

## PPK2 Mod

Ziel des Projekts ist die Bereichserweiterung eines PPK2s, dem Power Profiling Kit von Nordic. PC Software und Hardware ist Open Source. Die Messbereichserweiterung soll die Auflösung im Nano-Ampere-Bereich erhöhen.

Das PPK2 erreicht seine hohe Dynamik über 5 Ranges die analog durchgeschaltet werden.

Der Range Switch wird durch Über- oder Unterspannung am Ausgang des Instrumentation Amplifiers ausgelöst.

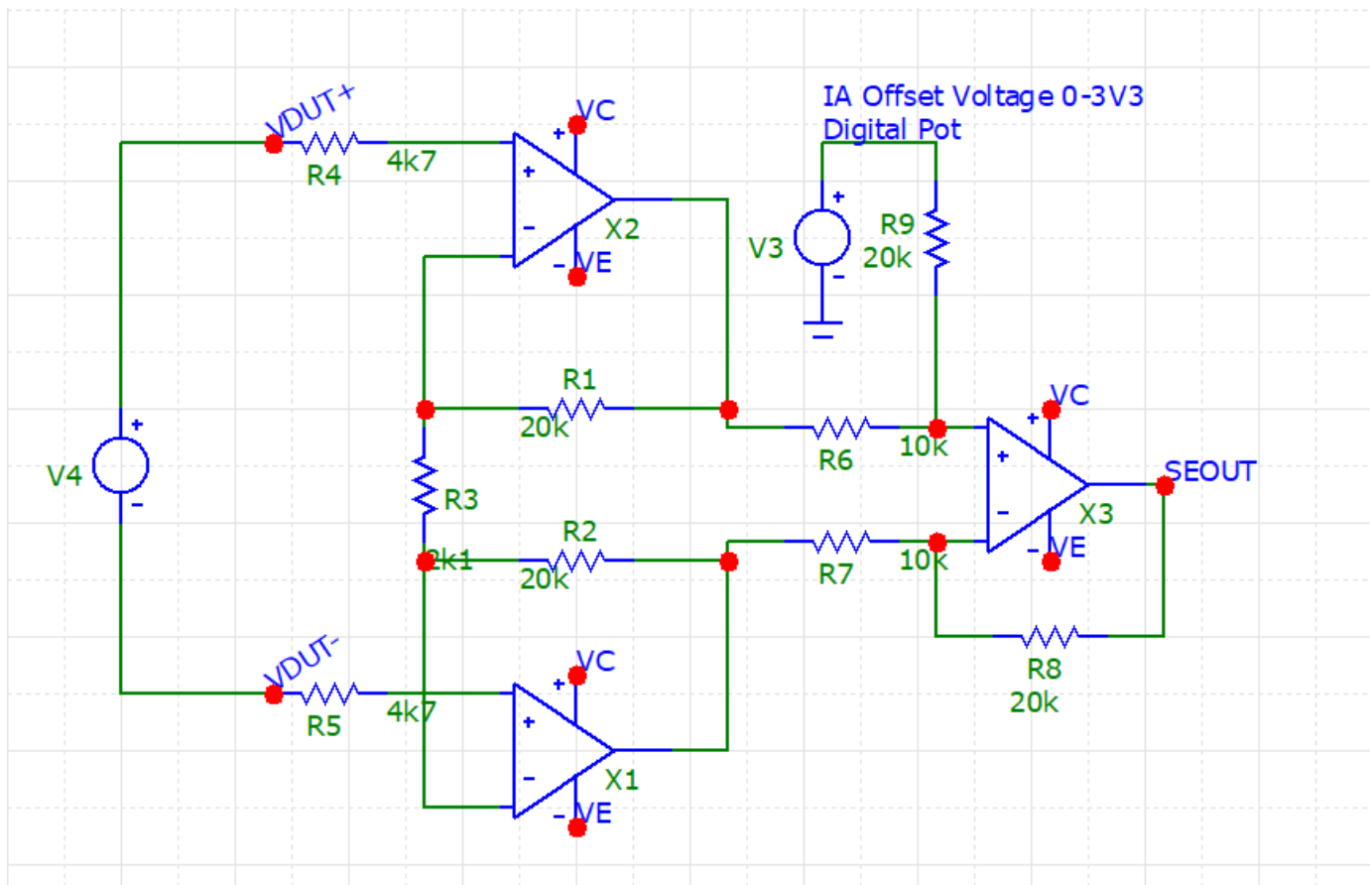
Da der Range Switch somit unabhängig von den jeweiligen Shunt-Widerständen schaltet ist eine Messbereichsverschiebung leicht möglich indem alle Shunt-Widerstände um eine Dekade erhöht werden.

