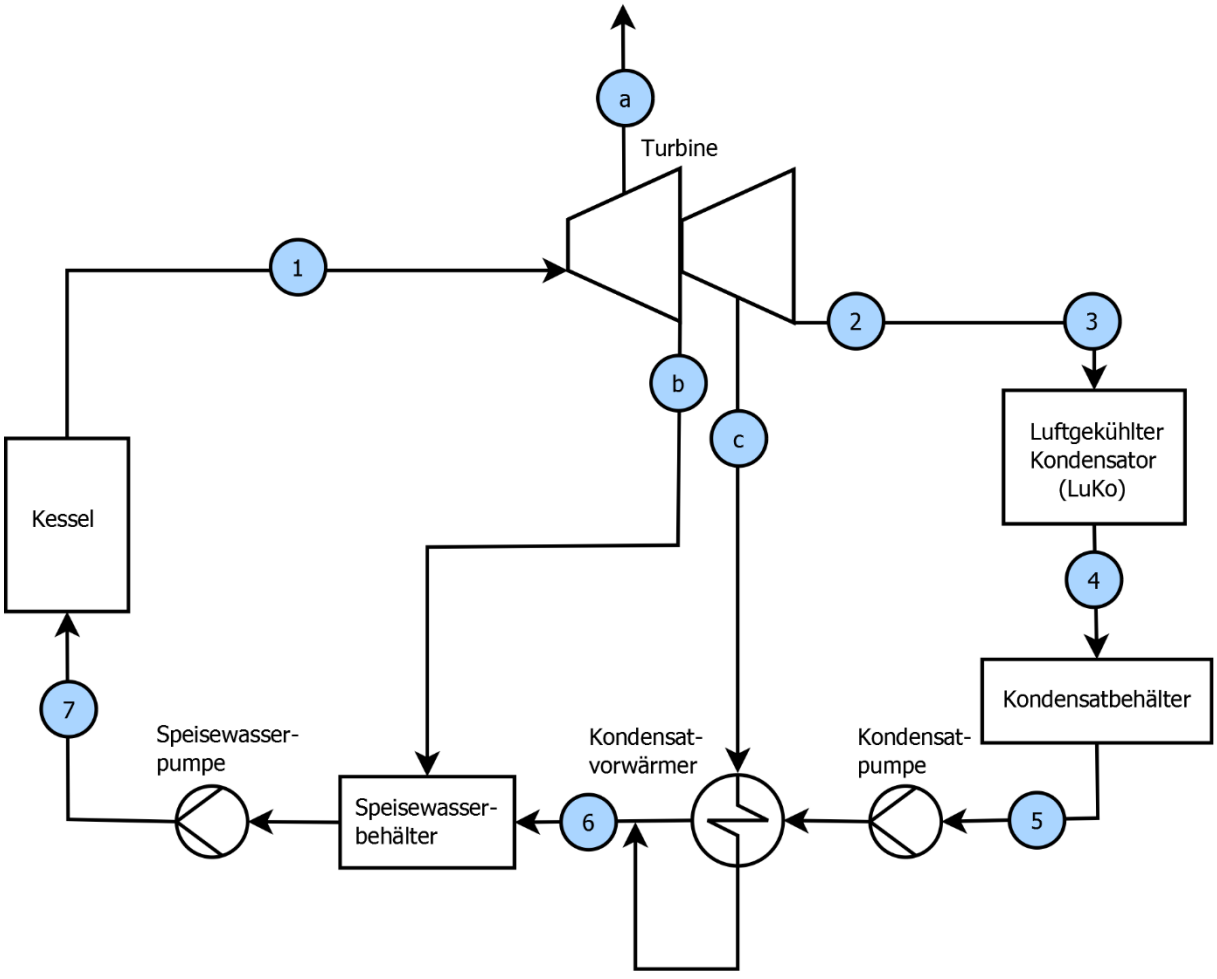
mycon GmbH
Senner Str. 156
D-33659 Bielefeld
Tel. +49 521 40 30 90
Fax +49 521 40 24 82
www.mycon-germany.com
info@mycon-germany.com

Berechnung des Gütefaktors mittels VBA

In diesem Dokument wird der Quellcode zur Berechnung des Gütefaktors von Luftgekühlten Kondensatoren (kurz LuKos) in Dampfkraftanlagen mit Hilfe von ***Visual Basic for Applications*** (VBA) und Microsoft-Excel vorgestellt.

Für die Auswertung werden folgende Messwerte gebraucht:



Messpunkt	Bezeichnung des Messwerts	Einheit
1	Temperatur Turbineneintritt	°C
	Druck Turbineneintritt	bar
	Dampfmassenstrom Turbineneintritt	t/h
a	Temperatur Mitteldruckdampf	°C
	Druck Mitteldruckdampf	bar
	Massenstrom Mitteldruckdampf	t/h

b	Temperatur Entnahme	°C
	Druck Entnahme	bar
c	Temperatur Anzapfung	°C
	Druck Anzapfung	bar
2	Temperatur Turbinenaustritt	°C
	Druck Turbinenaustritt	bar
3	Temperatur LuKo-Eintritt	°C
4	Temperatur LuKo-Austritt	°C
5	Massenstrom Kondensat	t/h
6	Temperatur nach Kondensatvorwärmer	°C
7	Temperatur Kesseleintritt	°C
	Druck Kesseleintritt	bar
Zusätzlich	Umgebungstemperatur (Außenluft)	°C
	Druck im Speisewasserbehälter	bar
Ventilatoren	Wert, über den sich die Leistung der Ventilatoren beurteilen lässt	A|kW|%

Für die Bilanzierung werden Funktionen benötigt, die Druck, Temperatur, Enthalpie und Entropie ins Verhältnis setzen können. Diese liefern die empirischen Funktionen des Industriestandards IAPWS-IF97 [1]. Die Implementierung in VBA wird durch folgenden Quellcode realisiert:

```
1.  'Copyright <2020> <mycon GmbH>
2.  '<author: P. Modler>
3.
4.  'Permission is hereby granted, free of charge, to any person
    obtaining a copy of this software and associated documentation files
    (the "Software"), to deal in the Software without restriction,
    including without limitation the rights to use, copy, modify, merge,
    publish, distribute, sublicense, and/or sell copies of the Software,
    and to permit persons to whom the Software is furnished to do so,
    subject to the following conditions:
5.
6.  'The above copyright notice and this permission notice shall be
    included in all copies or substantial portions of the Software.
7.
8.  'THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
    EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
    MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
    NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
    BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
    ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
    CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE.
9.
10.
11.
```

```
12.  'In diesem Modul befinden sich die Funktionen, die die Wasser- und
13.  'Dampfzustände berechnen.
