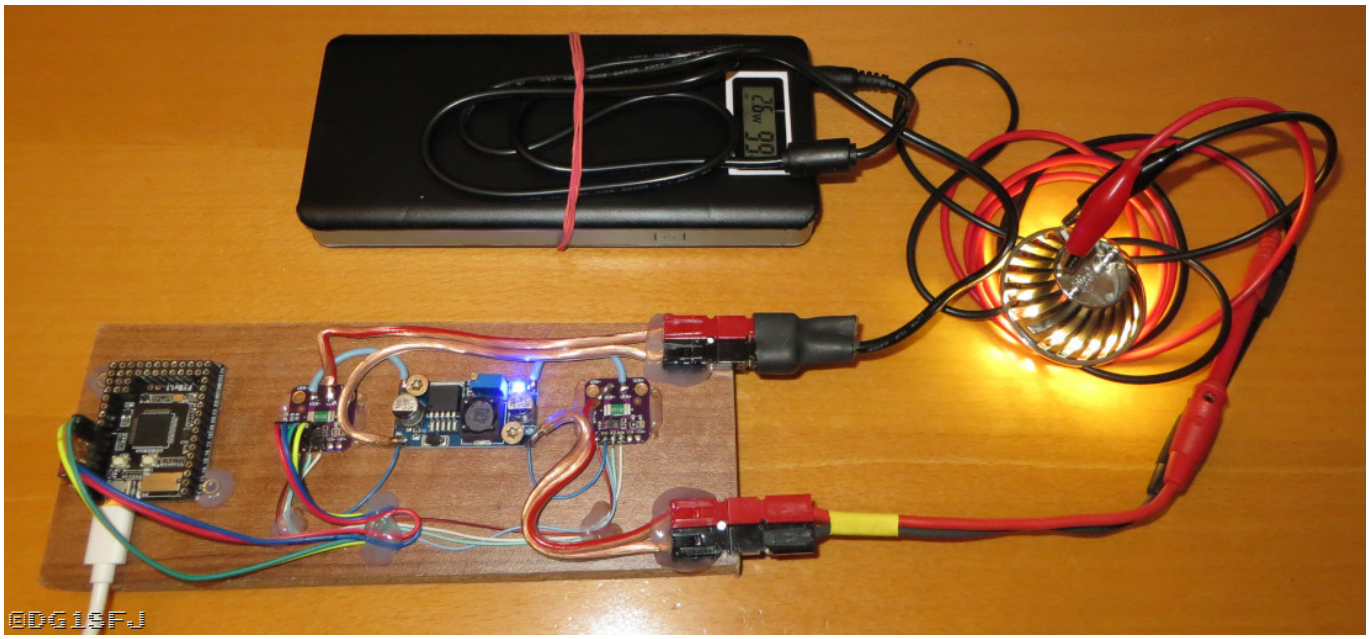
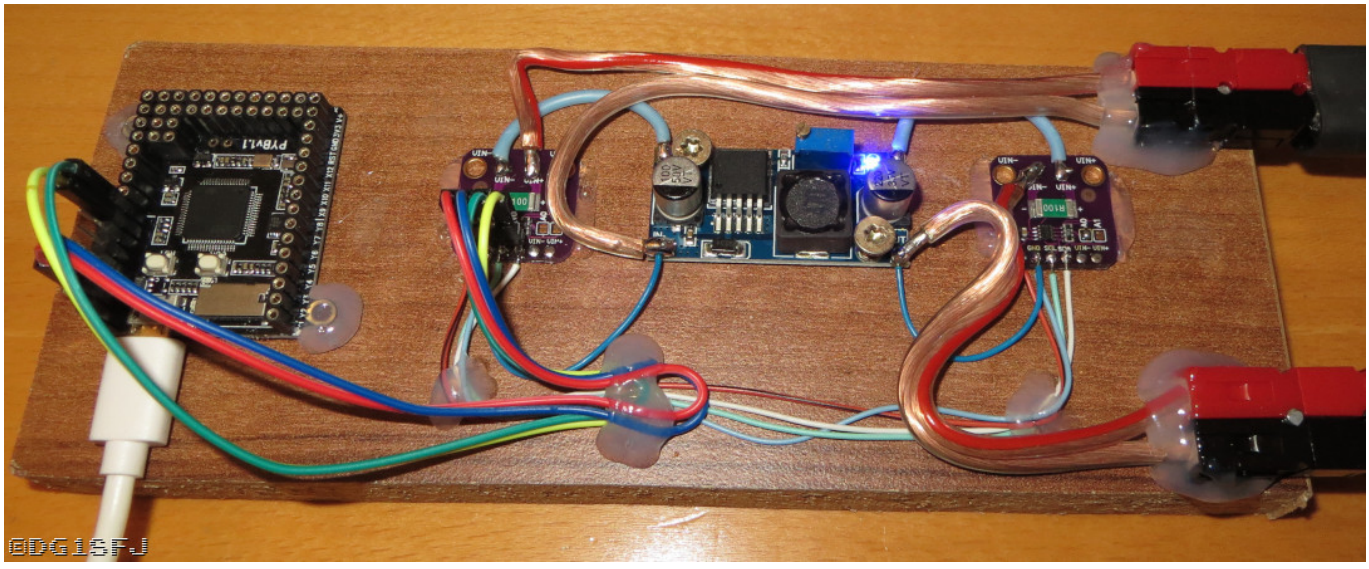
Messaufbau Wirkungsgrad DCDC Wandler

