```
import time
import board
import busio
#Bibliothek für die GPIO's
#import RPi.GPIO as GPIO
#numpy steht für Numeric Phyton eine mächtige Bibliothek für die
Benutzung von Arrays
import numpy as np
#Bibliothek für den Lichtsensor TSL2561
import adafruit_tsl2561
#Bibliotheken für den Analog Sensor ADS1115
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
#Bibliotheken für influxdb
from influxdb_client import InfluxDBClient, Point

#wir geben an, ob wir die GPIOs per Boardnummern (1-40) oder über
ihre GPIO Nummer ansprechen wollen. Da wir letzteres wollen, lautet
der Befehl dazu:
#GPIO.setmode(GPIO.BCM)
#Pin 11 = GPIO17 als Ausgang definieren
#GPIO.setup(17, GPIO.OUT)
#influxdb
username = 'solar'
password = 'solar'

database = 'solar'
retention_policy = 'solardata'

bucket = f'{database}/{retention_policy}'

client = InfluxDBClient(url='http://localhost:8086',
token=f'{username}:{password}', org='-')
write_api = client.write_api()

# Create the I2C bus
i2c = busio.I2C(board.SCL, board.SDA)#die Adressen der
angesclossenen Sensoren können mit dem Befehl 'i2cdetect -y 1' in
der Konsole ermittelt werden

# Create the TSL2561 instance, passing in the I2C bus
tsl = adafruit_tsl2561.TSL2561(i2c, 0x39)
# Enable the light sensor
tsl.enabled = True
time.sleep(1)
# Set gain 0=1x, 1=16x
tsl.gain = 0
# Set integration time (0=13.7ms, 1=101ms, 2=402ms, or 3=manual)
tsl.integration_time = 1

# Create the ADC object using the I2C bus
adsBatKomp = ADS.ADS1115(i2c,address=0x4B) #misst den Strom  der
Batterie --> aller daran angeschlossenen Komponenten und die
Spannung am Ausgang des Spannungsreglers --> Pi
```

```
adsLRBat = ADS.ADS1115(i2c, address=0x48)#misst Spannung und Strom
vom Laderegler zur Batterie, kann wenn der Laderegler weggeschalten
wird, die Spannung der Batterie messen


# Achtung Differenzmessung z.B P0 = GND = 0V - P1 = 3V == 0-3 = -3
(mit -1 multiplizieren um Vorzeichen umzukehren)
chan_strom_Bat_Komp = AnalogIn(adsBatKomp, ADS.P0,ADS.P1) #Strom
messen Batterie --> aller daran angeschlossenen Komponenten
chan_spannung_Pi = AnalogIn(adsBatKomp, ADS.P2, ADS.P3)#Spannung
messen Eingang Pi
chan_strom_LR_Bat = AnalogIn(adsLRBat, ADS.P0,ADS.P1) #Stromfluss
Laderegler --> Batterie messen
chan_spannung_LRorBat = AnalogIn(adsLRBat, ADS.P2, ADS.P3)#Spannung
Batterie oder Laderegler messen (für die Spannung vom Laderegler
muss chan_strom_LR_Bat hinzuaddiert werden)

#Arrays für die Messwerte erstellen und mit dem ersten gemessenen
Wert vorausfüllen
array_groesse = 120 # Größe der Arrays für die Messwerte / sollte
ca. 12 x Anzahl Sekunden für den Daten Speichertakt (Variable
'daten_speichern_takt') sein
#Strom gemessen für alle Komponenten ab der Batterie,
Spannungswandler Pi Wifi Modul
x = float(chan_strom_Bat_Komp.voltage)/1.05*10000*-1
strom_Bat_Komp = np.linspace(x,x,array_groesse)
#Spannung die am Pi anliegt, Ausgang Spannungsregler
x = float(chan_spannung_Pi.voltage)*10.9*-1
spannung_pi = np.linspace(x,x,array_groesse)
#Strom: Laderegler Ausgang --> Batterie
x = float(chan_strom_LR_Bat.voltage)/1.05*10000
strom_LR_Bat = np.linspace(x,x,array_groesse)
#Spannung der Batterie --- kann nur richtig gemessen werden wenn gar
keine Spannung vom Laderegler anliegt oder wenn die Spannung vom
Laderegler
#zur Batterie kleiner ist als die Spannung der Batterie
x = float(chan_spannung_LRorBat.voltage)*11.128*-1
spannung_Bat = np.linspace(x,x,array_groesse)
#Spannung Laderegler, sollte die Spannung der Batterie höher sein
als die des Ladereglers wird die Spannung der Batterie gemessen
x = (float(chan_spannung_LRorBat.voltage)*11.128*-1) +
(float(chan_strom_LR_Bat.voltage)/1.05)
spannung_LR = np.linspace(x,x,array_groesse)

lux_wert = tsl.lux #Rückgabe von tsl.lux ist der Wert oder None
if lux_wert is not None:
    x = float(lux_wert)
    licht_lux = np.linspace(x,x,array_groesse)
else:
    licht_lux = np.linspace(100,100,array_groesse)

array_index_nr = 0 #Zählvariable für die Dauerschleife von 0 bis
'array_groesse - 1' für 'array_groesse' Felder und dann wieder auf 0
daten_speichern_takt = 10 # alle n Sekunden einen Datensatz
```

```
speichern
zeit_periode_speichern_start = time.time()
#messzeit = time.time()
while True:# Faktor bei Strom z.B. 1.05 ist Shunt Widerstand +
Drahtwiderstand bis Messpunkt
    strom_Bat_Komp[array_index_nr] =
float(chan_strom_Bat_Komp.voltage)/1.05*10000*-1 #Multiplikation mit
-1 um Vorzeichen umzukehren.
