#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import time

import MySQLdb

import RPi.GPIO as GPIO

#import sys

import random # para simulacion usando random.choice

import urllib2 # libreria para reles wifi

from smbus import SMBus

#import os # para envio de msg Telegram-cli .--OBSOLETO

#import subprocess # Uso E-mail

import Adafruit\_ADS1x15 # Import the ADS1x15 module.

import telebot # Librería de la API del bot.

from telebot import types # Tipos para la API del bot.

import token

usar\_telegram = 0 # poner a 0 para no usar / poner a 1 para usar

if usar\_telegram == 1:

###### Poner Nuestro token del bot creado

TOKEN = 'XXXXXXXXXXX'

bot = telebot.TeleBot(TOKEN) # Creamos el objeto de nuestro bot.

bot.skip\_pending = True # Skip the pending messages

###### ID de Usuario TELEGRAM

Aut = [xxxxx]

cid = Aut[0]

###### Direccion ip de la LAN para reles wifi

ip\_lan = "http://192.168.1."

#------------------------------------------------

###### ------------------------------------------

simular = 1 # Simulacion datos FV si =1

simular\_reles = 1 # Simular reles fisicos

mux = 0 # Poner a 1 si existe un multiplexor de 16 canales

sensortemperatura = 0 # Poner a 0 si no se ha instalado el DS18B20

# -----------------------------------------------

# -----------------------------------------------

bus = SMBus(1) # Bus I2C

GPIO.setmode(GPIO.BOARD)

###### Parametros Base de Datos

servidor = 'localhost'

usuario = 'rpi'

clave = 'fv'

basedatos = 'control\_solar'

# ------------------------------------------

###### PARAMETROS INSTALACION

# ------------------------------------------

SHUNT1 = 500.0/50 #Shunt Ibat

SHUNT2 = 200.0/75 #Shunt Iplaca

RES0 = 11.0\*25.1/25.4 # Divisor tension Vbat

RES1 = (4.7+1)/1 # Divisor tension salida Aux FM80 (Vflot/Diver)

RES2 = (38+1)/1\*64/60# Divisor tension Vplaca

vflotacion = 13.7 #valor por defecto de flotacion a 25ºC a 12V

vsis = 2 # Voltaje sistema - 1=12V 2=24V 4=48V

TIPO\_DIVER = 'vplaca' # valores posibles 'vplaca' , 'vaux' ..... poner 'NO' para desactivar diver

AH = 1200.0 # Capacidad C20

CP = 1.107 # Indice Peukert

EC = 1.0 # Eficiencia Carga

###### Alta ADS1115 ADC (16-bit).

#adc con pin addr a GND

#A0=iplaca // A1=Iplaca // // A2=Ibat // A3=ibat

adc = Adafruit\_ADS1x15.ADS1115(address=0x48, busnum=1)

#adc con pin addr a 3V3

#A0=Vbat // A1=Vflot o Diver // A2= Vplaca// A3= Mux

adc1 = Adafruit\_ADS1x15.ADS1115(address=0x49, busnum=1)

#---------------------------------------------------------------

DatosFV = {} #Creamos diccionario para los datos FV

OP = {'id\_rele':0,'nombre':1,'modo':2,'estado':3,'grabacion':4,'salto':5,'prioridad':6,'id\_rele2':7,'operacion':8,'parametro':9,'condicion':10,'valor':11}

OPH = {'id\_rele':0,'nombre':1,'modo':2,'estado':3,'grabacion':4,'salto':5,'prioridad':6,'id\_rele2':7,'parametro\_h':8,'valor\_h\_ON':9,'valor\_h\_OFF':10}

NDIA = {'D':0,'L':1,'M':2,'X':3,'J':4,'V':5,'S':6}

#Inicializando las variables del programa

Grabar = 1

T\_ejecucion\_max = 0.0

hora3 = 5.0

hora\_m = time.time() #para calcular tiempo entre muestras real

t\_muestra = 5 # Inicializo Tiempo entre muestra real...idealmente TP[0][2]

ibat = 0.0 # Intensidad Bateria

vbat = 24.0 # Voltaje Bateria

iplaca = 0.0 # Intensidad Placas (salida regulador)

vplaca = 0.0 # Volataje Placas (valor antes del regulador)

vaux = 0.0 # Voltaje salida auxiliar Regulador

temp = 0.0 # temperatura baterias

vflot = 0.0 # Voltaje asociado a estado de flotacion

nwifi\_lectura = 0 #utilizado en secuenciacion lectura reles wifi

#---Variables calculo SOC --------------------------------

Ip = 0.0

Ip1 = 0.0

Ip2 = 0.0

float(CP)

float(Ip)

float(Ip1)

float(Ip2)