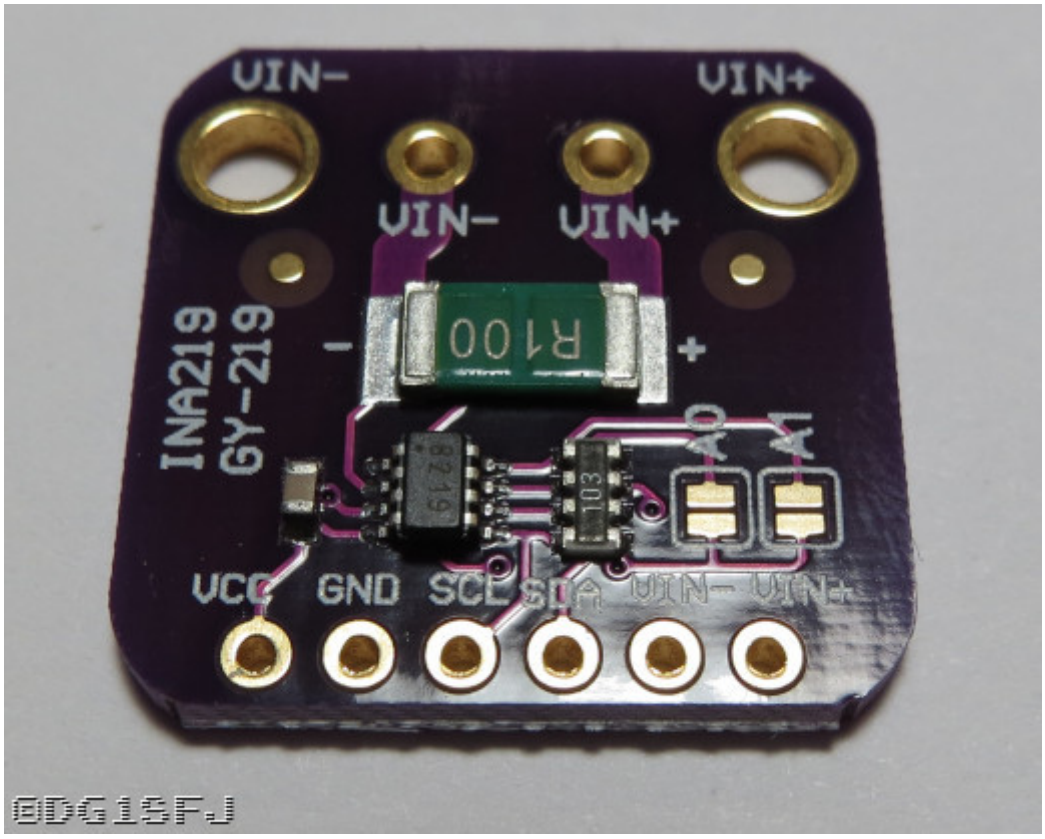
Messaufbau zum Wirkungsgrad eines DCDC Wandlers