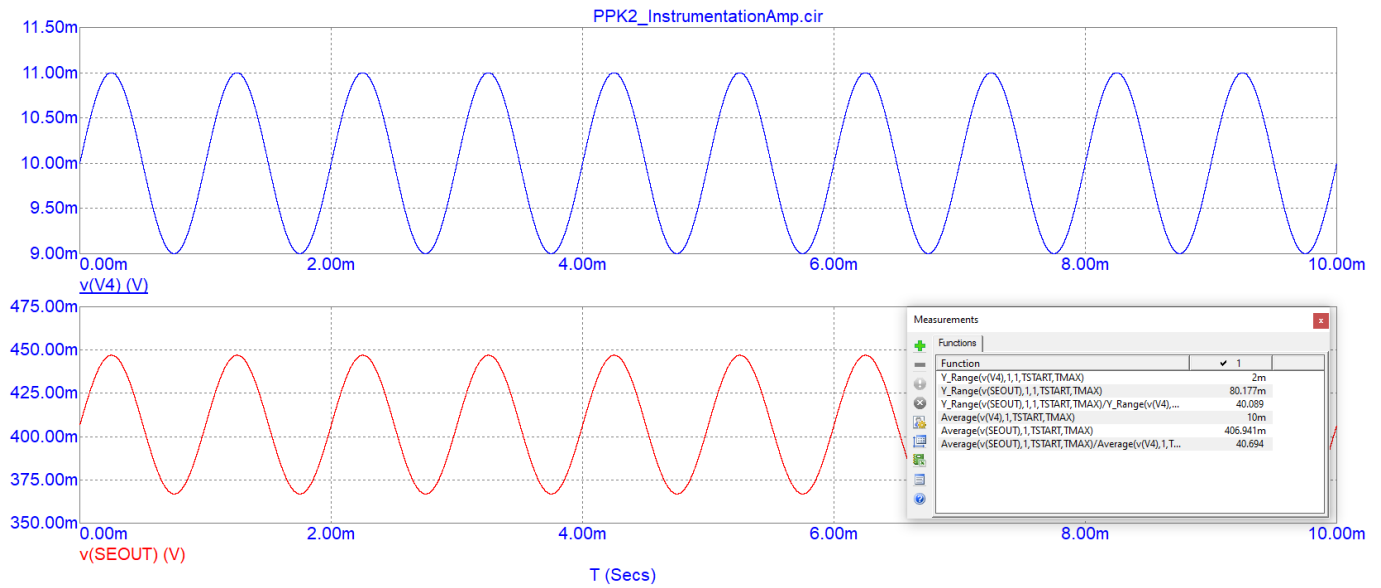
## Simulationen

...zur weiteren Evaluierung und zum Verständnis der weniger gut dokumentierten Schaltung