    spannung_pi[array_index_nr] =
float(chan_spannung_Pi.voltage)*11*-1 #Faktor 11 wegen
Spannungsteiler
    strom_LR_Bat[array_index_nr] = float(chan_strom_LR_Bat.voltage)/
1.05*10000*-1
    spannung_Bat[array_index_nr] =
float(chan_spannung_LRorBat.voltage)*11.128*-1
    spannung_LR[array_index_nr] =
(float(chan_spannung_LRorBat.voltage)*11.128*-1) +
(float(chan_strom_LR_Bat.voltage)/1.05*-1)
    leistung_Verbrauch_Komp = strom_Bat_Komp.mean() /1000 *
spannung_Bat.mean()
    leistung_LR_Bat = strom_LR_Bat.mean() / 1000 *
spannung_LR.mean()
    lade_entladestrom = strom_LR_Bat.mean() - strom_Bat_Komp.mean()
    lux_wert = tsl.lux
    if lux_wert is not None:
        licht_lux[array_index_nr] = lux_wert

    if zeit_periode_speichern_start + daten_speichern_takt <=
time.time():
        print("{:>6.3f}VPi\t{:>6.0f}mA_Bat_Komp\t{:>6.3f}
V_LR\t{:>6.3f}V_Bat\t{:>6.0f}mA_LR_Bat\t{:>6.1f} Watt_Komp\t{:>5.1f}
Watt_LR_Bat\t{:>6.0f} Lux".format(spannung_pi.mean(),
strom_Bat_Komp.mean(),spannung_LR.mean(),spannung_Bat.mean(),strom_L
R_Bat.mean(),leistung_Verbrauch_Komp,leistung_LR_Bat,licht_lux.mean(
)))#gemessen mit Shunt 0,1 Ohm ergibt voltage*10000 = mA
        #Daten in die Datenbank schreiben
        point = Point("messungen").tag("messort",
"LR_Bat").tag("messart","strom").field("Wert", strom_LR_Bat.mean())
        write_api.write(bucket=bucket, record=point)
        point = Point("messungen").tag("messort",
"Bat_Komp").tag("messart","strom").field("Wert",
strom_Bat_Komp.mean())
        write_api.write(bucket=bucket, record=point)
        point = Point("messungen").tag("messort",
"SR_Pi").tag("messart","spannung").field("Wert", spannung_pi.mean())
        write_api.write(bucket=bucket, record=point)
        point = Point("messungen").tag("messort",
"Bat_SR").tag("messart","leistung").field("Wert",
leistung_Verbrauch_Komp)
        write_api.write(bucket=bucket, record=point)
        point = Point("messungen").tag("messort",
"LR_Bat").tag("messart","leistung").field("Wert", leistung_LR_Bat)
        write_api.write(bucket=bucket, record=point)
        point = Point("messungen").tag("messort",
```

```
"LR_Bat").tag("messart","spannung").field("Wert",
spannung_LR.mean())
        write_api.write(bucket=bucket, record=point)
        point = Point("messungen").tag("messort",
"Bat_SR").tag("messart","spannung").field("Wert",
spannung_Bat.mean())
        write_api.write(bucket=bucket, record=point)
        point = Point("messungen").tag("messort",
"Berechnung").tag("messart","ladestrom").field("Wert",
lade_entladestrom)
        write_api.write(bucket=bucket, record=point)
        point = Point("messungen").tag("messort",
"Lichtsensor").tag("messart","licht").field("Wert",
licht_lux.mean())
        write_api.write(bucket=bucket, record=point)

        #Messung der Batteriespannung muss gesondert behandelt
werden
        #der Ausgang schaltet ein Relais welches den Laderegler von
der Batterie trennt um die tatsächliche Spannung der Batterie zu
messen
        #GPIO.output(17, GPIO.LOW)
        #entprellen
        #time.sleep(2)
        #messen
        #spannung_batterie =
float(chan_spannung_LRorBat.voltage)*11.128*-1
        #Laderegler wird wieder zugeschalten
        #GPIO.output(17, GPIO.HIGH)
        #point = Point("messungen").tag("messort",
"Bat_LR_getrennt").tag("messart","spannung").field("Wert",
spannung_batterie)
        #write_api.write(bucket=bucket, record=point)
        zeit_periode_speichern_start = time.time()


    if array_index_nr == array_groesse - 1 :
        array_index_nr = 0
        #print(time.time() - messzeit)
        #messzeit = time.time()
    else:
        array_index_nr += 1
```