14.  'Die Berechnung wird auf Grundlage des Industriestandards IAPWS-
     IF97 durchgeführt. (Empirisch)
15.  '
16.  'Berechnet werden kann:
17.  '
18.  '-Die Enthalpie abhängig von Druck und Temperatur im Wassergebiet:
19.  '   hWas(T,p)
20.  '-Die Enthalpie abhängig von Druck und Temperatur im Dampfgebiet:
21.  '   hDam(T,p)
22.  '-Die Entropie abhängig von Druck und Temperatur im Dampfgebiet:
23.  '   sDam(T,p)
24.  '
25.  '-Der Kondensationsdruck abhängig von der Temperatur:
26.  '   KonDru(T)
27.  '-Die Kondensationstemperatur abgängig vom Druck:
28.  '   KonTem(p)
29.  '-Die Enthalpie auf der Sattdampfkurve abhängig von der
30.  ' Kondensationstemperatur im Kondensatzustand:
31.  '   hWasKon(T)
32.  '-Die Enthalpie auf der Sattdampfkurve abhängig von der
33.  ' Kondensationstemperatur im Dampfzustand:
34.  '   hWasDam(T)
35.  '-Die Entropie auf der Sattdampfkurve abhängig von der
     Kondensationstemperatur im Kondensatzustand:
36.  '   sWasKon(T)
37.  '-Die Entropie auf der Sattdampfkurve abhängig von der
     Kondensationstemperatur im Dampfzustand:
38.  '   sWasDam(T)
39.  '
40.  '-Der Druck abhängig von Temperatur und Enthalpie im Wassergebiet:
41.  '   pvonhTWas(h, T, Optional Startwert, Optional Schrittweite)
42.  '
43.  '-Die Temperatur abhängig von Druck und Enthalpie im Wassergebiet:
44.  '   TvonhpWas(h, p, Optional Startwert, Optional Schrittweite)
45.  '
46.  '-Der Druck abhängig von Temperatur und Enthalpie im Dampfgebiet:
47.  '   pvonhTDam(h, T, Optional Startwert, Optional Schrittweite)
48.  '
49.  '-Die Temperatur abhängig von Druck und Enthalpie im Dampfgebiet:
50.  '   TvonhpDam(h, p, Optional Startwert, Optional Schrittweite)
51.  '
52.  '-Der Druck abhängig von Temperatur und Entropie im Dampfgebiet:
53.  '   pvonsTDam(h, T, Optional Startwert, Optional Schrittweite)
54.  '
55.  '-Die Temperatur abhängig von Druck und Entropie im Dampfgebiet:
56.  '   TvonspDam(h, p, Optional Startwert, Optional Schrittweite)
57.  '
58.  '
59.  '-Der Dampfgehalt abhängig von Enthalpie und Temperatur
60.  '   xvonhT(h,T)
61.  '-Der Dampfgehalt abhängig von Entropie und Temperatur
62.  '   xvonsT(s,T)
```

```
'
'-Die Enthalpie abhängig von Dampfgehalt und Temperatur
'   hvonxT(x,T)
'-Die Entropie abhängig von Dampfgehalt und Temperatur
'   svonxT(x,T)
'
'!!!Die Temperaturen werden immer in [°C]  !!!
'!!!der Druck in [bar]                     !!!
'!!!und die Enthalpie in [kJ/kg] angegeben.!!!

   Dim p0, T0, k, m As Double
   Dim b(6), c(6), d(6), e(6), F(6), G(6), km(6) As String
   Dim y, A2, A3 As Double
   Dim StartH As Double
   Dim kr As Double



   Const y1 = 0.21

   Const AbweichungHpDam = 0.00001    'Gibt vor um wie viel [%] die
Enthalpie bei der Iteration abweichen darf
   Const AbweichungHTDam = 0.001    'Gibt vor um wie viel [%] die
Entropie bei der Iteration abweichen darf

   Const AbweichungHpWas = 0.0001    'Gibt vor um wie viel [%] die
Enthalpie bei der Iteration abweichen darf
   Const AbweichungHTWas = 0.01    'Gibt vor um wie viel [%] die
Entropie bei der Iteration abweichen darf

   Const AbweichungSp = 0.001    'Gibt vor um wie viel [%] die
Enthalpie bei der Iteration abweichen darf
   Const AbweichungST = 0.001    'Gibt vor um wie viel [%] die
Entropie bei der Iteration abweichen darf

   Const SchrittWT = 1000
   Const SchrittWp = 1500

   Const MaxIter = 150

  Const R = 0.461526 '[kJ/kgK]
 Const Rm = 8.31451 '[J/molK]
 Const Mol = 18.015257 '[g/mol]

Const Tc = 647.096 '[K]
Const pc = 22.064 '[MPa]
Const rohc = 322 '[kg/m^3]
Const Tt = 273.16 '[K]
Const pt = 611.657 '[Pa]
Const Tb = 373.1243 '[K]

Dim II(46), J(46), n(46), J0(9), n0(9) As Double


```

```
111.
112.
113. Sub const1()
114. 'Konstanten für den Bereich 1
115.
116. II(0) = 0
117. II(1) = 0
118. II(2) = 0
119. II(3) = 0
120. II(4) = 0
121. II(5) = 0
122. II(6) = 0
123. II(7) = 0
124. II(8) = 0
125. II(9) = 1
126. II(10) = 1
127. II(11) = 1
128. II(12) = 1
129. II(13) = 1
130. II(14) = 1
131. II(15) = 2
132. II(16) = 2
133. II(17) = 2
134. II(18) = 2
135. II(19) = 2
136. II(20) = 3
137. II(21) = 3
138. II(22) = 3
139. II(23) = 4
140. II(24) = 4
141. II(25) = 4
142. II(26) = 5
143. II(27) = 8
144. II(28) = 8
145. II(29) = 21
146. II(30) = 23
147. II(31) = 29
148. II(32) = 30
149. II(33) = 31
150. II(34) = 32
151.
152. J(0) = 0
153. J(1) = -2
154. J(2) = -1
155. J(3) = 0
156. J(4) = 1
157. J(5) = 2
158. J(6) = 3
159. J(7) = 4
160. J(8) = 5
161. J(9) = -9
162. J(10) = -7
163. J(11) = -1
164. J(12) = 0
```

```
J(13) = 1
J(14) = 3
J(15) = -3
J(16) = 0
J(17) = 1
J(18) = 3
J(19) = 17
J(20) = -4
J(21) = 0
J(22) = 6
J(23) = -5
J(24) = -2
J(25) = 10
J(26) = -8
J(27) = -11
J(28) = -6
J(29) = -29
J(30) = -31
J(31) = -38
J(32) = -39
J(33) = -40
J(34) = -41

n(0) = 0
n(1) = 0.14632971213167
n(2) = -0.84548187169114
n(3) = -0.3756360367204 * 10 ^ 1
n(4) = 0.33855169168385 * 10 ^ 1
n(5) = -0.95791963387872
n(6) = 0.15772038513228
n(7) = -0.16616417199501 * 10 ^ -1
n(8) = 0.81214629983568 * 10 ^ -3
n(9) = 0.28319080123804 * 10 ^ -3
n(10) = -0.60706301565874 * 10 ^ -3
n(11) = -0.18990068218419 * 10 ^ -1
n(12) = -0.32529748770505 * 10 ^ -1
n(13) = -0.21841717175414 * 10 ^ -1
n(14) = -0.5283835796993 * 10 ^ -4
n(15) = -0.47184321073267 * 10 ^ -3
n(16) = -0.30001780793026 * 10 ^ -3
n(17) = 0.47661393906987 * 10 ^ -4
n(18) = -0.44141845330846 * 10 ^ -5
n(19) = -0.72694996297594 * 10 ^ -15
n(20) = -0.31679644845054 * 10 ^ -4
n(21) = -0.28270797985312 * 10 ^ -5
n(22) = -0.85205128120103 * 10 ^ -9
n(23) = -0.22425281908 * 10 ^ -5
n(24) = -0.65171222895601 * 10 ^ -6
n(25) = -0.14341729937924 * 10 ^ -12
n(26) = -0.40516996860117 * 10 ^ -6
n(27) = -0.12734301741641 * 10 ^ -8
n(28) = -0.17424871230634 * 10 ^ -9
n(29) = -0.68762131295531 * 10 ^ -18
n(30) = 0.14478307828521 * 10 ^ -19
```

```
219. n(31) = 0.26335781662795 * 10 ^ -22
220. n(32) = -0.11947622640071 * 10 ^ -22
221. n(33) = 0.18228094581404 * 10 ^ -23
222. n(34) = -0.93537087292458 * 10 ^ -25
223.