DS=0.0

Puerto = 0

estado = 0

hora\_actual = time.strftime("%H")

tiempo = time.strftime("%Y-%m-%d %H:%M:%S")

kWh\_bat = 0.0

kWhp\_bat = 0.0

kWhn\_bat = 0.0

kWh\_placa = 0.0

kWh\_consumo = 0.0

GPIO.setup(7, GPIO.IN) # si se usa para señal vflot

# --------------------- DEFINICION DE FUNCIONES --------------

def act\_rele(adr,port,out) : # Activar Reles

if simular\_reles == 0:

if int(adr/10) == 2: #Rele WIFI

try:

ip = ip\_lan + str(adr) + "/R" + str(port) + "=" + str(out)

urllib2.urlopen(ip)

except:

if simular != 1:

logBD('Error rele wifi'+str(adr)+str(port)+'='+ str(out))

if int(adr/10) == 3: # Rele I2C

try:

estado = bus.read\_byte(adr) #devuelve el valor en decimal

if out == 100 :

i2c\_out = estado & (2\*\*(port-1) ^ (255))

else :

i2c\_out = estado | 2\*\*(port-1)

bus.write\_byte(adr,i2c\_out)

except:

if simular != 1:

logBD('Error bus I2C '+str(adr)+str(port)+ '='+ str(out))

return

def logBD(texto) : # Incluir en tabla de Log

try:

cursor.execute("""INSERT INTO log (Tiempo,log) VALUES(%s,%s)""",(tiempo,texto))

#print tiempo,' ', texto

db.commit()

except:

db.rollback()

return

def leer\_ibat(x) : # leer ibat

try:

y = - round(adc.read\_adc\_difference(3, gain=16, data\_rate=8) \* 0.0078125 \* SHUNT1, 2) # A2-A3

except:

y = x

logBD('-ERROR MEDIDA FV-sensor ibat')

if y < -200 or y > 200:

logBD('lectura incoherente ibat='+str(y))

y = x

return y

def leer\_iplaca(x) : # leer iplaca

try:

y = - round(adc.read\_adc\_difference(0, gain=16, data\_rate=8) \* 0.0078125 \* SHUNT2, 2) # A0-A1

except:

y = x

logBD('-ERROR MEDIDA FV-sensor iplaca')

if y > 150 or y < -1:

logBD('lectura incoherente iplaca='+str(y))

y = x

if y < 0.15:

y = 0

return y

def leer\_vbat(x) : # leer vbat

try:

y = round(adc1.read\_adc(0, gain=1) \* 0.000125 \* RES0, 2) # A0 4,096V/32767=0.000125

except:

y = x

logBD('-ERROR MEDIDA FV-sensor vbat')

if y < 20 or y > 32:

logBD('lectura incoherente vbat='+str(y))

y = x

return y

def leer\_vaux(x) : # leer vaux

try:

y = round(adc1.read\_adc(1, gain=1) \* 0.000125 \* RES1, 2) # A1 4,096V/32767=0.000125

except:

y = x

logBD('-ERROR MEDIDA FV-sensor vaux')

if y < -1 or y > 14:

logBD('lectura incoherente vaux='+str(y))

y = x

return y

def leer\_vplaca(x) : # leer vplaca

try:

y = round(adc1.read\_adc(2, gain=1) \* 0.000125 \* RES2, 2) # A2 4,096V/32767=0.000125

except:

y = x

logBD('-ERROR MEDIDA FV-sensor vplaca')

if y < 0 or y > 90:

logBD('lectura incoherente vplaca='+str(y))

y = x

return y

def leer\_diver(x) : # leer estado diversion

if x == "vaux":

vaux = leer\_vaux(0)

if vaux > 1:

diver = 1

else:

diver = 0

elif x == "vplaca":

vplaca = leer\_vplaca(0) # si error lectura devuelve 0

ibat = leer\_ibat(-10) # si error lectura devuelve -10

if vplaca > TP[0][5] and ibat > -10:

diver = 1

else:

diver = 0

else:

diver = 0

return diver

def leer\_temp(x) : # leer temperatura

try:

###### Actualizar 28-xxxxx segun DS18B20 instalado

tempfile= open("/sys/bus/w1/devices/28-04146c9606ff/w1\_slave")

thetext=tempfile.read()

tempfile.close()

tempdata = thetext.split("\n")[1].split(" ")[9]

y = round(float(tempdata[2:]) / 1000,2)

except:

y = x

logBD('-ERROR MEDIDA FV-sensor temp')

if y < -10 or y > 50:

logBD('lectura incoherente temp='+str(y))

y = x

return y

## RECUPERAR DE LA BD ALGUNOS DATOS ##

try:

db = MySQLdb.connect(host = servidor, user = usuario, passwd = clave, db = basedatos)

cursor = db.cursor()

sql='SELECT DS, DATE(Tiempo) FROM datos ORDER BY id DESC limit 1'

cursor.execute(sql)

var=cursor.fetchone()

DS=float(var[0])

HOY=str(var[1])

if HOY == time.strftime("%Y-%m-%d"):

sql='SELECT kWhp\_bat FROM datos ORDER BY id DESC limit 1'

cursor.execute(sql)

var=cursor.fetchone()

kWhp\_bat=float(var[0])

sql='SELECT kWhn\_bat FROM datos ORDER BY id DESC limit 1'

cursor.execute(sql)

var=cursor.fetchone()

kWhn\_bat=float(var[0])

sql='SELECT kWh\_placa FROM datos ORDER BY id DESC limit 1'

cursor.execute(sql)

var=cursor.fetchone()

kWh\_placa=float(var[0])

else:

kWhp\_bat=0.0

kWhn\_bat=0.0

kWh\_placa=0.0

print "Nuevo dia", time.strftime("%Y-%m-%d")

except Exception as e:

print "Error, la base de datos no existe"

## Definir matrices Rele\_Out y Rele\_Out\_Ant y apagar todos los reles

Rele\_Out = [[0] \* 8 for i in range(40)] # Situacion actual

Rele\_Out\_Ant = [[0] \* 8 for i in range(40)] # Situacion anterior

TR\_D = [[0] \* 3 for i in range(40)] #Para excedentes Id\_rele,Control,Timestamp

## ------ inicializamos reles apagandolos ------------------------

sql\_reles = 'SELECT \* FROM reles'

nreles = cursor.execute(sql\_reles)

nreles = int(nreles) # = numero de reles

TR = cursor.fetchall()

for I in range(nreles): #apagado fisico

Puerto = (TR[I][0] % 10) - 1

addr = int((TR[I][0] - Puerto)/10)

Rele\_Out[addr][Puerto] = 0

Rele\_Out\_Ant[addr][Puerto] = 0

act\_rele(addr,Puerto+1,0)

if nreles > 0 : # apagado en BD

sql = "UPDATE reles SET estado = 0"

cursor.execute(sql)

## ------------------------------------------------------------

### Calcular voltaje sistema (12,24 o 48)

#print ('ERROR LECTURA VOLTAJE BATERIA.....SISTEMA POR DEFECTO a 24V')

vbat = leer\_vbat(24) # pongo por defecto a 24v

log=''

if vbat > 11 and vbat < 15.5 :

vsis = 1

elif vbat > 22 and vbat < 31 :

vsis = 2

elif vbat > 44 and vbat < 62 :

vsis = 4

else :

log='Error: Imposible reconocer el voltaje del sistema'

#sys.exit() # Activar modulo sys si se descomenta

vflotacion = 13.7 \* vsis

print('Pulsa Ctrl-C para salir...')

log = ' Arrancando programa fv.py \nBateria = ' + str(vbat) + 'v' + log

logBD(log) # incluyo mensaje en el log

if usar\_telegram == 1:

bot.send\_message( cid, log)

# -------------------------------- BUCLE PRINCIPAL --------------------------------------

while True:

cursor.close()

db.close()

hora1=time.time()