Simulationsprogramm: Microcap

## Instrumentation Amplifier

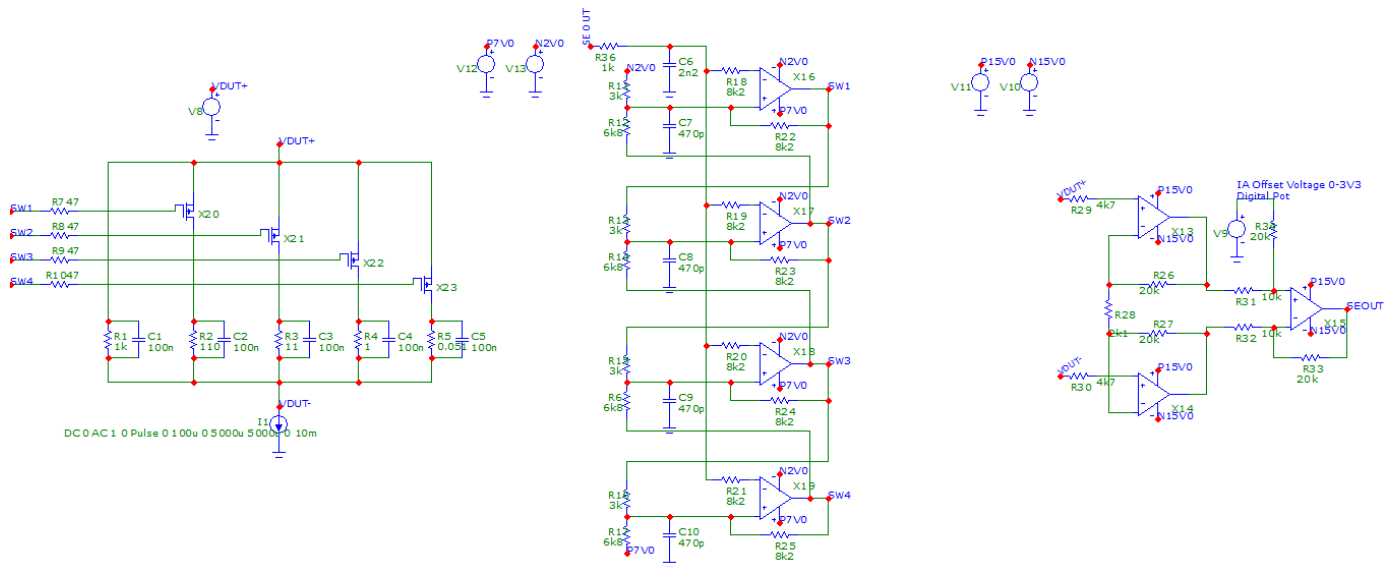


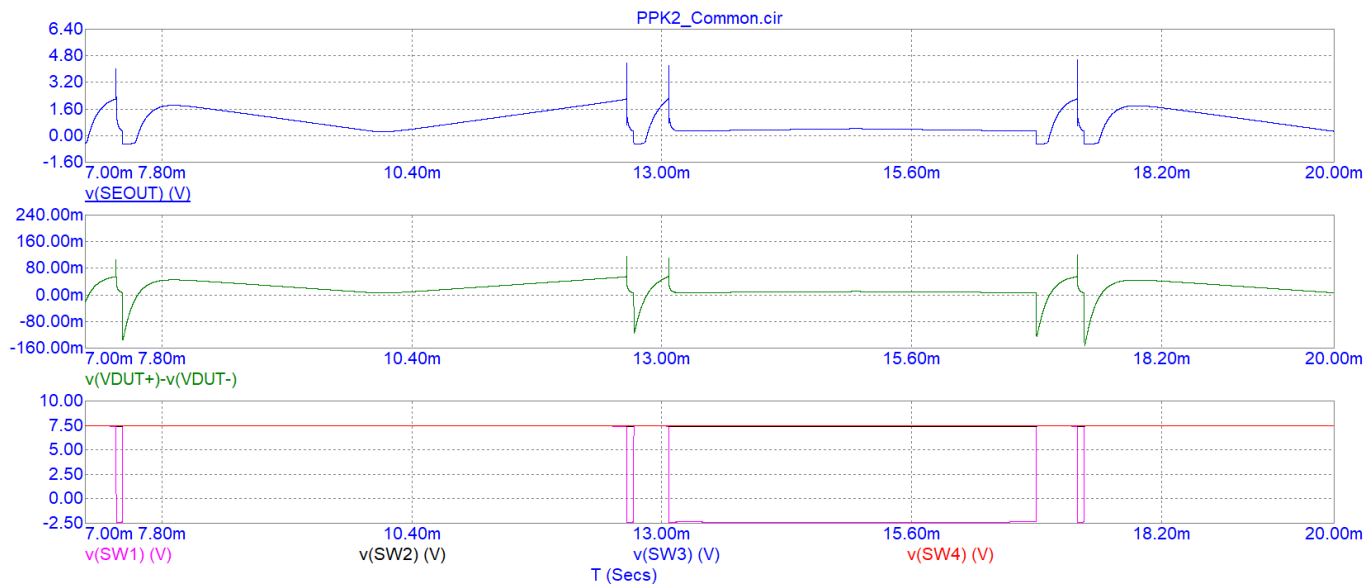


Simulierte Verstärkung: x40

Offset Voltage: Die Spannung am Offset Voltage Eingang wird 1:1 am Ausgang widergespiegelt.

### Range Switching





Die Simulation weist Instabilität auf. Die Schaltpunkte liegen bei 2.2V - 270mV (SEOUT) und somit nahe den errechneten:

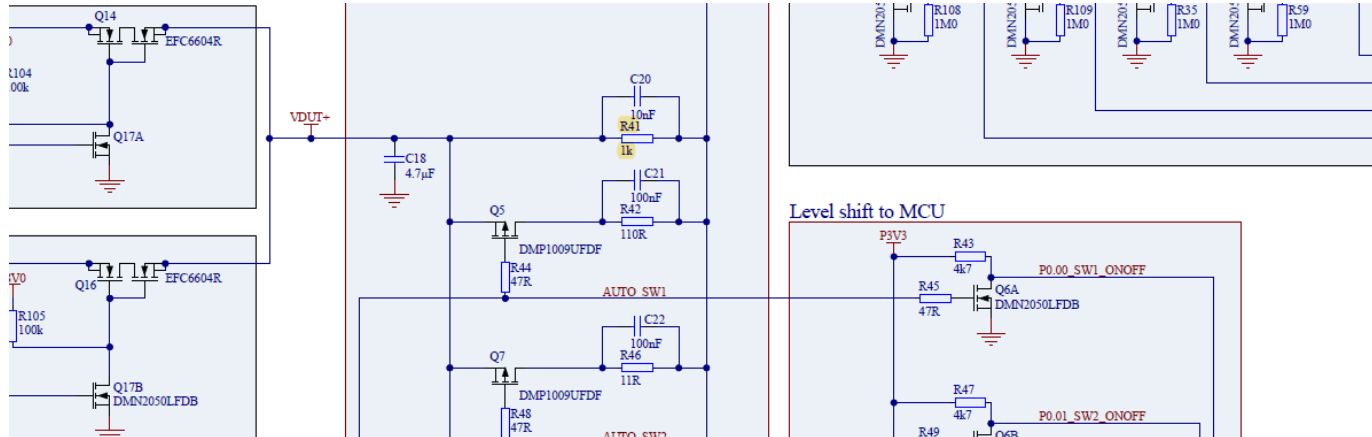
Range 1-3: 2V - 200mV

Range 4: 2.2V - 110mV

TODO: Simulation für Range Switch mit Bereichserweiterung.

## Tests

### Tests mit 10kOhm Shunt Widerstand



Der Shunt Widerstand R41 wird durch einen 10kOhm Widerstand ersetzt.

Umrechnung in Software:

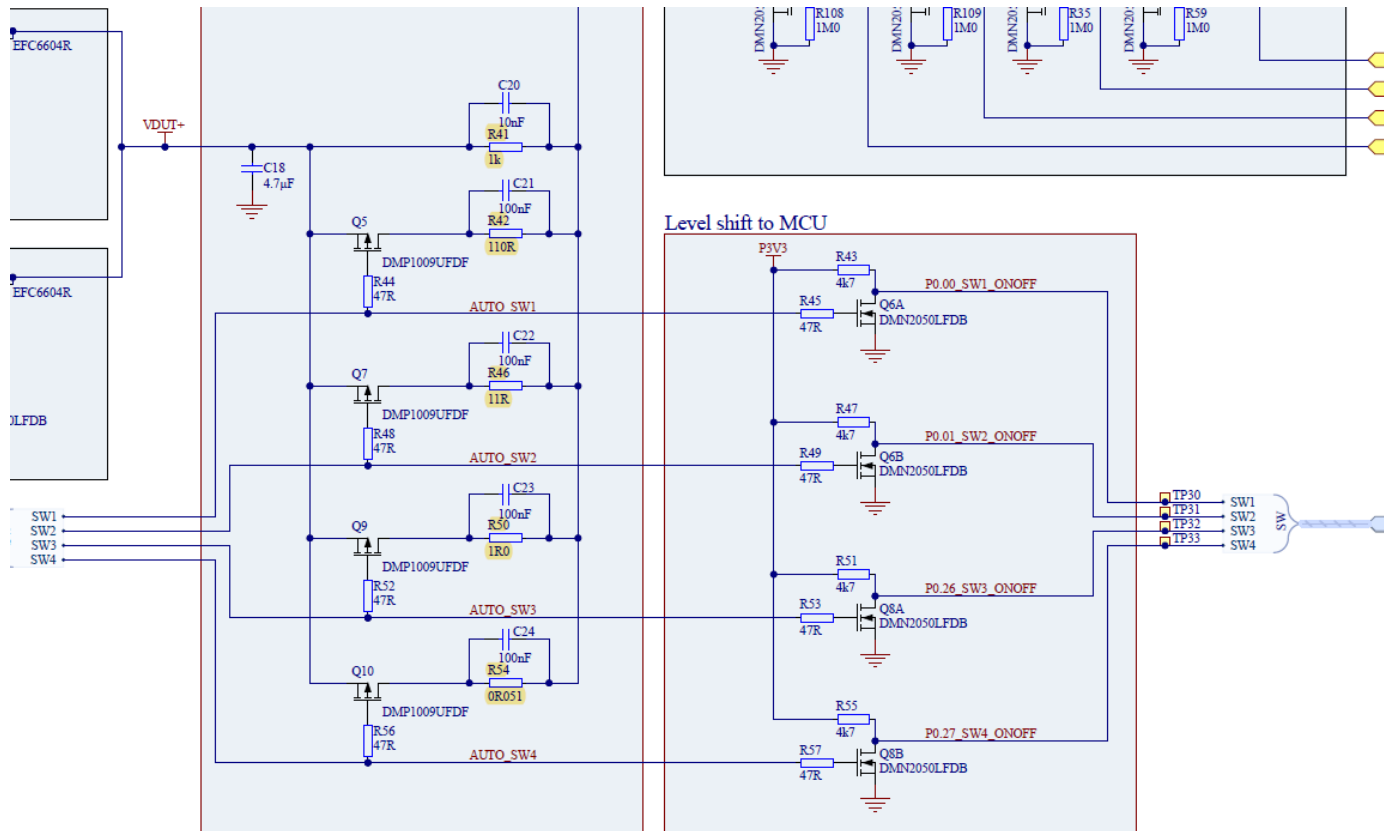
```
const resultWithoutGain =
  (adcVal - this.modifiers.o[range]) *
  (this.adcMult / (this.modifiers.r[range] * (range == 0 ? 10 : 1)));
let adc =
  this.modifiers.ug[range] *
  (resultWithoutGain *
    (this.modifiers.gs[range] * resultWithoutGain +
      this.modifiers.gi[range]) +
    (this.modifiers.s[range] * (this.currentVdd / 1000) +
      this.modifiers.i[range]));
```

Umrechnung: Abfrage ob der PPK2 in der niedrigsten Range ist, wenn ja muss der Shunt Widerstand mit 10 multipliziert werden.

Resultat: Durch fehlende Kalibration und 1% Widerstandstoleranz ist das Ergebnis im richtigen Bereich aber nicht exakt.

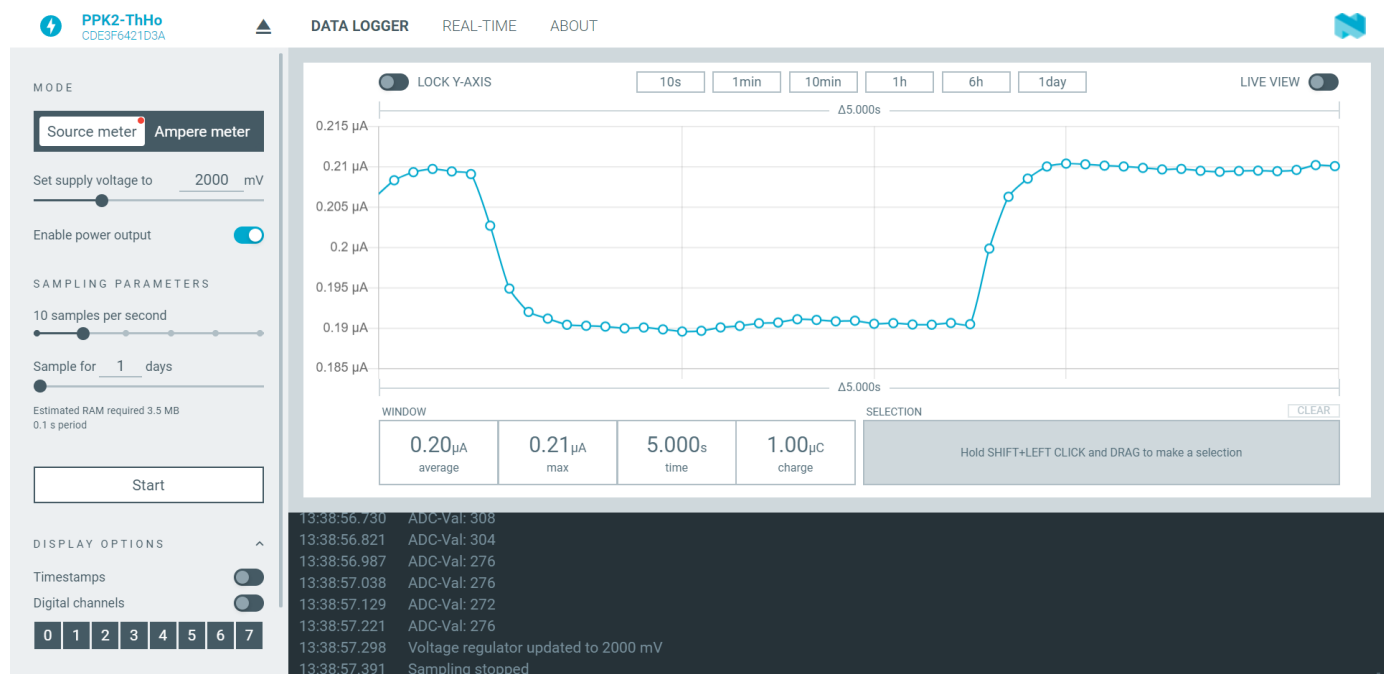
Bei ungünstigen Lastströmen begibt sich der analoge Range Switch in eine Oszillation da die vorhandene Hysterese nicht mehr ausreicht um den Messbereich über 2 Dekaden hinweg stabil zu halten.

