```
 1 import threading
 2 import time
 3 import serial
 4 import tkinter as tk
 5
 6 index = [0, 1, 2, 0, 1]
 7 cut_off_thrs = 3.3  # Lower threshold for consuming battery
    to stop feeding itself
 8 back_thrs = 3.5  # Higher threshold for charging battery to
    start feeding itself
 9 cut_off_other_thrs = 3.5  # Lower threshold for consuming
   battery to stop feeding others
10 back_other_thrs = 3.8  # Higher threshold for charging
   battery to start feeding others
11 Mains_power = 3
12 act_cmd = str.encode('2')  # Selector code for power source
    switching at Arduino
13 read_cmd = str.encode('1')  # Selector code for reading the
    values from Arduino
14
15 home_state = [0, 1, 2]  # Home states : Each home self
   feeding itself , code=3: means mains power supply
   consumption
16 home_serv_thrs = [3.5] * 3  # Home Serving thresholds : 3.5
    volt is normal cut off voltage for serving others
17 total_consumption = [[0.0] * 4 for i in range(3)]
18
19
20 def initialize_serial():
21     ser = serial.Serial(port='COM3', baudrate=9600, parity=
   serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE)
22     time.sleep(2)
23     return ser
24
25
26 def calculate_consumptions():
27     for i in range(3):
28         if home_state[i] == i:
29             total_consumption[i][0] = total_consumption[i][
   0] + p_array[i] / 60.0  # Using own green power
30         elif home_state[i] == index[i + 1] or home_state[i
   ] == index[i + 2]:
31             total_consumption[i][1] = total_consumption[i][
   1] + p_array[i] / 60.0  # Using other green power
32         else:
33             total_consumption[i][2] = total_consumption[i][
```

```
33 2] + p_array[i] / 60.0  # Using mains power
34         if home_state[index[i + 1]] == i or home_state[
   index[i + 2]] == i:
35             total_consumption[i][3] = total_consumption[i][
   3] + p_array[i] / 60.0  # Serving green power to others
36
37
38 def check_values_and_act(ser):  # Control each house
   voltage and check if switch needed
39
40     print(" Checking and acting ...")
41     for i in range(3):
42         if v_array[home_state[i]] <= cut_off_thrs:  #
   battery gets below 3.3v, switch action needed
43             ser.write(act_cmd)
44             ch = str(i)
45             cmd2_1 = str.encode(ch)
46             ser.write(cmd2_1)
47             print(cmd2_1)
48             max_ind = v_array.index(max(v_array[index[i + 1
   ]], v_array[index[i + 2]]))
49             if v_array[max_ind] >= home_serv_thrs[max_ind
   ]:  # use other house if it can feed this house
50                 home_state[i] = max_ind
51                 ch = str(max_ind)
52                 cmd2_2 = str.encode(ch)
53                 ser.write(cmd2_2)
54                 print(cmd2_2)
55             else:  # Else use Mains power to supply this
   house
56                 home_state[i] = Mains_power
57                 ch = str(Mains_power)
58                 cmd2_1 = str.encode(ch)
59                 ser.write(cmd2_1)
60                 print(cmd2_1)
61             home_serv_thrs[i] = back_other_thrs
62             res = int(ser1.readline())
63
64         if v_array[i] >= back_other_thrs:
65             home_serv_thrs[i] = cut_off_other_thrs  #
   charging battery gets above 3.7 , set thrs back to 3.5
66             if home_state[index[i + 1]] == Mains_power:  #
   if Mains power serving change it
67                 ser.write(act_cmd)
68                 home_state[index[i + 1]] = i
69                 ch = str(index[i+1])
```

```
 70                 cmd2_1 = str.encode(ch)
 71                 ser.write(cmd2_1)
 72                 print(cmd2_1)
 73                 ch = str(i)
 74                 cmd2_2 = str.encode(ch)
 75                 ser.write(cmd2_2)
 76                 print(cmd2_2)
 77                 res = int(ser1.readline())
 78             if home_state[index[i + 2]] == Mains_power:
    # if Mains power serving change it
 79                 ser.write(act_cmd)
 80                 home_state[index[i + 2]] = i
 81                 ch = str(index[i + 2])
 82                 cmd2_1 = str.encode(ch)
 83                 ser.write(cmd2_1)
 84                 print(cmd2_1)
 85                 ch = str(i)
 86                 cmd2_2 = str.encode(ch)
 87                 ser.write(cmd2_2)
 88                 print(cmd2_2)
 89                 res = int(ser1.readline())
 90
 91         if home_state[i] != i and v_array[i] >= back_thrs
    :  # charging battery gets above 3.5v
 92             ser.write(act_cmd)