### ------------ Cargar tablas parametros, reles , reles\_c, reles\_h ---------------------

db = MySQLdb.connect(host = servidor, user = usuario, passwd = clave, db = basedatos)

cursor = db.cursor()

sql\_reles='SELECT \* FROM parametros'

nparametros=cursor.execute(sql\_reles)

nparametros=int(nparametros) # = numero de filas de parametros.---- debe ser 1

TP=cursor.fetchall()

sql\_reles='SELECT \* FROM reles'

nreles=cursor.execute(sql\_reles)

nreles=int(nreles) # = numero de reles

TR=cursor.fetchall()

sql\_reles='SELECT \* FROM reles INNER JOIN reles\_c ON reles.id\_rele = reles\_c.id\_rele'

ncon=cursor.execute(sql\_reles)

ncon=int(ncon) # = numero de condiciones

R=cursor.fetchall()

sql\_reloj='SELECT \* FROM reles INNER JOIN reles\_h ON reles.id\_rele = reles\_h.id\_rele'

hcon=cursor.execute(sql\_reloj)

hcon=int(hcon) # = numero de condiciones horarias

H=cursor.fetchall()

# ------------------------ LECTURA FECHA / HORA ----------------------

tiempo = time.strftime("%Y-%m-%d %H:%M:%S")

diasemana = time.strftime("%w")

hora = time.strftime("%H:%M:%S") #No necesario .zfill() ya pone los ceros a la izquierda

# ------------------------ LECTURA PARAMETROS FV----------------------

t\_muestra\_ant=t\_muestra

t\_muestra=time.time()-hora\_m

hora\_m=time.time()

if simular==1:

ibat=random.choice([0,12,22,33,46,56,65,78,101,

-10,-20,-30,-40,-50,-60,-70,-80,-90])

iplaca=random.choice([0,10,20,30,45,57,67,77,88,99,102,110])

vbat=random.choice([22.5,23,24,24.4,25,26,27,27.5,28,29])

vplaca=random.choice([60,59.4,61,59.9,52,60.1,61.6,58.7,62,57.3])

diver=random.choice([1,1,1,0,1,0,1,1])

temp=random.choice([10,12,14,16,18,20,22,24,26,28,30,32,34])

else:

ibat = leer\_ibat(ibat)

vbat = leer\_vbat(vbat)

iplaca = leer\_iplaca(iplaca)

vplaca = leer\_vplaca(vplaca)

vaux = leer\_vaux(vaux)

TIPO\_DIVER = TP[0][6]

diver = leer\_diver(TIPO\_DIVER)

vflot = 0

# pongo 0, 1 ,2 o 3 en vflot dependiendo de estado vplaca/salida aux

if vaux > 1: # por ahora utilizare campo vflot para ver diver en la BD

vflot = 1

if diver == 1:

vflot = vflot + 2

if sensortemperatura == 1:

temp = leer\_temp(temp)

######## VALORES DEL MULTIPLEXOR ----PCF en Direccion 39---#########

if mux == 1:

###### Asegurar que el PCF del mux esta en la direccion 39 ==> A0=A1=A2=1

for K in range(16):

act\_rele(39,1,int(not(K%2))) #Pin S0 74HC4067

act\_rele(39,2,int(not((K//2)%2))) #Pin S1 74HC4067

act\_rele(39,3,int(not((K//4)%2))) #Pin S2 74HC4067

act\_rele(39,4,int(not((K//8)%2))) #Pin S3 74HC4067

#print 'direcc=',int(not(K%2)),

#print int(not((K//2)%2)),

#print int(not((K//4)%2)),

#print int(not((K//8)%2)),

try:

###### pin del ADS1115 para mux

mux1 = adc.read\_adc(1, gain=1)

#print ' Mux1=',mux1

#### FALTA INCORPORAR A BD ######

except:

logBD('-ERROR MEDIDA MUX1-'+ str(K))