### Tests mit Messbereichsversatz

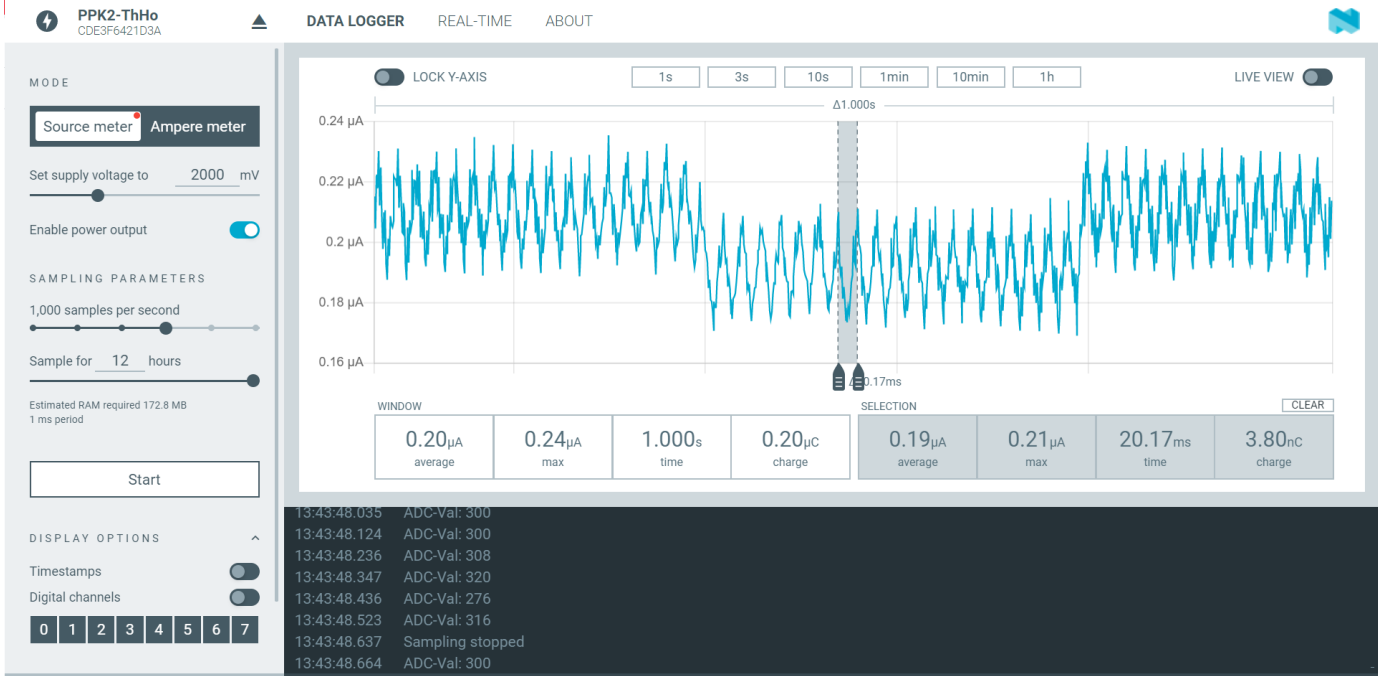


R41,R42,R46,R50,R54 werden jeweils um eine Dekade (Faktor x10) erhöht. Dadurch verschiebt sich der Messbereich. Das PPK2 verliert dadurch seine High Current Capabilities, gewinnt jedoch Auflösung im nA Bereich.

Resultat:



Die obige Abbildung zeigt einen Lastwechsel von 11Ohm auf 10MOhm bei einer Versorgungsspannung von 2V. Anmerkung: Samplingrate 10Samples/s Starkes Averaging da sonst Netzbrummen anliegt.



Hier liegt die Samplerate bei 1000 Samples pro Sekunde. Deutliches Netzbrummen (Periode 20ms) liegt vor

Im Durchschnitt errechnet der Power Profiler dennoch 19uA sowie 21uA Laststrom.



Die obige Abbildung zeigt einen Lastsprung von 11MOhm auf 1MOhm.

Lastwiderstand	Strom PPK2 mit Bereichsversatz	Strom PPK2 Default	Strom Advantest TR6848
11MOhm	190nA	260nA	189nA
1MOhm	2062nA	2060nA	2050nA

Anmerkung: Bei einer Last von 11MOhm schwankt der Wert des Default PPK2s bereits ab der 2ten Kommastelle (0.25-0.22uA). Der Bereichsverschobene PPK2 schwankt lediglich um die 3te Kommastelle (0.196 - 0.194uA).

Da durch die vorherigen Tests eine Bereichserweiterung sinnvoll erscheint wird diese in den nächsten Schritten durchgeführt und getestet.