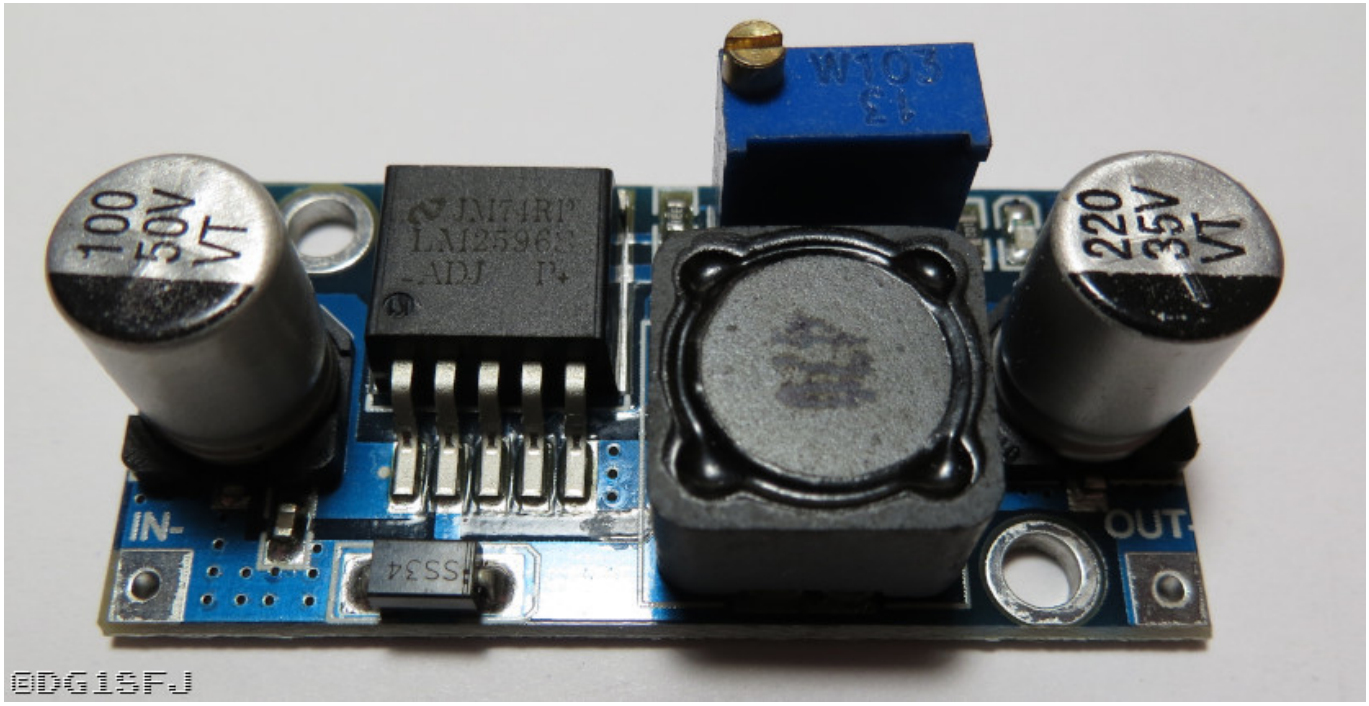
In den letzten Jahren habe ich viele der kleinen DCDC Wandler Fertigmodule eingesetzt. Aber welchen Wirkungsgrad haben diese kleinen Module ? 4 Messgeräte für Spannung/Strom anschließen wollte ich nicht und so war ich auf der Suche nach einer kleinen und schnellen Lösung. Die habe ich dann gefunden in Form der INA-219 Strommessmodule welche von einem Micropython Pyboard ausgelesen werden. Das ganze war dann auch gleich das erste Projekt in Python auf einem Microcontroller.



Auf der linken Seite das Micropython Pyboard1.1 mit USB Anschluss welches auf dem PC eine Serielle Schnittstelle und ein Flashdrive erzeugt. In der Mitte ein typisches DCDC Wandler Modul sowie links und rechts die INA Module.



Der Vorteile dieser Mini-Module ist das der INA219 mit Shunt fertig aufgelötet ist und man per I2C die Strom und die Spannungswerte auslesen kann. Mit den Lötbrücken rechts kann man noch die I2C Adresse verstellen wenn man mehrere Module zeitgleich am Bus hat.