##################################################################

### CALCULO KWH\_BAT y KWH\_PLACA

hora\_anterior=hora\_actual

hora\_actual=time.strftime("%H")

if (hora\_anterior== "23" and hora\_actual=="00"): #cambio de dia

kWh\_bat = 0.0

kWhp\_bat = 0.0

kWhn\_bat = 0.0

kWh\_placa = 0.0

else:

if ibat < 0:

kWhn\_bat = round(kWhn\_bat - (ibat \* vbat \* t\_muestra/3600),2)

else:

kWhp\_bat = round(kWhp\_bat + (ibat \* vbat \* t\_muestra/3600),2)

kWh\_placa = round(kWh\_placa + (iplaca \* vbat \* t\_muestra/3600),2)

###para kWh dividir por 1000

### CALCULO SOC% A C20

if ibat < 0 :

Ip1 = -ibat

Ip1 = Ip1\*\*CP

Ip1 = AH\*Ip1

Ip2 = AH / 20

Ip2 = (Ip2\*\*CP)\*20

Ip= -Ip1/Ip2

else :

Ip = ibat \* EC

if (ibat>0 and ibat<1 and abs(vbat-vflotacion)<0.2) :

DS = DS + (AH-DS)/50

else :

DS = DS + (Ip \* t\_muestra/3600)

if DS > AH :

DS = AH

if DS < 0 :

DS = 0

if TP[0][4] != 0: # Actualizo SOC si en la BD es distinto de 0

DS = AH\*TP[0][4]/100

cursor.execute("UPDATE parametros SET nuevo\_soc=0 WHERE id\_parametros=1")

db.commit()

soc = round(DS/AH\*100,2)

### FIN CALCULO SOC%

#------------- Asignamos valores al diccionario para parametros condiciones ...

DatosFV['Vbat'] = vbat

DatosFV['Ibat'] = ibat

DatosFV['SOC'] = soc

DatosFV['Iplaca'] = iplaca

DatosFV['Vflot'] = vflot

DatosFV['Temp'] = temp

DatosFV['Vplaca'] = vplaca

DatosFV['Diver'] = diver

#------------------------ ALGORITMO CONDICIONES RELES -----------------------------

#### Cargamos los valores actuales de los reles en Rele\_Out\_Ant####

nwifi = 0

for I in range(nreles): # Calculo Numero reles wifi y Rele\_Out\_Ant por defecto

Puerto = (TR[I][0] % 10) - 1

addr = int((TR[I][0] - Puerto) / 10)

Rele\_Out\_Ant[addr][Puerto] = Rele\_Out[addr][Puerto]

if int(addr/10) == 2:

nwifi = nwifi + 1

nwifi\_lectura = nwifi\_lectura + 1 # lectura de un unico rele wifi por ciclo

if nwifi\_lectura > nwifi: # para evitar colapsar al NodeMCU

nwifi\_lectura = 0

nwifi\_lectura1 = 0

if simular\_reles == 0: #Captura de los valores reales que estan en los reles

for I in range(nreles):

Puerto = (TR[I][0] % 10) - 1

addr = int((TR[I][0] - Puerto) / 10)

#print 'Puerto=',Puerto, addr

if int(addr/10) == 3: # Reles I2C

try:

estado = bus.read\_byte(addr)

estado = bin(estado ^ 255)[2:10].zfill(8)

Rele\_Out\_Ant[addr][Puerto] = int(estado[7-Puerto]) \* 100

except:

logBD('Error lectura I2C en direccion/ '+str(addr)+str(Puerto))

if int(addr/10) == 2: # Reles Wifi lectura un solo rele por ciclo

nwifi\_lectura1 = nwifi\_lectura1 + 1

if nwifi\_lectura == nwifi\_lectura1:

try:

ip = ip\_lan + str(addr) + "/R"+str(Puerto+1) + "=?"

sock = urllib2.urlopen(ip,timeout=10)

lines = sock.readlines()

salida\_node = lines[1]

if salida\_node[0:7] == "PWM\_R" + str(Puerto+1) + "=":

Rele\_Out\_Ant[addr][Puerto] = int(salida\_node[7:10])

if salida\_node[0:5] == "ERROR":

Rele\_Out\_Ant[addr][Puerto] = 0

logBD(' ERROR Rele ' + str(Puerto+1) + ' en NodeMCU de direccion '+str(addr))

except:

logBD('Rele ' + str(Puerto + 1) + ' en NodeMCU de direccion '+str(addr)+' NO ENCONTRADO')