 93             ch = str(i)
 94             cmd2_1 = str.encode(ch)
 95             ser.write(cmd2_1)
 96             print(cmd2_1)
 97             ch = str(i)
 98             cmd2_2 = str.encode(ch)
 99             ser.write(cmd2_2)
100             print(cmd2_2)
101             home_state[i] = i
102             res = int(ser1.readline())
103
104         if home_state[i] == i and v_array[i] <
    home_serv_thrs[i]:  # battery can't feed others , check
    others
105             if home_state[index[i + 1]] == i:  # index[i
     + 1] served by this house, it needs to be served by other
106                 ser.write(act_cmd)
107                 ch = str(index[i + 1])
108                 cmd2_1 = str.encode(ch)
109                 ser.write(cmd2_1)
110                 print(cmd2_1)
```

```
111                 if v_array[index[i + 2]] > home_serv_thrs[
    index[i + 2]]:
112                     ch = str(index[i + 2])
113                     cmd2_2 = str.encode(ch)
114                     ser.write(cmd2_2)
115                     print(cmd2_2)
116                     home_state[index[i + 1]] = index[i + 2
    ]
117                 else:
118                     ch = str(Mains_power)
119                     cmd2_2 = str.encode(ch)
120                     ser.write(cmd2_2)
121                     print(cmd2_2)
122                     home_state[index[i + 1]] = Mains_power
      # mains power
123                 res = int(ser1.readline())
124 
125             if home_state[index[i + 2]] == i:  # index[i
     + 2] served by this house, it needs to be served by other
126                 ser.write(act_cmd)
127                 ch = str(index[i + 2])
128                 cmd2_1 = str.encode(ch)
129                 ser.write(cmd2_1)
130                 print(cmd2_1)
131                 if v_array[index[i + 1]] > home_serv_thrs[
    index[i + 1]]:
132                     ch = str(index[i + 1])
133                     cmd2_2 = str.encode(ch)
134                     ser.write(cmd2_2)
135                     print(cmd2_2)
136                     home_state[index[i + 2]] = index[i + 1
    ]
137                 else:
138                     ch = str(Mains_power)
139                     cmd2_2 = str.encode(ch)
140                     ser.write(cmd2_2)
141                     print(cmd2_2)
142                     home_state[index[i + 2]] = Mains_power
      # mains power
143                 res = int(ser1.readline())
144 
145 
146 def getvalues(ser):
147     """thread worker function"""
148     global v_array
149     global i_array
```

```
150
151     while (not thread_quit_flag):
152
153         ser.write(read_cmd)
154         time.sleep(0.3)
155
156         for i in range(4):
157             v_array[i] = int(ser1.readline()) * 5.0 / 1023
158         for i in range(3):
159             i_array[i] = int(ser1.readline()) * 5.0 / 1023
160             if v_array[home_state[i]] < i_array[i]:
161                 p_array[i] = 0.0
162             else:
163                 v_diff = v_array[home_state[i]] - i_array[
    i]
164                 p_array[i] = ( v_diff / 0.33 ) * i_array[i
    ]
165
166         print("volts :",v_array)
167
168         calculate_consumptions()
169         check_values_and_act(ser)
170
171     ser.close()
172
173
174 def doQuitwin0():
175     global thread_quit_flag
176     thread_quit_flag = True
177     win0.destroy()
178
179
180 def win1_refresh():
181     for i in range(3):
182         for j in range(4):
183             power_value = "{:06.2f}".format(
    total_consumption[i][j]) + " Watt/min    "
184             if j == 2:
185                 fcolor = "red"
186             else:
187                 fcolor = "green"
188             lbl = tk.Label(win1, text=power_value, font=("
    Arial Bold", 18), foreground=fcolor)
189             lbl.grid(column=j + 1, row=i + 5)
190
191     win1.after(600, win1_refresh)
```

```
192
193
194 def win0_refresh():
195     for i in range(3):
196         if home_state[i] == i:
197             fcolor = "green"
198             if home_state[index[i + 1]] == i or home_state
    [index[i + 2]] == i:  # if some other house using my power
199                 bcolor = "yellow"
200             else:
201                 bcolor = "white"   # I am using my power
     , i don't serve to others
202         else:
203             if home_state[i] == Mains_power:
204                 fcolor = "red"     # i am using mains
    power
205             else:
206                 fcolor = "lawn green"  # i am using other