Beispiel eines DCDC Wandlers (Buck) mit dem LM2596S. Die Dinger gibt es fertig für ein paar Euro in der Bucht.

Alles was jetzt noch fehlt ist ein kleines Python Programm welches Strom/Spannung am Ein und Ausgang ausliest und daraus die Werte Wirkungsgrad, Eingangsleistung, Ausgangsleistung, Verlustleistung, Eingangsspannung/Strom, Ausgangsspannung/Strom auf der seriellen Schnitte

ausgibt.

Quellcode anbei als erster Startpunkt, es enthält keine Abfragen auf negative Werte, Teilung durch 0 u.s.w. Das kann jeder noch selbst einprogrammieren wie er möchte.

Für meine Zwecke haben diese paar Zeilen schon voll ausgereicht :

```
from pyb import I2C
import ustruct
import time

i2c = I2C(2)                         # create on bus 2 (Pins auf der rechten
Seite nahe dem USB)
i2c = I2C(2, I2C.MASTER)             # create and init as a master
i2c.init(I2C.MASTER, baudrate=10000) # init as a master

while True:
    cur1 = (ustruct.unpack(">h", i2c.mem_read(2, 0x40, 1))[0])/10000
    vol1 = ((ustruct.unpack(">H", i2c.mem_read(2, 0x40, 2))[0] & 0xFFF8) >>
3)*0.004
    cur2 = (ustruct.unpack(">h", i2c.mem_read(2, 0x41, 1))[0])/10000
    vol2 = ((ustruct.unpack(">H", i2c.mem_read(2, 0x41, 2))[0] & 0xFFF8) >>
3)*0.004
    pow1 = vol1 * cur1
    pow2 = vol2 * cur2

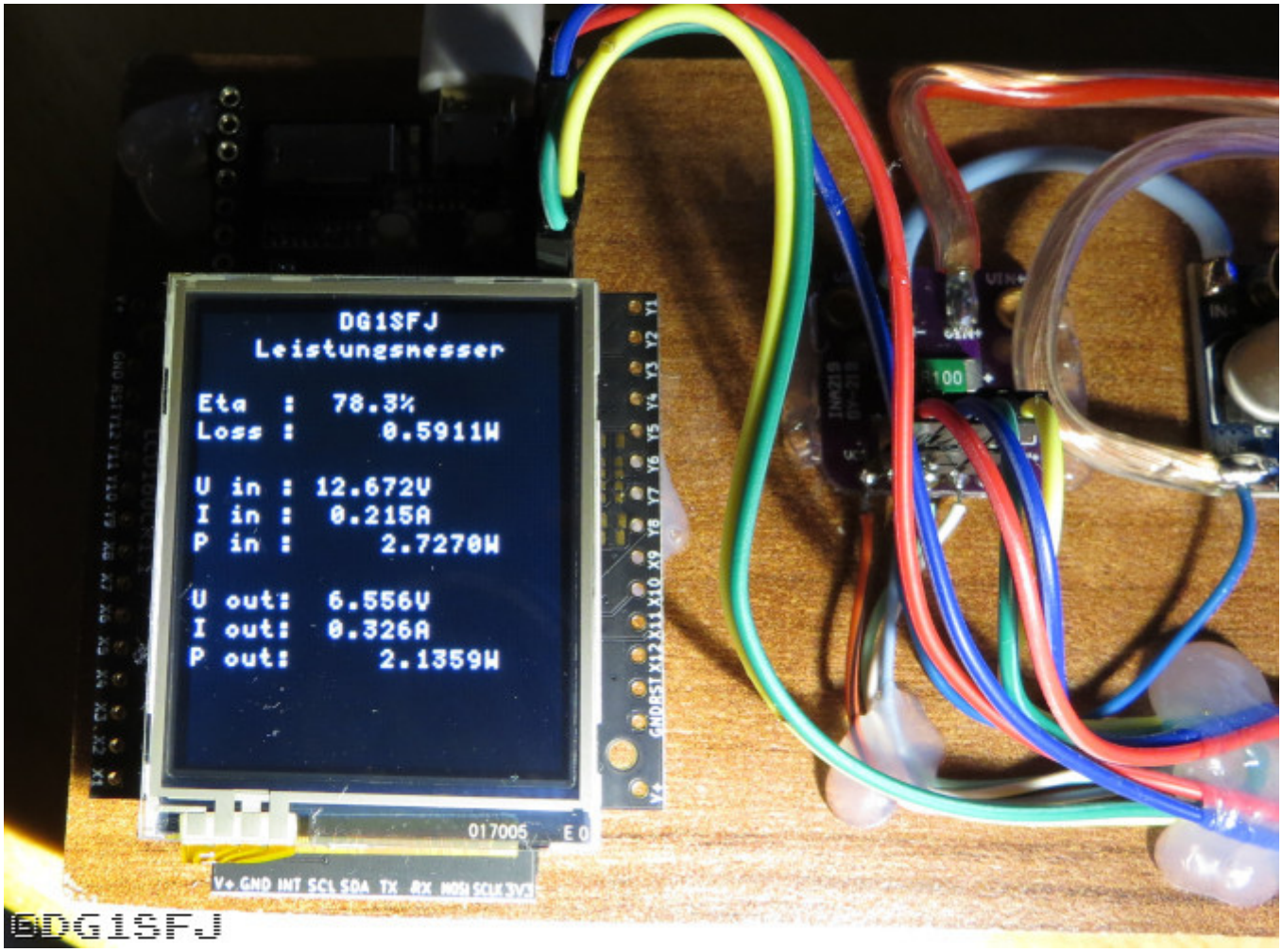
    etapro = 100/pow1*pow2
    loss = pow1-pow2

    print ("Eta: %5.1f%% | Loss: %10.4fW | In: %6.3fV %6.3fA %10.4fW | Out:
%6.3fV %6.3fA %10.4fW" % (etapro,loss,vol1,cur1,pow1,vol2,cur2,pow2))
    time.sleep(1.0)
```