#### Apagamos virtualmente y encendemos SI condiciones FV o HORARIAS por defecto####

for I in range(ncon): # enciendo reles con condiciones FV

Puerto = (R[I][0] % 10) - 1

addr = int((R[I][0]-Puerto) / 10)

Rele\_Out[addr][Puerto] = R[I][5] # pongo valor del salto

for I in range(hcon): # enciendo reles con condiciones horario

Puerto = (H[I][0] % 10)-1

addr = int((H[I][0] - Puerto)/10)

Rele\_Out[addr][Puerto] = H[I][5] # pongo valor del salto

for I in range(nreles): # reles diver se ponen a situacion anterior

if TR[I][6] != 0:

Puerto = (TR[I][0] % 10) - 1

addr = int((TR[I][0] - Puerto) / 10)

Rele\_Out[addr][Puerto] = Rele\_Out\_Ant[addr][Puerto]

# -------------------- Bucle de condiciones de horario --------------------------

Rele\_Out\_H = [[0] \* 8 for i in range(40)] # Inicializo variable a 0

for I in range(hcon):

Puerto = (H[I][0] % 10) - 1

addr = int((H[I][0] - Puerto) / 10)

diaok = 0 # variables de control para ver si esta dentro de horario

horaok = 0

if H[I][OPH['parametro\_h']] == 'T': #Todos los dias de la semana

diaok = 1

elif str(NDIA[H[I][OPH['parametro\_h']]]) == diasemana:

diaok = 1

if str(H[I][OPH['valor\_h\_ON']]).zfill(8) > str(H[I][OPH['valor\_h\_OFF']]).zfill(8): #True si periodo pasa por 0:00

if (hora >= str(H[I][OPH['valor\_h\_ON']]).zfill(8) and hora <= "23:59:59"):

horaok = 1

if (hora >= "00:00:00" and hora <= str(H[I][OPH['valor\_h\_OFF']]).zfill(8)):

horaok = 1

elif (hora >= str(H[I][OPH['valor\_h\_ON']]).zfill(8) and hora <= str(H[I][OPH['valor\_h\_OFF']]).zfill(8)):

horaok = 1

if diaok == 1 and horaok == 1:

Rele\_Out\_H[addr][Puerto]+= 1

for I in range(hcon):

Puerto = (H[I][0] % 10)-1

addr = int((H[I][0] - Puerto) / 10)

if Rele\_Out\_H[addr][Puerto] == 0:

Rele\_Out[addr][Puerto] = 0 #apago rele

Rele\_Out\_H[addr][Puerto] = -1 # para quitar posibilidad de ser rele diver en el ciclo

# -------------------- Bucle de condiciones de parametros FV --------------------------

for I in range(ncon):

Puerto = (R[I][0] % 10) - 1

addr = int((R[I][0] - Puerto) / 10)

if R[I][OP['condicion']] == '<':

if R[I][OP['operacion']] == 'ON' and DatosFV[R[I][OP['parametro']]] > R[I][OP['valor']] and Rele\_Out\_Ant[addr][Puerto] == 0 :

Rele\_Out[addr][Puerto] = 0

if R[I][OP['operacion']] == 'OFF' and DatosFV[R[I][OP['parametro']]] <= R[I][OP['valor']] :

Rele\_Out[addr][Puerto] = 0

if R[I][OP['condicion']] == '>':

if R[I][OP['operacion']] == 'ON' and DatosFV[R[I][OP['parametro']]] < R[I][OP['valor']] and Rele\_Out\_Ant[addr][Puerto] == 0 :

Rele\_Out[addr][Puerto] = 0

if R[I][OP['operacion']] == 'OFF' and DatosFV[R[I][OP['parametro']]] >= R[I][OP['valor']] :

Rele\_Out[addr][Puerto] = 0

#-------------------- Bucle encendido/apagado reles ------------------------------------

Flag\_Rele\_Encendido = 0

for I in range(nreles):

Puerto = (TR[I][0] % 10) - 1

addr = int((TR[I][0] - Puerto) / 10)

### forzado ON/OFF

if TR[I][OP['modo']] == 'ON' :

Rele\_Out[addr][Puerto] = 100

if TR[I][OP['modo']] == 'OFF' :