224. End Sub
225.
226. Sub const1rh()
227. 'Konstanten für die Rückwärtsgleichung T(h,p) für den Bereich 1
228.
229. II(0) = 0
230. II(1) = 0
231. II(2) = 0
232. II(3) = 0
233. II(4) = 0
234. II(5) = 0
235. II(6) = 0
236. II(7) = 1
237. II(8) = 1
238. II(9) = 1
239. II(10) = 1
240. II(11) = 1
241. II(12) = 1
242. II(13) = 1
243. II(14) = 2
244. II(15) = 2
245. II(16) = 3
246. II(17) = 3
247. II(18) = 4
248. II(19) = 5
249. II(20) = 6
250.
251. J(0) = 0
252. J(1) = 0
253. J(2) = 1
254. J(3) = 2
255. J(4) = 6
256. J(5) = 22
257. J(6) = 32
258. J(7) = 0
259. J(8) = 1
260. J(9) = 2
261. J(10) = 3
262. J(11) = 4
263. J(12) = 10
264. J(13) = 32
265. J(14) = 10
266. J(15) = 32
267. J(16) = 10
268. J(17) = 32
269. J(18) = 32
270. J(19) = 32
271. J(20) = 32
272.
```

```
273. n(0) = 0
274. n(1) = -0.23872489924521 * 10 ^ 3
275. n(2) = 0.40421188637945 * 10 ^ 3
276. n(3) = 0.11349746881718 * 10 ^ 3
277. n(4) = -0.58457616048039 * 10 ^ 1
278. n(5) = -0.1528548241314 * 10 ^ -3
279. n(6) = -0.10866707695377 * 10 ^ -5
280. n(7) = -0.13391744872602 * 10 ^ 2
281. n(8) = 0.43211039183559 * 10 ^ 2
282. n(9) = -0.54010067170506 * 10 ^ 2
283. n(10) = 0.30535892203916 * 10 ^ 2
284. n(11) = -0.65964749423638 * 10 ^ 1
285. n(12) = 0.93965400878363 * 10 ^ -2
286. n(13) = 0.1157364750534 * 10 ^ -6
287. n(14) = -0.25858641282073 * 10 ^ -4
288. n(15) = -0.40644363084799 * 10 ^ -8
289. n(16) = 0.66456186191635 * 10 ^ -7
290. n(17) = 0.80670734103027 * 10 ^ -10
291. n(18) = -0.93477771213947 * 10 ^ -12
292. n(19) = 0.58265442020601 * 10 ^ -14
293. n(20) = -0.15020185953503 * 10 ^ -16
294.
295. End Sub
296.
297. Sub const1rs()
298. 'Konstanten für die Rückwärtsgleichung T(h,s) für den Bereich 1
299.
300. II(0) = 0
301. II(1) = 0
302. II(2) = 0
303. II(3) = 0
304. II(4) = 0
305. II(5) = 0
306. II(6) = 0
307. II(7) = 1
308. II(8) = 1
309. II(9) = 1
310. II(10) = 1
311. II(11) = 1
312. II(12) = 1
313. II(13) = 2
314. II(14) = 2
315. II(15) = 2
316. II(16) = 2
317. II(17) = 2
318. II(18) = 3
319. II(19) = 3
320. II(20) = 4
321.
322. J(0) = 0
323. J(1) = 0
324. J(2) = 1
325. J(3) = 2
326. J(4) = 3
```

```
J(5) = 11
J(6) = 31
J(7) = 0
J(8) = 1
J(9) = 2
J(10) = 3
J(11) = 12
J(12) = 31
J(13) = 0
J(14) = 1
J(15) = 2
J(16) = 9
J(17) = 31
J(18) = 10
J(19) = 32
J(20) = 32

n(0) = 0
n(1) = 0.17478268058307 * 10 ^ 3
n(2) = 0.34806930892873 * 10 ^ 2
n(3) = 0.65292584978455 * 10 ^ 1
n(4) = 0.33039981775489 * 10 ^ 0
n(5) = -0.19281382923196 * 10 ^ -6
n(6) = -0.24909197244573 * 10 ^ -22
n(7) = -0.26107636489332 * 10 ^ 0
n(8) = 0.22592965981586 * 10 ^ 0
n(9) = -0.64256463395226 * 10 ^ -1
n(10) = 0.78876289270526 * 10 ^ -2
n(11) = 0.35672110607366 * 10 ^ -9
n(12) = 0.17332496994895 * 10 ^ -23
n(13) = 0.56608900654837 * 10 ^ -3
n(14) = -0.32635483139717 * 10 ^ -3
n(15) = 0.44778286690632 * 10 ^ -4
n(16) = -0.51322156908507 * 10 ^ -9
n(17) = -0.42522657042207 * 10 ^ -25
n(18) = 0.26400441360689 * 10 ^ -12
n(19) = 0.78124600459723 * 10 ^ -28
n(20) = -0.30732199903668 * 10 ^ -30

End Sub

Sub const2()

II(0) = 0
II(1) = 1
II(2) = 1
II(3) = 1
II(4) = 1
II(5) = 1
II(6) = 2
II(7) = 2
II(8) = 2
II(9) = 2
II(10) = 2
```

Senner Str. 156
D-33659 Bielefeld
Tel. +49 521 40 30 90
Fax +49 521 40 24 82
www.mycon-germany.com
info@mycon-germany.com


```
381. II(11) = 3
382. II(12) = 3
383. II(13) = 3
384. II(14) = 3
385. II(15) = 3
386. II(16) = 4
387. II(17) = 4
388. II(18) = 4
389. II(19) = 5
390. II(20) = 6
391. II(21) = 6
392. II(22) = 6
393. II(23) = 7
394. II(24) = 7
395. II(25) = 7
396. II(26) = 8
397. II(27) = 8
398. II(28) = 9
399. II(29) = 10
400. II(30) = 10
401. II(31) = 10
402. II(32) = 16
403. II(33) = 16
404. II(34) = 18
405. II(35) = 20
406. II(36) = 20
407. II(37) = 20
408. II(38) = 21
409. II(39) = 22
410. II(40) = 23
411. II(41) = 24
412. II(42) = 24
413. II(43) = 24
414.
415.