Und so sieht die Ausgabe der Messwerte auf der seriellen Schnittstelle aus im Sekundentakt :

```
Datei  Bearbeiten  Ansicht  Suchen  Terminal  Hilfe
Eta:  77.1% | Loss:     0.5109W | In: 13.112V  0.170A     2.2290W | Out:  6.568V  0.262A     1.7182W
Eta:  77.4% | Loss:     0.5017W | In: 13.116V  0.169A     2.2205W | Out:  6.568V  0.262A     1.7188W
Eta:  76.6% | Loss:     0.5246W | In: 13.116V  0.171A     2.2441W | Out:  6.568V  0.262A     1.7195W
Eta:  77.2% | Loss:     0.5074W | In: 13.120V  0.170A     2.2265W | Out:  6.564V  0.262A     1.7191W
Eta:  77.0% | Loss:     0.5155W | In: 13.116V  0.171A     2.2402W | Out:  6.568V  0.263A     1.7248W
Eta:  77.8% | Loss:     0.4925W | In: 13.120V  0.169A     2.2160W | Out:  6.568V  0.262A     1.7234W
Eta:  76.8% | Loss:     0.5194W | In: 13.116V  0.171A     2.2428W | Out:  6.568V  0.262A     1.7234W
Eta:  77.5% | Loss:     0.5004W | In: 13.120V  0.170A     2.2238W | Out:  6.568V  0.262A     1.7234W
Eta:  76.8% | Loss:     0.5214W | In: 13.120V  0.171A     2.2461W | Out:  6.568V  0.263A     1.7248W
Eta:  78.2% | Loss:     0.4814W | In: 13.120V  0.168A     2.2055W | Out:  6.568V  0.263A     1.7241W
Eta:  78.2% | Loss:     0.4794W | In: 13.116V  0.168A     2.2035W | Out:  6.568V  0.263A     1.7241W
Eta:  76.9% | Loss:     0.5174W | In: 13.116V  0.171A     2.2441W | Out:  6.568V  0.263A     1.7267W
Eta:  76.6% | Loss:     0.5266W | In: 13.116V  0.172A     2.2507W | Out:  6.568V  0.263A     1.7241W
Eta:  77.7% | Loss:     0.4951W | In: 13.116V  0.169A     2.2219W | Out:  6.568V  0.263A     1.7267W
Eta:  78.3% | Loss:     0.4800W | In: 13.116V  0.168A     2.2074W | Out:  6.568V  0.263A     1.7274W
Eta:  77.0% | Loss:     0.5174W | In: 13.116V  0.171A     2.2455W | Out:  6.568V  0.263A     1.7280W
Eta:  76.9% | Loss:     0.5181W | In: 13.116V  0.171A     2.2468W | Out:  6.568V  0.263A     1.7287W
Eta:  77.9% | Loss:     0.4899W | In: 13.120V  0.169A     2.2186W | Out:  6.568V  0.263A     1.7287W
Eta:  77.1% | Loss:     0.5121W | In: 13.112V  0.171A     2.2408W | Out:  6.568V  0.263A     1.7287W
Eta:  78.0% | Loss:     0.4866W | In: 13.116V  0.169A     2.2140W | Out:  6.568V  0.263A     1.7274W
Eta:  76.9% | Loss:     0.5183W | In: 13.116V  0.171A     2.2468W | Out:  6.572V  0.263A     1.7284W
Eta:  76.7% | Loss:     0.5246W | In: 13.116V  0.172A     2.2520W | Out:  6.568V  0.263A     1.7274W
Eta:  77.1% | Loss:     0.5148W | In: 13.116V  0.171A     2.2441W | Out:  6.568V  0.263A     1.7294W
DG1SFJ
```