Rele\_Out[addr][Puerto] = 0

### dejar rele como esta

if Rele\_Out[addr][Puerto] == 100 and Rele\_Out\_Ant[addr][Puerto] < 100 and Flag\_Rele\_Encendido == 1 :

Rele\_Out[addr][Puerto] = Rele\_Out\_Ant[addr][Puerto] #dejar rele en el estado anterior

### encender rele

if Rele\_Out[addr][Puerto] == 100 and Rele\_Out\_Ant[addr][Puerto] < 100 and Flag\_Rele\_Encendido == 0:

Flag\_Rele\_Encendido = 1

act\_rele(addr,Puerto+1,100)

### apagar rele

if Rele\_Out[addr][Puerto] == 0 and Rele\_Out\_Ant[addr][Puerto] > 0 :

act\_rele(addr,Puerto+1,0) #apagar rele

### CONTROL DE EXCEDENTES -- DIVER

ndiver = 1 # Numero de ciclos diver por muestra

tdiver = str(diver) # Almacena secuencia diver

while True:

if simular == 1:

diver = random.choice([1,0,1,0,1,0,1,0,1])

else:

diver = leer\_diver(TIPO\_DIVER)

tdiver = tdiver + str(diver)

if diver == 1 : #hay excedentes

# Pongo 1 en TR\_D[P][1] en reles candidatos a diver

# y busco prioridad\_actual

prioridad\_actual = 100

for P in range(nreles):

Puerto = (TR[P][0] % 10) - 1

addr = int((TR[P][0] - Puerto) / 10)

TR\_D[P][0] = TR[P][0]

TR\_D[P][1] = 0

if TR[P][2] == 'PRG' and Rele\_Out[addr][Puerto] < 100 and TR[P][6]!= 0 and Rele\_Out\_H[addr][Puerto] != -1:

TR\_D[P][1] = 1

if TR[P][6] < prioridad\_actual :

prioridad\_actual = TR[P][6]

else:

TR\_D[P][1] = 0

if prioridad\_actual != 100:

# Pongo 2 en TR\_D[P][1] si coincide prioridad actual

# y busco rele con la marca temporal mas antigua

timerele = 9999999999

id\_rele\_diver = 0

for P in range(nreles):

if TR[P][6] == prioridad\_actual and TR\_D[P][1] == 1 and TR\_D[P][2] < timerele:

timerele = TR\_D[P][2]

id\_rele\_diver = TR\_D[P][0]

for P in range(nreles):

if TR[P][0] == id\_rele\_diver:

Puerto = (TR[P][0] % 10) - 1

addr = int((TR[P][0] - Puerto) / 10)

Rele\_Out[addr][Puerto] = Rele\_Out[addr][Puerto] + TR[P][5]

if Rele\_Out[addr][Puerto] > 100:

Rele\_Out[addr][Puerto] = 100

act\_rele(addr,Puerto+1,Rele\_Out[addr][Puerto])

timerele = time.clock()

TR\_D[P][2] = timerele

else: # NO hay excedentes diver==0

prioridad\_actual = 0

for P in range(nreles):

Puerto = (TR[P][0] % 10) - 1

addr = int((TR[P][0] - Puerto) / 10)

TR\_D[P][0] = TR[P][0]

if TR[P][2] == 'PRG' and Rele\_Out[addr][Puerto] > 0 and TR[P][6]!= 0 :

TR\_D[P][1] = 1

if TR[P][6] > prioridad\_actual :

prioridad\_actual = TR[P][6]

else:

TR\_D[P][1] = 0

if prioridad\_actual != 0:

timerele = 9999999999

id\_rele\_diver = 0

for P in range(nreles):

if TR[P][6] == prioridad\_actual and TR\_D[P][1] == 1 and TR\_D[P][2] < timerele:

timerele = TR\_D[P][2]

id\_rele\_diver = TR\_D[P][0]

for P in range(nreles):

if TR[P][0] == id\_rele\_diver:

Puerto = (TR[P][0] % 10) - 1

addr = int((TR[P][0] - Puerto) / 10)

Rele\_Out[addr][Puerto] = Rele\_Out[addr][Puerto] - TR[P][5]

if Rele\_Out[addr][Puerto] < 0:

Rele\_Out[addr][Puerto] = 0

act\_rele(addr,Puerto+1,Rele\_Out[addr][Puerto])

timerele = time.clock()

TR\_D[P][2] = timerele

ndiver = ndiver + 1

time.sleep(0.2)

if time.time() - hora1 > TP[0][2] - 1: #Tmuestra menos 1 segundo

break

#if int(hora\_actual) < 21 and int(hora\_actual) > 8:

# print hora, ndiver, tdiver

# -------------------------- Escribir en la BD Tabla Reles el Estado RELES -------------------------

for I in range(nreles):

Puerto = (TR[I][0] % 10) - 1

addr = int((TR[I][0] - Puerto) / 10)

id = TR[I][0]

estado = Rele\_Out[addr][Puerto]

sql = "UPDATE reles SET estado =" +str(estado)+ " WHERE id\_rele = " + str(id)

sql\_reles = 'SELECT id\_rele,segundos\_on,nconmutaciones FROM reles\_segundos\_on WHERE fecha='+'"'+time.strftime("%Y-%m-%d")+'" and id\_rele =' + str(id)

try:

nreles\_on = cursor.execute(sql\_reles)

nreles\_on = int(nreles\_on)

if estado > 0 :

segundos\_on = TP[0][2]

nconmutaciones = 1

if nreles\_on >= 1:

TS = cursor.fetchall()

segundos\_on = TS[0][1] + round((t\_muestra\*estado/100),2) # calculo funcionamiento real reles PWM

if TR[I][3] == 0:

nconmutaciones = TS[0][2] + 1

else:

nconmutaciones = TS[0][2]

#UPDATE

sql\_reles = ("UPDATE reles\_segundos\_on SET segundos\_on =" +str(segundos\_on)+

",nconmutaciones =" + str(nconmutaciones)+ " WHERE id\_rele = " +

str(id) + ' and fecha = "' + time.strftime("%Y-%m-%d") +'"')

cursor.execute(sql\_reles)

else :

#INSERT

cursor.execute("""INSERT INTO reles\_segundos\_on

(id\_rele,fecha,segundos\_on,nconmutaciones)

VALUES (%s,%s,%s,%s)""",

(id,time.strftime("%Y-%m-%d"),segundos\_on,1))

if Rele\_Out[addr][Puerto] != TR[I][3]:

cursor.execute(sql)

## db.commit()

except:

db.rollback()

logBD('Error grabacion Reles\_segundos\_on')

if TP[0][1] == "S" and Grabar == 1 and TR[I][4] == "S" and Rele\_Out[addr][Puerto] != TR[I][3]:

try:

cursor.execute("""INSERT INTO reles\_grab (Tiempo,id\_rele,valor\_rele)

VALUES(%s,%s,%s)""",(tiempo,TR[I][0],estado))

## db.commit()

except:

db.rollback()

logBD('tabla reles\_grab NO grabados por fallo')

#------------------------Escribir en la tabla valores FV ---------------------------

if TP[0][0] == "S" and Grabar == 1:

try:

cursor.execute("""INSERT INTO datos (Tiempo,Ibat,Vbat,SOC,DS,Vflot,kWhp\_bat,kWhn\_bat,Iplaca,Vplaca,kWh\_placa,Temp)

VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)""",

(tiempo,ibat,vbat,soc,DS,vflot,kWhp\_bat,kWhn\_bat,iplaca,vplaca,kWh\_placa,temp))

db.commit()

except:

db.rollback()

logBD('Registro DATOS no grabado')

else :

db.commit()

#print "registro DATOS NO grabado"

Grabar = Grabar + 1

if Grabar >= TP[0][3] + 1:

Grabar = 1

if t\_muestra > 5:

logBD('Tiempo muestra elevado ='+str(round(t\_muestra,1))+'sg')

hora2 = time.time()

###### ajuste fino tiempo bucle=0.10

T\_ejecucion = hora2 - hora1 + 0.10

if T\_ejecucion\_max < T\_ejecucion:

T\_ejecucion\_max = T\_ejecucion

# Repetir bucle cada X segundos

espera = TP[0][2] - T\_ejecucion

if espera > 0:

time.sleep(espera)