416. J(0) = 0
417. J(1) = 0
418. J(2) = 1
419. J(3) = 2
420. J(4) = 3
421. J(5) = 6
422. J(6) = 1
423. J(7) = 2
424. J(8) = 4
425. J(9) = 7
426. J(10) = 36
427. J(11) = 0
428. J(12) = 1
429. J(13) = 3
430. J(14) = 6
431. J(15) = 35
432. J(16) = 1
433. J(17) = 2
434. J(18) = 3
```

Senner Str. 156
D-33659 Bielefeld
Tel. +49 521 40 30 90
Fax +49 521 40 24 82
www.mycon-germany.com
info@mycon-germany.com


```
J(19) = 7
J(20) = 3
J(21) = 16
J(22) = 35
J(23) = 0
J(24) = 11
J(25) = 25
J(26) = 8
J(27) = 36
J(28) = 13
J(29) = 4
J(30) = 10
J(31) = 14
J(32) = 29
J(33) = 50
J(34) = 57
J(35) = 20
J(36) = 35
J(37) = 48
J(38) = 21
J(39) = 53
J(40) = 39
J(41) = 26
J(42) = 40
J(43) = 58


n(0) = 0
n(1) = -0.17731742473213 * 10 ^ -2
n(2) = -0.17834862292358 * 10 ^ -1
n(3) = -0.45996013696365 * 10 ^ -1
n(4) = -0.57581259083432 * 10 ^ -1
n(5) = -0.5032527872793 * 10 ^ -1
n(6) = -0.33032641670203 * 10 ^ -4
n(7) = -0.18948987516315 * 10 ^ -3
n(8) = -0.39392777243355 * 10 ^ -2
n(9) = -0.43797295650573 * 10 ^ -1
n(10) = -0.26674547914087 * 10 ^ -4
n(11) = 0.20481737692309 * 10 ^ -7
n(12) = 0.43870667284435 * 10 ^ -6
n(13) = -0.3227767723857 * 10 ^ -4
n(14) = -0.15033924542148 * 10 ^ -2
n(15) = -0.40668253562649 * 10 ^ -1
n(16) = -0.78847309559367 * 10 ^ -9
n(17) = 0.12790717852285 * 10 ^ -7
n(18) = 0.48225372718507 * 10 ^ -6
n(19) = 0.22922076337661 * 10 ^ -5
n(20) = -0.16714766451061 * 10 ^ -10
n(21) = -0.21171472321355 * 10 ^ -2
n(22) = -0.23895741934104 * 10 ^ 2
n(23) = -0.5905956432427 * 10 ^ -17
n(24) = -0.12621808899101 * 10 ^ -5
n(25) = -0.38946842435739 * 10 ^ -1
n(26) = 0.11256211360459 * 10 ^ -10
```

```
n(27) = -0.82311340897998 * 10 ^ 1
n(28) = 0.19809712802088 * 10 ^ -7
n(29) = 0.10406965210174 * 10 ^ -18
n(30) = -0.10234747095929 * 10 ^ -12
n(31) = -0.10018179379511 * 10 ^ -8
n(32) = -0.80882908646985 * 10 ^ -10
n(33) = 0.10693031879409 * 10 ^ 0
n(34) = -0.33662250574171 * 10 ^ 0
n(35) = 0.89185845355421 * 10 ^ -24
n(36) = 0.30629316876232 * 10 ^ -12
n(37) = -0.42002467698208 * 10 ^ -5
n(38) = -0.59056029685639 * 10 ^ -25
n(39) = 0.37826947613457 * 10 ^ -5
n(40) = -0.12768608934681 * 10 ^ -14
n(41) = 0.73087610595061 * 10 ^ -28
n(42) = 0.55414715350778 * 10 ^ -16
n(43) = -0.9436970724121 * 10 ^ -6

J0(0) = 0
J0(1) = 0
J0(2) = 1
J0(3) = -5
J0(4) = -4
J0(5) = -3
J0(6) = -2
J0(7) = -1
J0(8) = 2
J0(9) = 3

n0(0) = 0
n0(1) = -0.96927686500217 * 10 ^ 1
n0(2) = 0.10086655968018 * 10 ^ 2
n0(3) = -0.5608791128302 * 10 ^ -2
n0(4) = 0.71452738081455 * 10 ^ -1
n0(5) = -0.40710498223928 * 10 ^ 0
n0(6) = 0.14240819171444 * 10 ^ 1
n0(7) = -0.4383951131945 * 10 ^ 1
n0(8) = -0.28408632460772 * 10 ^ 0
n0(9) = 0.21268463753307 * 10 ^ -1


End Sub

Sub const2rha()

II(0) = 0
II(1) = 0
II(2) = 0
II(3) = 0
II(4) = 0
II(5) = 0
II(6) = 0
II(7) = 1
II(8) = 1
```

Senner Str. 156
D-33659 Bielefeld
Tel. +49 521 40 30 90
Fax +49 521 40 24 82
www.mycon-germany.com
info@mycon-germany.com


```
543. II(9) = 1
544. II(10) = 1
545. II(11) = 1
546. II(12) = 1
547. II(13) = 1
548. II(14) = 1
549. II(15) = 1
550. II(16) = 2
551. II(17) = 2
552. II(18) = 2
553. II(19) = 2
554. II(20) = 2
555. II(21) = 2
556. II(22) = 2
557. II(23) = 2
558. II(24) = 3
559. II(25) = 3
560. II(26) = 4
561. II(27) = 4
562. II(28) = 4
563. II(29) = 5
564. II(30) = 5
565. II(31) = 5
566. II(32) = 6
567. II(33) = 6
568. II(34) = 7
569.
570.
571. J(0) = 0
572. J(1) = 0
573. J(2) = 1
574. J(3) = 2
575. J(4) = 3
576. J(5) = 7
577. J(6) = 20
578. J(7) = 0
579. J(8) = 1
580. J(9) = 2
581. J(10) = 3
582. J(11) = 7
583. J(12) = 9
584. J(13) = 11
585. J(14) = 18
586. J(15) = 44
587. J(16) = 0
588. J(17) = 2
589. J(18) = 7
590. J(19) = 36
591. J(20) = 38
592. J(21) = 40
593. J(22) = 42
594. J(23) = 44
595. J(24) = 24
596. J(25) = 44
```

```
597. J(26) = 12
598. J(27) = 32
599. J(28) = 44
600. J(29) = 32
601. J(30) = 36
602. J(31) = 42
603. J(32) = 34
604. J(33) = 44
605. J(34) = 28
606.
607.
608. n(0) = 0
609. n(1) = 0.10898952318288 * 10 ^ 4
610. n(2) = 0.84951654495535 * 10 ^ 3
611. n(3) = -0.10781748091826 * 10 ^ 3
612. n(4) = 0.33153654801263 * 10 ^ 2
613. n(5) = -0.74232016790248 * 10 ^ 1
614. n(6) = 0.11765048724356 * 10 ^ 2
615. n(7) = 0.1844574935579 * 10 ^ 1
616. n(8) = -0.41792700549624 * 10 ^ 1
617. n(9) = 0.62478196935812 * 10 ^ 1
618. n(10) = -0.17344563108114 * 10 ^ 2
619. n(11) = -0.20058176862096 * 10 ^ 3
620. n(12) = 0.27196065473796 * 10 ^ 3
621. n(13) = -0.45511318285818 * 10 ^ 3
622. n(14) = 0.30919688604755 * 10 ^ 4
623. n(15) = 0.25226640357872 * 10 ^ 6
624. n(16) = -0.61707422868339 * 10 ^ -2
625. n(17) = -0.31078046629583 * 10 ^ 0
626. n(18) = 0.11670873077107 * 10 ^ 2
627. n(19) = 0.12812798404046 * 10 ^ 9
628. n(20) = -0.98554909623276 * 10 ^ 9
629. n(21) = 0.28224546973002 * 10 ^ 10
630. n(22) = -0.35948971410703 * 10 ^ 10
631. n(23) = 0.17227349913197 * 10 ^ 10
632. n(24) = -0.13551334240775 * 10 ^ 5
633. n(25) = 0.1284873466465 * 10 ^ 8
634. n(26) = 0.13865724283226 * 10 ^ 1
635. n(27) = 0.23598832556514 * 10 ^ 6
636. n(28) = -0.13105236545054 * 10 ^ 8
637. n(29) = 0.73999835474766 * 10 ^ 4
638. n(30) = -0.5519669703006 * 10 ^ 6
639. n(31) = 0.37154085996233 * 10 ^ 7
640. n(32) = 0.1912772923966 * 10 ^ 5
641. n(33) = -0.41535164835634 * 10 ^ 6
642. n(34) = -0.62459855192507 * 10 ^ 2
643.