Zum Pyboard gibt es ein passendes Display, das LCD160CRv1.1H zu kaufen. Ein paar Zeilen Code dazu und schon ist das autarke Messgerät fertig :



Der Code ist nur unwesentlich länger. Wer will kann natürlich noch Grafik einfügen, alles Bunt machen u.s.w. :

```
from pyb import I2C
import ustruct
import time
import lcd160cr

lcd = lcd160cr.LCD160CR('X')
lcd.set_orient(lcd160cr.PORTRAIT)
lcd.set_text_color(lcd.rgb(255,255,255), lcd.rgb(0,0,0))
lcd.set_font(1)
lcd.erase()
lcd.set_pos(20,10)
lcd.write('Leistungsmesser')
lcd.set_pos(50,0)
lcd.write('DG1SFJ')

i2c = I2C(2)                         # create on bus 2 (Pins auf der rechten
Seite nahe dem USB)
i2c = I2C(2, I2C.MASTER)             # create and init as a master
i2c.init(I2C.MASTER, baudrate=10000) # init as a master

while True:
    cur1 = (ustruct.unpack(">h", i2c.mem_read(2, 0x40, 1))[0])/10000
    vol1 = ((ustruct.unpack(">H", i2c.mem_read(2, 0x40, 2))[0] & 0xFFF8) >>
3)*0.004
    cur2 = (ustruct.unpack(">h", i2c.mem_read(2, 0x41, 1))[0])/10000
    vol2 = ((ustruct.unpack(">H", i2c.mem_read(2, 0x41, 2))[0] & 0xFFF8) >>
3)*0.004
    pow1 = vol1 * cur1
    pow2 = vol2 * cur2

    etapro = 100/pow1*pow2
    loss = pow1-pow2

    print ("Eta: %5.1f%% | Loss: %10.4fW | In: %6.3fV %6.3fA %10.4fW | Out:
%6.3fV %6.3fA %10.4fW" % (etapro,loss,vol1,cur1,pow1,vol2,cur2,pow2))
    lcd.set_pos(0,30)
    lcd.write("Eta  : %5.1f%%" % (etapro))

    lcd.set_pos(0,40)
    lcd.write("Loss : %10.4fW" % (loss))

    lcd.set_pos(0,60)
    lcd.write("U in : %6.3fV" % (vol1))
    lcd.set_pos(0,70)
    lcd.write("I in : %6.3fA" % (cur1))
    lcd.set_pos(0,80)
    lcd.write("P in : %10.4fW" % (pow1))

    lcd.set_pos(0,100)
    lcd.write("U out: %6.3fV" % (vol2))
    lcd.set_pos(0,110)
```

```
    lcd.write("I out: %6.3fA" % (cur2))
    lcd.set_pos(0,120)
    lcd.write("P out: %10.4fW" % (pow2))

    time.sleep(1.0)
```

From:
https://dg1sfj.de/ - **dg1sfj.de**

Permanent link:
https://dg1sfj.de/doku.php?id=elektronik:selbstbau:messaufbau

Last update: **2025/01/17 15:47**