## Bereichserweiterung 1A-10nA

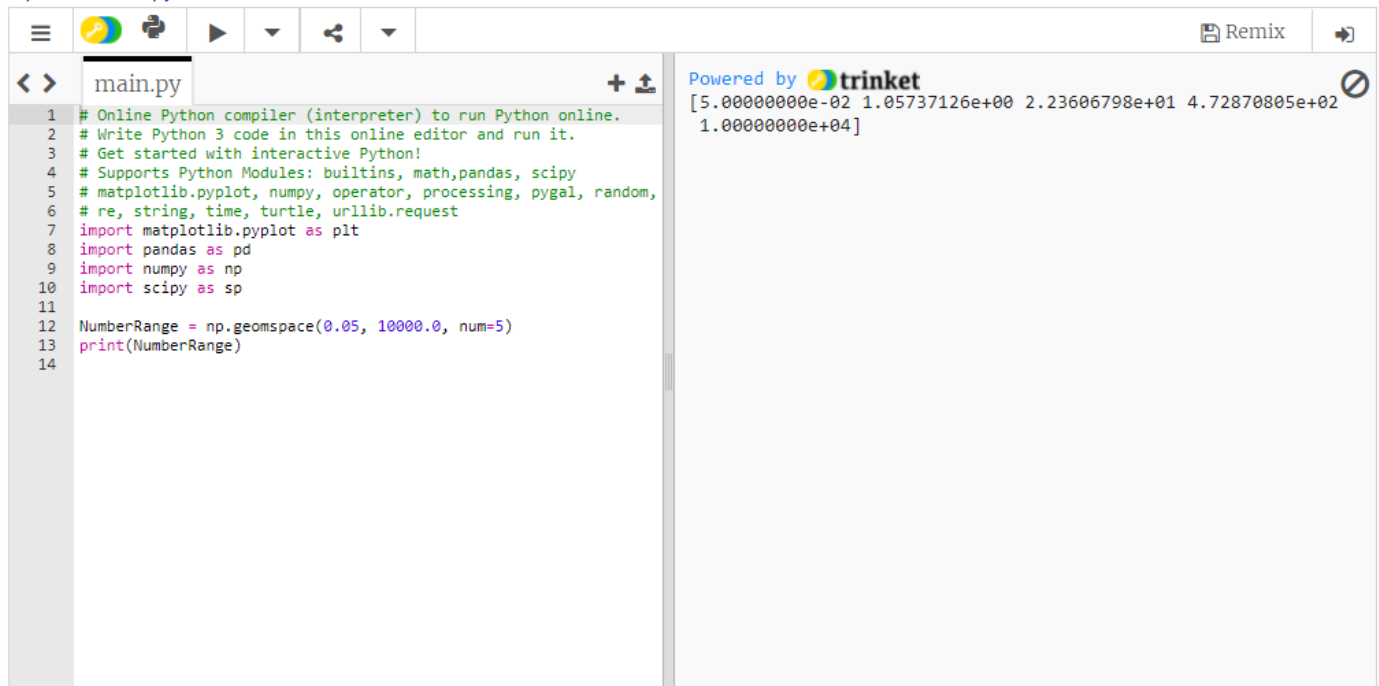
### Hardwareänderungen

Da die Messbereiche vorwiegend von den Shunt-Widerständen abhängen können diese einfach ausgetauscht werden indem man alle Widerstandswerte x10 multipliziert. Dadurch wird der Messbereich verschoben. Da das gewünschte Ergebnis jedoch eine Messbereichserweiterung ist muss hier der kleinste Shunt Widerstand gleich bleiben. Der kleinste Messbereich wird um einen Faktor 10 verkleinert.

Die Bereiche dazwischen müssen nun anders aufgeteilt werden. Eine logarithmische Verteilung ist notwendig:

Dimensionierung der Bereich mit Python per Geospace (NumPy):

<https://trinket.io/python3/8aac81c5b8>



```
1 # Online Python compiler (interpreter) to run Python online.
2 # Write Python 3 code in this online editor and run it.
3 # Get started with interactive Python!
4 # Supports Python Modules: builtins, math,pandas, scipy
5 # matplotlib.pyplot, numpy, operator, processing, pygal, random,
6 # re, string, time, turtle, urllib.request
7 import matplotlib.pyplot as plt
8 import pandas as pd
9 import numpy as np
10 import scipy as sp
11
12 NumberRange = np.geospace(0.05, 10000.0, num=5)
13 print(NumberRange)
14
```

Powered by  trinket

[5.00000000e-02 1.05737126e+00 2.23606798e+01 4.72870805e+02 1.00000000e+04]

Daraus ergeben sich folgende Logarithmisch verteilten Widerstandswerte:

50mOhm, 1Ohm, 22Ohm, 470Ohm, 10kOhm

Herstellernummern:

ERJL03KF50MV
ERA3APB471V
RT0603DRD0710KL
CRCW06031R00FKEAHP
ERJ3RED22R0V

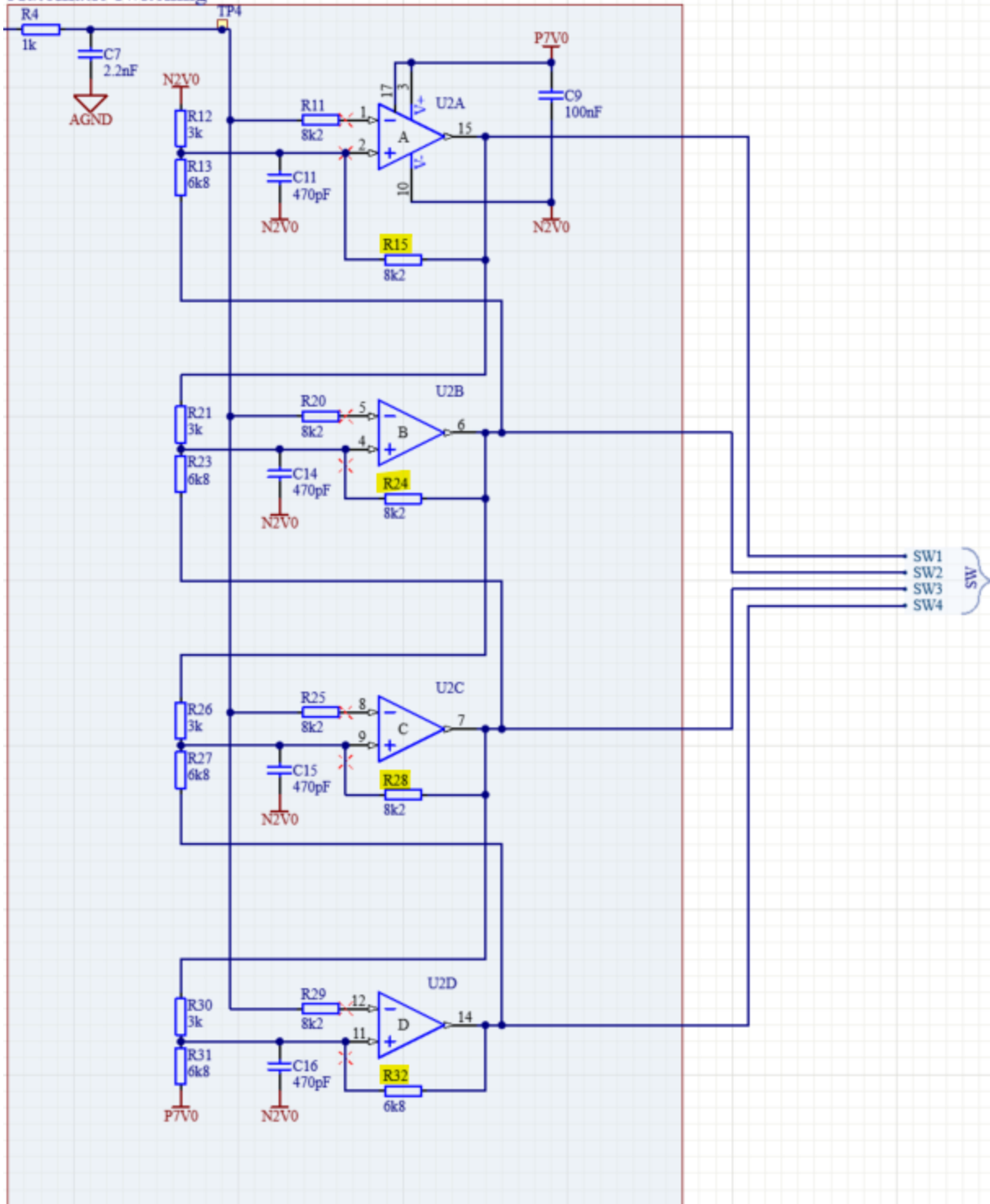
Widerstandsgehäuse: 0603, Original-Footprint: 0402

Größere Widerstandsgehäuse haben von Natur aus weniger Eigenerwärmung und sind somit besser als Shunt-Widerstände geeignet.

Problematik: Mögliche kalte Lötstellen da die Pads nicht gut erreichbar sind.

Problematik: Durch die höheren Messbereiche ergeben sich auch andere Spannungspegel am Ausgang des Instrumentation Amplifiers. Der Automatic Range Switch schaltet dadurch nicht mehr richtig und begibt sich bei einigen Stromwerten zwischen den Bereichen in eine Oszillation. (Starkes Negatives Feedback wie bereits in der Simulation zu sehen)

## Automatic switching



Die Widerstände R15, R24, R28, R32 bestimmen die Hysterese und somit auch die Schaltschwellen. Kleiner Widerstandswerte an diesen Stellen ergeben eine höhere Hysterese und somit einen weiteren Bereich bis zur nächsten Schaltschwelle (Hysterese durch verschieben der Referenzspannung an  $V_{in+}$ ).

Dimensionierung: Trial and Error. Durch langsames herantasten (inkrementelles verkleinern der Widerstandswerte) wurden die Widerstände R15, R24, R28, R32 auf 6k2 festgelegt.

Eine weitere Verkleinerung dieser Widerstände bewirkt dass Ranges nicht mehr bzw. zu Spät schalten und somit der ADC oben oder unten an seine Eingangsspannungsgrenzen kommt.

Durch die neuen Range-Hysterese Widerstände ergeben sich ADC-Counts von 150Cnts bis 6000Cnts (ADC Counts nach Gain-Correction, Spannungswerte nahe Aussteuerungen, 14Bit ADC)

## Softwareänderungen

Die meiste Funktionalität des Geräts wird durch die Software erreicht. Die Firmware liefert lediglich raw ADC Counts sowie die derzeitige Range und beim Verbinden des Geräts die ab-Werk Kalibrationsdaten.

Da die Kalibrationsdaten (auf einem I2C EEPROM auf dem Gerät gespeichert) nicht mit den neuen Widerständen und Bereichen kompatibel sind, muss die Kalibration neu durchgeführt werden.

Das originale Kalibrationsmodell ist unübersichtlich und nicht definierbar. Es bleibt unklar welche Korrekturen hier vorgenommen werden und wie die Faktoren berechnet werden.

Abhilfe: Neues, transparentes Kalibrationsmodell:  
9 Punkt (pro Range) Kalibration über Eingangsspannung und Strom.

Kalibrationsdaten der Range 0:

```
//Range 0 0nA-10nA-3uA-5uA
range0CalData = [
  { x: 0.8, y: 10.0e-9, z: 99.50e-9 },
  { x: 3.3, y: 10.0e-9, z: 129.5e-9 },
  { x: 5.0, y: 10.0e-9, z: 121.5e-9 },
  { x: 0.8, y: 220.0e-9, z: 301.5e-9 },
  { x: 3.3, y: 220.0e-9, z: 331.2e-9 },
  { x: 5.0, y: 220.0e-9, z: 322.7e-9 },
  { x: 0.8, y: 5.0e-6, z: 5.086e-6 },
  { x: 3.3, y: 5.0e-6, z: 5.112e-6 },
  { x: 5.0, y: 5.0e-6, z: 5.113e-6 }];
```

x: DUT Spannung in V

y: Referenz Strom von bekannter und geeichter Stromquelle

z: Ausgangs Strom des PPK2s ohne Kalibration (Offset = 0 wenn y=z)

Diese Kalibrationsdaten werden bilinear Interpoliert um eine bessere Abdeckung der Range zu erreichen.