644. End Sub
645.
646. Sub const2rsa()
647.
648. II(0) = 0
649. II(1) = -1.5
650. II(2) = -1.5
```

Senner Str. 156
D-33659 Bielefeld
Tel. +49 521 40 30 90
Fax +49 521 40 24 82
www.mycon-germany.com
info@mycon-germany.com


```
651. II(3) = -1.5
652. II(4) = -1.5
653. II(5) = -1.5
654. II(6) = -1.5
655. II(7) = -1.25
656. II(8) = -1.25
657. II(9) = -1.25
658. II(10) = -1
659. II(11) = -1
660. II(12) = -1
661. II(13) = -1
662. II(14) = -1
663. II(15) = -1
664. II(16) = -0.75
665. II(17) = -0.75
666. II(18) = -0.5
667. II(19) = -0.5
668. II(20) = -0.5
669. II(21) = -0.5
670. II(22) = -0.25
671. II(23) = -0.25
672. II(24) = -0.25
673. II(25) = -0.25
674. II(26) = 0.25
675. II(27) = 0.25
676. II(28) = 0.25
677. II(29) = 0.25
678. II(30) = 0.5
679. II(31) = 0.5
680. II(32) = 0.5
681. II(33) = 0.5
682. II(34) = 0.5
683. II(35) = 0.5
684. II(36) = 0.5
685. II(37) = 0.75
686. II(38) = 0.75
687. II(39) = 0.75
688. II(40) = 0.75
689. II(41) = 1
690. II(42) = 1
691. II(43) = 1.25
692. II(44) = 1.25
693. II(45) = 1.5
694. II(46) = 1.5
695.
696. J(0) = 0
697. J(1) = -24
698. J(2) = -23
699. J(3) = -19
700. J(4) = -13
701. J(5) = -11
702. J(6) = -10
703. J(7) = -19
704. J(8) = -15
```

```
705. J(9) = -6
706. J(10) = -26
707. J(11) = -21
708. J(12) = -17
709. J(13) = -16
710. J(14) = -9
711. J(15) = -8
712. J(16) = -15
713. J(17) = -14
714. J(18) = -26
715. J(19) = -13
716. J(20) = -9
717. J(21) = -7
718. J(22) = -27
719. J(23) = -25
720. J(24) = -11
721. J(25) = -6
722. J(26) = 1
723. J(27) = 4
724. J(28) = 8
725. J(29) = 11
726. J(30) = 0
727. J(31) = 1
728. J(32) = 5
729. J(33) = 6
730. J(34) = 10
731. J(35) = 14
732. J(36) = 16
733. J(37) = 0
734. J(38) = 4
735. J(39) = 9
736. J(40) = 17
737. J(41) = 7
738. J(42) = 18
739. J(43) = 3
740. J(44) = 15
741. J(45) = 5
742. J(46) = 18
743.
744. n(0) = 0
745. n(1) = -0.39235983861984 * 10 ^ 6
746. n(2) = 0.5152657382727 * 10 ^ 6
747. n(3) = 0.40482443161048 * 10 ^ 5
748. n(4) = -0.32193790923902 * 10 ^ 3
749. n(5) = 0.96961424218694 * 10 ^ 2
750. n(6) = -0.22867846371773 * 10 ^ 2
751. n(7) = -0.44942914124357 * 10 ^ 6
752. n(8) = -0.50118336020166 * 10 ^ 4
753. n(9) = 0.35684463560015 * 10 ^ 0
754. n(10) = 0.4423533584819 * 10 ^ 5
755. n(11) = -0.13673388811708 * 10 ^ 5
756. n(12) = 0.42163260207864 * 10 ^ 6
757. n(13) = 0.22516925837475 * 10 ^ 5
758. n(14) = 0.47442144865646 * 10 ^ 3
```

```
759. n(15) = -0.14931130797647 * 10 ^ 3
760. n(16) = -0.19781126320452 * 10 ^ 6
761. n(17) = -0.2355439947076 * 10 ^ 5
762. n(18) = -0.19070616302076 * 10 ^ 5
763. n(19) = 0.55375669883164 * 10 ^ 5
764. n(20) = 0.38293691437363 * 10 ^ 4
765. n(21) = -0.60391860580567 * 10 ^ 3
766. n(22) = 0.19363102620331 * 10 ^ 4
767. n(23) = 0.4266064369861 * 10 ^ 4
768. n(24) = -0.59780638872718 * 10 ^ 4
769. n(25) = -0.70401463926862 * 10 ^ 3
770. n(26) = 0.33836784107553 * 10 ^ 3
771. n(27) = 0.20862786635187 * 10 ^ 2
772. n(28) = 0.33834172656196 * 10 ^ -1
773. n(29) = -0.43124428414893 * 10 ^ -4
774. n(30) = 0.16653791356412 * 10 ^ 3
775. n(31) = -0.13986292055898 * 10 ^ 3
776. n(32) = -0.78849547999872 * 10 ^ 0
777. n(33) = 0.72132411753872 * 10 ^ -1
778. n(34) = -0.59754839398283 * 10 ^ -2
779. n(35) = -0.12141358953904 * 10 ^ -4
780. n(36) = 0.23227096733871 * 10 ^ -6
781. n(37) = -0.10538463566194 * 10 ^ 2
782. n(38) = 0.20718925496502 * 10 ^ 1
783. n(39) = -0.72193155260427 * 10 ^ -1
784. n(40) = 0.2074988708112 * 10 ^ -6
785. n(41) = -0.18340657911379 * 10 ^ -1
786. n(42) = 0.29036272348696 * 10 ^ -6
787. n(43) = 0.21037527893619 * 10 ^ 0
788. n(44) = 0.25681239729999 * 10 ^ -3
789. n(45) = -0.12799002933781 * 10 ^ -1
790. n(46) = -0.82198102652018 * 10 ^ -5
791.
792. End Sub
793.
794.  Private Sub KonKurT(T)
795. 'Dieses Sub legt die Konstanten für die Dampfkurvenberechnung nach
   Rivkin
796. 'abhängig von der Temperatur fest
797.
798.