    house power
207             bcolor = "white"
208         voltage_value = "      " + "{:.2f}".format(v_array
    [home_state[i]]) + " Volt    "
209         lbl = tk.Label(win0, text=voltage_value, font=("
    Arial Bold", 20), background=bcolor, foreground=fcolor)
210         lbl.grid(column=i * 3, row=2)
211
212     for i in range(3):
213         if home_state[i] == Mains_power:
214             fcolor = "red"
215         else:
216             fcolor = "green"
217         current_value = "      " + "{:.2f}".format(p_array
    [i]) + " Watt    "
218         lbl = tk.Label(win0, text=current_value, font=("
    Arial Bold", 20),foreground=fcolor)
219         lbl.grid(column=i * 3, row=6)
220
221     win0.after(600, win0_refresh)
222
223
224 def win0_define():
225     win0.title("Enerji Değerleri ")
226     win0.geometry('770x450+10+10')
227
228     lbl = tk.Label(win0, text=" Anlık Enerji Değerleri ",
    font=("Arial Bold", 25))
```

```
229     lbl.grid(column=3, row=0)
230
231     for i in range(3):
232         header = "Ev " + str(i + 1)
233         lbl = tk.Label(win0, text=header, font=("Arial
    Bold", 20))
234         lbl.grid(column=i * 3, row=1)
235
236     lbl = tk.Label(win0, text=" ", font=("Arial Bold", 20
    ))
237     lbl.grid(column=6, row=4)
238
239     for i in range(3):
240         header = "Güç " + str(i + 1)
241         lbl = tk.Label(win0, text=header, font=("Arial
    Bold", 20))
242         lbl.grid(column=i * 3, row=5)
243
244     lbl = tk.Label(win0, text=" ", font=("Arial Bold", 20
    ))
245     lbl.grid(column=6, row=7)
246
247     col_count, row_count = win0.grid_size()
248
249     for col in range(col_count):
250         win0.grid_columnconfigure(col, minsize=3)
251
252     for row in range(row_count):
253         win0.grid_rowconfigure(row, minsize=10)
254
255     win0.grid_rowconfigure(0, minsize=100)  # Baslik icin
256
257     button = tk.Button(win0, text="Harcama", font=("Arial
    Bold", 15), command=win1_define)
258     # button.place(anchor='ne')
259     button.grid(column=6, row=8)
260
261     win0.protocol('WM_DELETE_WINDOW', doQuitwin0)
262
263
264 def win1_define():
265     global win1
266     win1 = tk.Toplevel(win0)
267     win1.geometry("1100x350+400+350")
268
269     lbl = tk.Label(win1, text=" Harcama Değerleri ", font
```

```
269 =("Arial Bold", 25))
270     lbl.grid(column=2, row=0)
271
272     header = "   Evler"
273     lbl = tk.Label(win1, text=header, font=("Arial Bold",
    20))
274     lbl.grid(column=0, row=2)
275
276     header = "Aküden"
277     lbl = tk.Label(win1, text=header, font=("Arial Bold",
    20))
278     lbl.grid(column=1, row=2)
279
280     header = "Kullanılan"
281     lbl = tk.Label(win1, text=header, font=("Arial Bold",
    20))
282     lbl.grid(column=1, row=3)
283
284     header = "Komşudan"
285     lbl = tk.Label(win1, text=header, font=("Arial Bold",
    20))
286     lbl.grid(column=2, row=2)
287
288     header = "Kullanılan"
289     lbl = tk.Label(win1, text=header, font=("Arial Bold",
    20))
290     lbl.grid(column=2, row=3)
291
292     header = "Şebekeden"
293     lbl = tk.Label(win1, text=header, font=("Arial Bold",
    20))
294     lbl.grid(column=3, row=2)
295
296     header = "Kullanılan"
297     lbl = tk.Label(win1, text=header, font=("Arial Bold",
    20))
298     lbl.grid(column=3, row=3)
299
300     header = "Komşuya"
301     lbl = tk.Label(win1, text=header, font=("Arial Bold",
    20))
302     lbl.grid(column=4, row=2)
303
304     header = "Verilen"
305     lbl = tk.Label(win1, text=header, font=("Arial Bold",
    20))
```

```
306     lbl.grid(column=4, row=3)
307
308     for i in range(3):
309         house_no = "     " + str(i + 1)
310         lbl = tk.Label(win1, text=house_no, font=("Arial
    Bold", 18))
311         lbl.grid(column=0, row=i + 5)
312
313     col_count, row_count = win1.grid_size()
314
315     for col in range(col_count):
316         win1.grid_columnconfigure(col, minsize=160)
317
318     for row in range(row_count):
319         win1.grid_rowconfigure(row, minsize=30)
320
321     win1_refresh()
322
323
324 #
325 #     MAIN CODE
326 #
327
328 thread_quit_flag = False
329 v_array = [0.0] * 4
330 i_array = [0.0] * 4
331 p_array = [0.0] * 4
332
333 ser1 = initialize_serial()
334 print("Serial Port Initialized")
335
336 t = threading.Thread(target=getvalues, args=(ser1,))
337 t.start()
338 print("Thread started")
339 time.sleep(0.5)
340
341 win0 = tk.Tk()
342 win0_define()
343
344 win0_refresh()
345 win0.mainloop()
346
```