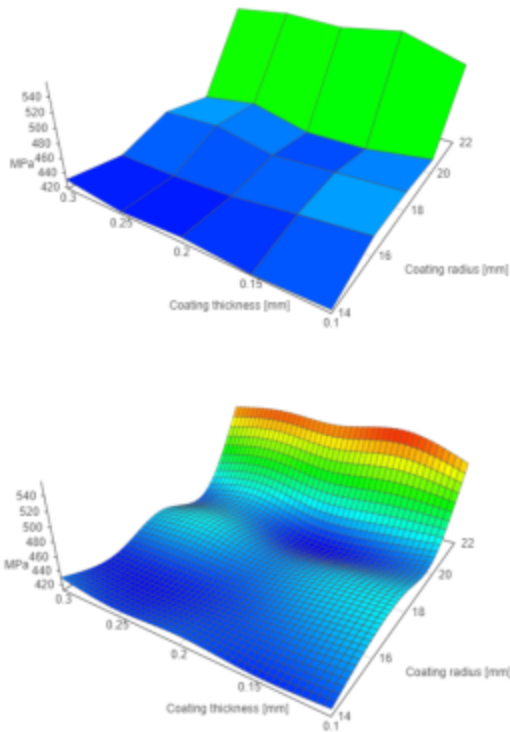
Interpolation:

```
//Interpolate (bilinear) the calibration Data in order to provide more data points for offset correction
this.calibrationDataSet[0] = interpolateArray(this.range0CalData, CALIBRATION_INTERPOLATION_DEGREE);
this.calibrationDataSet[1] = interpolateArray(this.range1CalData, CALIBRATION_INTERPOLATION_DEGREE);
this.calibrationDataSet[2] = interpolateArray(this.range2CalData, CALIBRATION_INTERPOLATION_DEGREE);
this.calibrationDataSet[3] = interpolateArray(this.range3CalData, CALIBRATION_INTERPOLATION_DEGREE);
this.calibrationDataSet[4] = interpolateArray(this.range4CalData, CALIBRATION_INTERPOLATION_DEGREE);

//Convert from calibration data to offset correction data (Reference-Measured Current)
for (let j = 0; j < this.calibrationDataSet.length; j++) {
  for (let i = 0; i < this.calibrationDataSet[j].length; i++) {
    let tmp = this.calibrationDataSet[j][i].y;
    this.calibrationDataSet[j][i].y = this.calibrationDataSet[j][i].z;
    this.calibrationDataSet[j][i].z = tmp - this.calibrationDataSet[j][i].z;
  }
}

//Log the interpolated calibration data to the console
console.log(this.calibrationDataSet);
```





Achtung: Abbildungen nur Symbolbilder, repräsentieren nicht die tatsächlichen Kalibrationsdaten

Durch diese Art der Kalibration ergibt sich ein Abbild des Gesamten Messbereichs.

Umrechnen von ADC Counts zu Strom:

```
//Calculate Voltage at ADC Input from Reference Voltage and ADC Counts and multiply by 2 (Gain Correction from Firmware)
let vADC = (2.0 * adcVal * this.adcMult);
//Calculate Current from Voltage at the ADC (times 2 bc. of 1:1 voltage divider) div. by 40 (InstAmp Gain) and divide by shunt Res.
let iADC = ((2.0 * vADC) / 40.0) / (this.modifiers.r[range]);

//Find the nearest Point in the interpolated calibration Matrix
let offset = getNearestPoint(this.calibrationDataSet[range], { x: (this.currentVdd / 1000), y: iADC });
//Apply the offset
let adc = iADC + offset.z;
```

Der Offset wird als der nächste Wert in der Kennlinie (kleinster Abstand über Pythagoras) angenommen.

Weitere Änderungen der Software:

- Debug Statements im Logger der Software (Betriebsspannung, Range, Widerstand, Offset, etc.)
- Erhöhung der Kommastellen der Stats Anzeigen (Average, Maximum und Charge)

### Kalibrationsvorgang

1. Kalibrationsdatentabell in "src\device\serialDevice.js" für die entsprechende Range editieren:
  - a. Gewünschte DUT Spannungen eintragen
  - b. Gewünschte Referenz-Ströme eintragen
  - c. Ausgangsströme gleich den Referenz-Strömen setzten ( $y=z \rightarrow \text{Offset} = y-z = 0$ )
2. Die Software neu Kompillieren. Kompletter Guide zum aufsetzen und kompillieren eines Node.js Projekts unter: [https://nordicsemiconduct.org.github.io/pc-nrfconnect-docs/app\\_development](https://nordicsemiconduct.org.github.io/pc-nrfconnect-docs/app_development)
3. Messtabelle durchgehen
  - a. DUT Spannung einstellen
  - b. Strom bei Stromquelle einstellen (als Stromsenke!!!)
  - c. Warten bis sich das Average-Reading eingestellt hat
  - d. Average Current aus der Software nRF Connect in die Tabelle übertragen
4. Code abspeichern und nRF Connect Power Profiler neu kompillieren

Die Referenz - Daten wurden:

- für die DUT Spannung Linear aufgeteilt (0.8V = Min, 3.3V, 5V = Max)
- für den Eingangsstrom geometrisch gemittelt (Range-Min, geom. Mittel der Range, Range-Max)

Der Interpolationsgrad wird über die Konstante `CALIBRATION_INTERPOLATION_DEGREE` eingestellt.

Weiters müssen die richtigen Widerstandwerte eingetragen werden ("src\device\serialDevice.js:150")  
Hier reichen die idealen Werte, der Rest wird durch die Kalibration korrigiert.