799.     If -30 < T And 0.01 >= T Then
800.         p0 = 0.006112
801.         T0 = 273.16
802.         k = 21.402
803.         m = 1.0518
804.     ElseIf 0.01 < T And 70 >= T Then
805.         p0 = 0.006112
806.         T0 = 273.16
807.         k = 15.583
808.         m = 1.2746
809.     ElseIf 70 < T And 200 >= T Then
810.         p0 = 1.01325
811.         T0 = 373.15
```

```
812.         k = 10.6195
813.         m = 1.2519
814.     ElseIf 200 < T And 300 >= T Then
815.         p0 = 15.5488
816.         T0 = 473.15
817.         k = 9.2164
818.         m = 1.0698
819.     ElseIf 300 < T And 374.15 >= T Then
820.         p0 = 221.2
821.         T0 = 647.3
822.         k = 21.4328
823.         m = 0.35417
824.     Else
825.         Debug.Print "Wert liegt außerhalb des Definitionsbereichs"
826.     End If
827. End Sub
828.
829. Private Sub KonKurp(p)
830. 'Dieses Sub legt die Konstanten für die Dampfkurvenberechnung nach
   Rivkin
831. 'abhängig vom Druck fest
832.
833.     If 0.000373 < p And 0.006112 >= p Then
834.         p0 = 0.006112
835.         T0 = 273.16
836.         k = 21.402
837.         m = 1.0518
838.     ElseIf 0.006112 < p And 0.3116 >= p Then
839.         p0 = 0.006112
840.         T0 = 273.16
841.         k = 15.583
842.         m = 1.2746
843.     ElseIf 0.3116 < p And 15.5488 >= p Then
844.         p0 = 1.01325
845.         T0 = 373.15
846.         k = 10.6195
847.         m = 1.2519
848.     ElseIf 15.5488 < p And 85.9269 >= p Then
849.         p0 = 15.5488
850.         T0 = 473.15
851.         k = 9.2164
852.         m = 1.0698
853.     ElseIf 85.9269 < p And 221.2 >= p Then
854.         p0 = 221.2
855.         T0 = 647.3
856.         k = 21.4328
857.         m = 0.35417
858.     Else
859.         Debug.Print "Wert liegt außerhalb des Definitionsbereichs"
860.     End If
861. End Sub
862.
863. Private Sub KonKur()
```

```
'Dieses Sub definiert die übrigen Konstanten für die Dampfkurvenberechnung nach Rivkin

    b(0) = 2.20732
    b(1) = -0.2117187
    b(2) = -0.002166605
    b(3) = 0.0001619692
    b(4) = 0.000048998
    b(5) = 0.000003691725
    b(6) = 0

    c(0) = -3153.99
    c(1) = 29137.65
    c(2) = -122497.3
    c(3) = 298456.8
    c(4) = -363216.8
    c(5) = 178529.6
    c(6) = 0

    d(0) = 0.00271288
    d(1) = -0.0251341
    d(2) = 0.1590227
    d(3) = -0.5625152
    d(4) = 1.16296
    d(5) = -1.299779
    d(6) = 0.6110896

    e(0) = -11.54816
    e(1) = 96.15764
    e(2) = -341.8428
    e(3) = 719.7764
    e(4) = -797.3969
    e(5) = 364.0519
    e(6) = 0

    F(0) = 6010.277
    F(1) = -47493#
    F(2) = 238841.6
    F(3) = -570404.6
    F(4) = 677286.5
    F(5) = -326486.2
    F(6) = 0

    G(0) = 29.60815
    G(1) = -132.7532
    G(2) = 168.014
    G(3) = 615.1844
    G(4) = -2409.461
    G(5) = 3125.479
    G(6) = -1470.736

    km(0) = 0.9997
    km(1) = -0.029
    km(2) = -0.2
```

```
917.      km(3) = -10#
918.      km(4) = -44000000000000#
919.      km(5) = 0
920.      km(6) = 0
921.
922. End Sub
923.
924. Private Sub KonWas()
925. 'Dieses Sub definiert die übrigen Konstanten für die Berechnung im
   Wassergebiet nach Rivkin
926.
927.      e(0) = 49.4
928.      e(1) = 402.5
929.      e(2) = 4.767
930.      e(3) = 0.03333
931.
932.      F(0) = -9.25
933.      F(1) = 1.67
934.      F(2) = 0.00736
935.      F(3) = -0.008
936.
937.      G(0) = -0.073
938.      G(1) = 0.079
939.      G(2) = 0.00068
940.
941.      km(0) = 0.0000000339
942.
943.
944. End Sub
945.
946. Private Sub KonDam()
947. 'Dieses Sub definiert die übrigen Konstanten für die Berechnung im
   Dampfgebiet
948.
949.      b(0) = 0.0003237
950.      b(1) = 0.00025
951.      b(2) = -0.0011354
952.      b(3) = -0.0004381
953.
954.
955.      c(0) = 0.0000056084
956.      c(1) = -0.0000025993
957.      c(2) = -0.000000012604
958.
959.
960.      I(0) = 8
961.      I(1) = 14
962.
963.      km(0) = 2127.87
964.      km(1) = 1482.85
965.      km(2) = 379.026
966.      km(3) = 46.174
967.      km(4) = 10816.1
968.
```

```
969. End Sub
970.
971. Function hWas(T1, p1)
972. 'Diese Funktion berechnet die Enthalpie für gegebenen Druck und
   Temperatur
973. 'im Wassergebiet
974.
975.     If IsNumeric(p1) = False Or IsNumeric(T1) = False Then
976.     'Überprüft, ob die Eingabewerte numerisch sind und beendet die
   Funktion wenn nicht
977.         hWas = "Eingabewerte nicht numerisch"
978.         Exit Function
979.     End If
980.
981. const1  'Einlesen der Konstanten für diese Gleichung
982.
983. pStern = 16.53 '[MPa]
984. TStern = 1386 '[K]
985.
986.
987. p = 0.1 * p1
988. T = T1 + 273.15
989.
990. pie = p / pStern
991. tau = TStern / T
992.
993. y = 0
994.
995. For I = 1 To 34
996.     a = n(I) * (7.1 - pie) ^ II(I) * J(I) * (tau - 1.222) ^ (J(I) -
   1)
997.     y = y + a
998. Next I
999.
1000.        hWas = R * T * tau * y
1001.
1002.        End Function
1003.
1004.        Function hDam(T1, p1)
1005.        'Diese Funktion berechnet die Enthalpie für gegebenen Druck
   und Temperatur
1006.        'im Dampfgebiet
1007.
1008.            If IsNumeric(p1) = False Or IsNumeric(T1) = False Then
1009.            'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1010.                hDam = "Eingabewerte nicht numerisch"
1011.                Exit Function
1012.            End If
1013.
1014.            const2
1015.
1016.        pStern = 1 '[MPa]
1017.        TStern = 540 '[K]
```

```
1018.
1019.       p = 0.1 * p1    'Umwandlung [bar] -> [Mpa]
1020.       T = T1 + 273.15 'Umwandlung [°C] -> [K]
1021.
1022.       pie = p / pStern
1023.       tau = TStern / T
1024.
1025.       y = 0
1026.       yt = 0
1027.
1028.       For I = 1 To 43
1029.           a = n(I) * pie ^ (II(I)) * J(I) * (tau - 0.5) ^ (J(I) -
    1)
1030.           y = y + a
1031.       Next I
1032.
1033.       For L = 1 To 9
1034.           bb = n0(L) * J0(L) * tau ^ (J0(L) - 1)
1035.           yt = yt + bb
1036.       Next L
1037.
1038.       hDam = R * T * tau * (y + yt)
1039.
1040.       End Function
1041.
