari)*P_EV_drive*delta_t/1000
maximum_energi_utiliti = max(e_utiliti_mtrs)*delta_t/1000
maximum_energi_jual = max(e_jual_mtrs)*delta_t/1000
maximum_SOC_EV = max(soc_ev_mtrs)*100
minimum_SOC_EV = min(soc_ev_mtrs)*100
total_biaya_energi = sum(numpy.multiply(e_utiliti_mtrs,
c_pembelian_full))*delta_t
total_pemasukan_energy =
sum(numpy.multiply(e_jual_mtrs,c_penjualan_full))*delta_t
total_profit_energy =  (-sum(numpy.multiply(e_utiliti_mtrs,
c_pembelian_full))*delta_t +
sum(numpy.multiply(e_jual_mtrs,c_penjualan_full))*delta_t)

#data gambar
width = 1.0
data_range =500

tanggal_data = dt.datetime(2019, 1, 1, 0, 0) #tahun, bulan, tanggal,
jam, menit
```

```
time = [tanggal_data + dt.timedelta(minutes = j*1) for j in
range(len(l_ev_e_mtrs))]

#penyimpanan data
if os.path.isdir('/Users/ianadrian/Downloads/Document/program
HEM/48jam/'+objective_function+'_'+opsi_v2h+'_'+harga+'_harga_'+str(h
ari)+'_days_'+str(T_rh)+'_RH_size') == False:
    os.mkdir('/Users/ianadrian/Downloads/Document/program
HEM/48jam/'+objective_function+'_'+opsi_v2h+'_'+harga+'_harga_'+str(h
ari)+'_days_'+str(T_rh)+'_RH_size')
    destination = '/Users/ianadrian/Downloads/Document/program
HEM/48jam/'+objective_function+'_'+opsi_v2h+'_'+harga+'_harga_'+str(h
ari)+'_days_'+str(T_rh)+'_RH_size'
else:
    destination = '/Users/ianadrian/Downloads/Document/program
HEM/48jam/'+objective_function+'_'+opsi_v2h+'_'+harga+'_harga_'+str(h
ari)+'_days_'+str(T_rh)+'_RH_size'

#file
nama_file =
"hasil_program_"+opsi_v2h+'_'+harga+"_"+objective_function
complete_name = os.path.join(destination, nama_file+".txt")
file = open(complete_name, "w+")
file.write("##########hasil##########\r\n")
file.write("total energi dari utiliti  = %f kWh\r\n" %energi_utiliti)
file.write("total energi jual ke jaringan = %f kWh\r\n" %energi_jual)
file.write("total pengisian EV = %f kWh\r\n" %total_EV_charge)
file.write("total energi untuk v2h/v2g  = %f kWh\r\n"
%total_EV_discharge)
file.write("total pemakaian EV = %f kWh\r\n" %total_pemakaian_ev)
file.write("maksimal energi dari utiliti = %f kWh\r\n"
%maximum_energi_utiliti)
file.write("maksimum energi jual = %f kWh\r\n" %maximum_energi_jual)
file.write("kondisi maksimal betterai EV = %f \r\n" %maximum_SOC_EV)
file.write("kondiri minimum batterai EV = %f \r\n" %minimum_SOC_EV)
file.write("total biaya energi = %f Rp\r\n" %total_biaya_energi)
file.write("Total energy pemasukan = %f Rp\r\n"
%total_pemasukan_energy)
file.write("Total profit = %f Rp\r\n" %total_profit_energy)
file.write("########################################")
file.close()


#e_ev_ch_mtrs
workbook1 =
xlsxwriter.Workbook(destination+'/e_ev_ch_mtrs'+opsi_v2h+'_'+harga+'_
'+objective_function+'.xlsx')
worksheet1 = workbook1.add_worksheet()
row = 0
column = 0
for item in e_ev_ch_mtrs:
    worksheet1.write(row, column, item)
    row += 1
workbook1.close()

#e_ev_dis_mtrs
```

```
workbook2 =
xlsxwriter.Workbook(destination+'/e_ev_dis_mtrs'+opsi_v2h+'_'+harga+'
_'+objective_function+'.xlsx')
worksheet2 = workbook2.add_worksheet()
row = 0
column = 0
for item in e_ev_dis_mtrs:
    worksheet2.write(row, column, item)
    row += 1
workbook2.close()

#e_utiliti_mtrs
workbook3 =
xlsxwriter.Workbook(destination+'/e_utiliti_mtrs'+opsi_v2h+'_'+harga+
'_'+objective_function+'.xlsx')
worksheet3 = workbook3.add_worksheet()
row = 0
column = 0
for item in e_utiliti_mtrs:
    worksheet3.write(row, column, item)
    row += 1
workbook3.close()

#e_jual_mtrs
workbook4 =
xlsxwriter.Workbook(destination+'/e_jual_mtrs'+opsi_v2h+'_'+harga+'_'
+objective_function+'.xlsx')
worksheet4 = workbook4.add_worksheet()
row = 0
column = 0
for item in e_jual_mtrs:
    worksheet4.write(row, column, item)
    row += 1
workbook4.close()


#l_ev_e_mtrs
workbook5 =
xlsxwriter.Workbook(destination+'/l_ev_e_mtrs'+opsi_v2h+'_'+harga+'_'
+objective_function+'.xlsx')
worksheet5 = workbook5.add_worksheet()
row = 0
column = 0
for item in l_ev_e_mtrs:
    worksheet5.write(row, column, item)
    row += 1
workbook5.close()

#PV_out
workbook6 =
xlsxwriter.Workbook(destination+'/pv_p'+opsi_v2h+'_'+harga+'_'+object
ive_function+'.xlsx')
worksheet6 = workbook6.add_worksheet()
row = 0
column = 0
for item in pv_p:
    worksheet6.write(row, column, item)
    row += 1
```

```
workbook6.close()

for j in range(0, 52+1):

    shift_SatuMinggu = shift_SatuMinggu + satu_minggu
```