1042.       Function sDam(T1, p1)
1043.       'Diese Funktion berechnet die Entropie für gegebenen Druck
    und Temperatur
1044.       'im Dampfgebiet
1045.
1046.           If IsNumeric(p1) = False Or IsNumeric(T1) = False Then
1047.           'Überprüft, ob die Eingabewerte numerisch sind und
    beendet die Funktion wenn nicht
1048.               sDam = "Eingabewerte nicht numerisch"
1049.               Exit Function
1050.           End If
1051.
1052.       const2
1053.
1054.       pStern = 1 '[MPa]
1055.       TStern = 540 '[K]
1056.
1057.       p = 0.1 * p1    'Umwandlung [bar] -> [Mpa]
1058.       T = T1 + 273.15 'Umwandlung [°C] -> [K]
1059.
1060.       pie = p / pStern
1061.       tau = TStern / T
1062.
1063.       y = 0
1064.       yt = 0
1065.       yr = 0
1066.       yt1 = Application.WorksheetFunction.Ln(pie)
1067.
1068.
```

```
1069.       For I = 1 To 43
1070.           a = n(I) * pie ^ (II(I)) * J(I) * (tau - 0.5) ^ (J(I) -
   1)
1071.           y = y + a
1072.           a = n(I) * pie ^ (II(I)) * (tau - 0.5) ^ J(I)
1073.           yr = yr + a
1074.       Next I
1075.
1076.       For L = 1 To 9
1077.           bb = n0(L) * J0(L) * tau ^ (J0(L) - 1)
1078.           yt = yt + bb
1079.           bb = n0(L) * tau ^ J0(L)
1080.           yt1 = yt1 + bb
1081.       Next L
1082.
1083.       sDam = R * (tau * (y + yt) - (yr + yt1))
1084.
1085.
1086.       End Function
1087.
1088.       Function KonTem(p)
1089.       'Diese Funktion berechnet die Kondensationstemperatur für
   gegebenen Druck
1090.
1091.           If IsNumeric(p) = False Then
1092.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1093.               KonTem = "Eingabewerte nicht numerisch"
1094.               Exit Function
1095.           End If
1096.
1097.           KonKurp (p)         'Aufrufen der Konstanten
1098.
1099.           KonTem = T0 * (1 - (1 / k) *
   Application.WorksheetFunction.Ln(p / p0)) ^ (-1 / m) - 273.15
1100.           'Formel zur Berechnung der Kondensationstemperatur nach
   Rivkin
1101.
1102.       End Function
1103.
1104.       Function KonDru(T)
1105.       'Diese Funktion berechnet den Kondensationsdruck für
   gegebene Temperatur
1106.
1107.           If IsNumeric(T) = False Then
1108.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1109.               KonDru = "Eingabewerte nicht numerisch"
1110.               Exit Function
1111.           End If
1112.
1113.           KonKurT (T)         'Aufrufen der Konstanten
1114.
1115.           KonDru = p0 * Exp(k * (1 - (T0 / (T + 273.15)) ^ m))
```

```
1116.           'Formel zur Berechnung des Kondensationsdrucks nach
   Rivkin
1117.
1118.       End Function
1119.
1120.       Function hWasKon(T)
1121.       'Berechnung der Enthalpie im Kondensatzustand
1122.
1123.           If IsNumeric(T) = False Then
1124.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1125.               hWasKon = "Eingabewerte nicht numerisch"
1126.               Exit Function
1127.           End If
1128.
1129.           KonKur                      'Aufrufen der Konstanten
1130.
1131.           y = (T + 273.15) / 1000     'Definition von y (T in
   Kelvin durch 1000)
1132.           hWasKon = 0                 'Startwert für die Schleife
1133.
1134.           For nn = 0 To 5              'Diese Schleife summiert
   nach der Formel von Rivkin
1135.
1136.               hWasKon = hWasKon + c(nn) * y ^ nn
1137.
1138.           Next nn
1139.
1140.
1141.
1142.       End Function
1143.
1144.       Function hWasDam(T)
1145.       'Berechnung der Enthalpie im Dampfzustand
1146.
1147.           If IsNumeric(T) = False Then
1148.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1149.               hWasDam = "Eingabewerte nicht numerisch"
1150.               Exit Function
1151.           End If
1152.
1153.           KonKur                      'Aufrufen der Konstanten
1154.
1155.           y = (T + 273.15) / 1000     'Definition von y (T in
   Kelvin durch 1000)
1156.           hWasDam = 0
1157.
1158.           For nn = 0 To 5              'Diese Schleife summiert
   nach der Formel von Rivkin
1159.
1160.               hWasDam = hWasDam + F(nn) * y ^ nn
1161.
1162.           Next nn
```

```
1163.
1164.       End Function
1165.
1166.       Function sWasKon(T)
1167.       'Berechnung der Entropie im Kondensatzustand
1168.
1169.           If IsNumeric(T) = False Then
1170.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1171.               sWasKon = "Eingabewerte nicht numerisch"
1172.               Exit Function
1173.           End If
1174.
1175.           KonKur                      'Aufrufen der Konstanten
1176.
1177.           y = (T + 273.15) / 1000     'Definition von y (T in
   Kelvin durch 1000)
1178.           sWasKon = 0
1179.
1180.           For nn = 0 To 5              'Diese Schleife summiert
   nach der Formel von Rivkin
1181.
1182.               sWasKon = sWasKon + e(nn) * y ^ nn
1183.
1184.           Next nn
1185.
1186.       End Function
1187.
1188.       Function sWasDam(T)
1189.       'Berechnung der Entropie im Dampfzustand
1190.
1191.           If IsNumeric(T) = False Then
1192.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1193.               sWasDam = "Eingabewerte nicht numerisch"
1194.               Exit Function
1195.           End If
1196.
1197.           KonKur                      'Aufrufen der Konstanten
1198.
1199.           y = (T + 273.15) / 1000     'Definition von y (T in
   Kelvin durch 1000)
1200.           sWasDam = 0
1201.
1202.           For nn = 0 To 6              'Diese Schleife summiert
   nach der Formel von Rivkin
1203.
1204.               sWasDam = sWasDam + G(nn) * y ^ nn
1205.
1206.           Next nn
1207.
1208.       End Function
1209.
```

```
1210.       Function pvonhTDam(h, T, Optional Startwert = 1, Optional
   Schrittweite = 1)
1211.       'Diese Function berechnet den Druck iterativ für gegebene
   Temperatur
1212.       'und Enthalpie im Dampfgebiet
1213.       'Optional kann man auch den Startwert des Druckes und die
   Schrittweite bei der Iteration beeinflussen.
1214.
1215.
1216.
1217.           If IsNumeric(h) = False Or IsNumeric(T) = False Then
1218.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1219.               pvonhTDam = "Eingabewerte nicht numerisch"
1220.               Exit Function
1221.           End If
1222.
1223.           StartH = hDam(T, Startwert)     'StartH ist der
   Startwert der Enthalpie für die Iteration
1224.
1225.           kr = 0  'Zähler für Iterationsschritte
1226.           J = 0   'Zähler für Oszillationen
1227.           L = 1   'Dieser Parameter ermöglicht eine dynamische
   Anpassung der Schrittweite
1228.           Q = 0   'Überwacht, ob eine Annäherung vorhanden ist
1229.
1230.
1231.
1232.           Do While Abs(StartH - h) >= h * AbweichungHpDam / 100
   'Die Schleife wird solange durchlaufen, bis die errechnete Enthalpie
   nah genug an der gegebenen Enthalpie ist
1233.
1234.               kn = k
1235.
1236.               Startwert = Startwert + Schrittweite * L * SchrittWp
   * (StartH - h) / h 'Der Wert für den Druck wird angepasst
1237.
1238.                   If Startwert < 0 Then
1239.                   'Korrigiert die Schrittweite und fängt von vorne
   an, falls
1240.                   'der Startwert negativ werden sollte
1241.                       L = L * 0.5
1242.                       Startwert = 1
1243.                   End If
1244.
1245.               k1 = StartH
1246.
1247.               StartH = hDam(T, Startwert)     'Neue Enthalpie zur
   Kontrolle, ob weiterer Schleifendurchlauf nötig
1248.
1249.               k2 = StartH
1250.               k = k1 - k2
1251.
```

```
1252.                If Abs(kn + k) < 0.1 * Abs(k) Then 'And (K < 0 And
   kn > 0 Or K > 0 And kn < 0) Then
1253.                    J = J + 1
1254.                Else
1255.                    J = 0
1256.                End If
1257.
1258.                If Abs(k) > Abs(kn) Then
1259.                    Q = Q + 1
1260.                Else
1261.                    Q = 0
1262.                End If
1263.
1264.                If J > 4 Then
1265.                'Nach 4 aufeinander folgenden Oszillierenden
   Iterationen soll die Schrittweite halbiert werden
1266.                    L = L * 0.5
1267.                End If
1268.
1269.                If Q > 4 Then
1270.                'Nach 3 aufeinander folgenden Iterationen ohne
   Annäherung soll die Schrittweite reduziert werden
1271.                    L = L * 0.5
1272.                End If
1273.
1274.                kr = kr + 1
1275.
1276.                If Abs(StartH) > 500000 Then
1277.                    Debug.Print "Iteration pvonhTDam divergiert"
1278.
1279.                    Startwert = "Divergenz"
1280.                    Exit Do
1281.                End If
1282.
1283.                If kr = MaxIter Then
1284.                    Debug.Print "Maximale Iterationen erreicht in
   pvonhTDam"
1285.
1286.                    Startwert = "Zu viele Iterationen"
1287.
1288.                    Exit Do
1289.                End If
1290.
1291.            Debug.Print "j:"; J; "Q:"; Q; "Iteration"; kr; ":";
   Startwert; "L:"; L
1292.            Loop
1293.
1294.            pvonhTDam = Startwert    'Der Wert wird an die Funktion
   übergeben
1295.            'Debug.Print kr
1296.
1297.
1298.        End Function
1299.
```

```
1300.       Function TvonhpDam(h, p1)
1301.       'Diese Function berechnet die Temperatur iterativ für
   gegebenen Druck
1302.       'und Enthalpie im Dampfgebiet
1303.       'Optional kann man auch den Startwert der Temperatur und die
   Schrittweite bei der Iteration beeinflussen.
1304.
1305.           If IsNumeric(h) = False Or IsNumeric(p1) = False Then
1306.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1307.               TvonhpDam = "Eingabewerte nicht numerisch"
1308.               Exit Function
1309.           End If
1310.
1311.           const2rha    'Einlesen der Konstanten für diese
   Gleichung
1312.
1313.       p = 0.1 * p1  'Umwandlung [bar] -> [Mpa]
1314.
1315.       pStern = 1    '[K]
1316.       hStern = 2000 '[kJ/kg]
1317.
1318.       pie = p / pStern
1319.       Nuu = h / hStern
1320.
1321.       y = 0
1322.
1323.       For I = 1 To 34
1324.           a = n(I) * pie ^ II(I) * (Nuu - 2.1) ^ J(I)
1325.           y = y + a
1326.       Next I
1327.
1328.       TvonhpDam = y - 273.15
1329.       End Function
1330.
1331.       Function pvonhTWas(h, T, Optional Startwert = 20, Optional
   Schrittweite = 1)
1332.
1333.       'Diese Function berechnet den Druck iterativ für gegebene
   Temperatur
1334.       'und Enthalpie im Wassergebiet.
1335.       'Optional kann man auch den Startwert des Druckes und die
   Schrittweite bei der Iteration beeinflussen.
1336.
1337.           If IsNumeric(h) = False Or IsNumeric(T) = False Then
1338.
1339.           'Überprüft, ob die Eingabewerte numerisch sind, sind sie
   es nicht, wird die
1340.           'Funtion beendet und als Ergebnis eine Fehlermeldung
   herausgegeben.
1341.
1342.               pvonhTWas = "Eingabewerte nicht numerisch"
1343.               Exit Function
1344.           End If
```

```
1345.
1346.           StartH = hWas(T, Startwert)
1347.           'StartH ist der Startwert der Enthalpie für die
   Iteration
1348.
1349.           kr = 0  'Zähler für Iterationsschritte
1350.           J = 0   'Zähler für Oszillationen
1351.           L = 1   'Dieser Parameter ermöglicht eine dynamische
   Anpassung der Schrittweite
1352.           Q = 0   'Überwacht, ob eine Annäherung vorhanden ist
1353.
1354.
1355.           Do While Abs(StartH - h) >= h * AbweichungHpWas / 100
1356.           'Die Schleife wird solange durchlaufen, bis die
1357.           'errechnete Enthalpie nah genug an der gegebenen
   Enthalpie ist
1358.
1359.               kn = k
1360.
1361.               Startwert = Startwert + Schrittweite * L * SchrittWp
   * 40 * (h - StartH) / h
1362.               'Der Wert für den Druck wird angepasst
1363.
1364.                   If Startwert < 0 Then
1365.                   'Korrigiert die Schrittweite und fängt von vorne
   an, falls
1366.                   'der Startwert negativ werden sollte
1367.                       L = L * 0.25
1368.                       Startwert = 20
1369.                   End If
1370.
1371.               k1 = StartH
1372.
1373.               StartH = hWas(T, Startwert)     'Neue Enthalpie zur
   Kontrolle, ob weiterer Schleifendurchlauf nötig
1374.
1375.               k2 = StartH
1376.               k = k1 - k2
1377.
1378.               If Abs(kn + k) < 0.1 * Abs(k) Then 'And (K < 0 And
   kn > 0 Or K > 0 And kn < 0) Then
1379.                   J = J + 1
1380.               Else
1381.                   J = 0
1382.               End If
1383.
1384.               If Abs(k) > Abs(kn) Then
1385.                   Q = Q + 1
1386.               Else
1387.                   Q = 0
1388.               End If
1389.
1390.               If J > 4 Then
```

```
1391.                'Nach 4 aufeinander folgenden Oszillierenden
   Iterationen soll die Schrittweite halbiert werden
1392.                    L = L * 0.5
1393.                    J = 0
1394.                End If
1395.
1396.                If Q > 2 Then
1397.                'Nach 3 aufeinander folgenden Iterationen ohne
   Annäherung soll die Schrittweite reduziert werden
1398.                    L = L * 0.5
1399.                    'Q = 0
1400.                End If
1401.
1402.
1403.                kr = kr + 1
1404.
1405.                If Abs(StartH) > 50000 Then
1406.                    Debug.Print "Iteration pvonhTWas divergiert"
1407.                    Startwert = "Divergenz"
1408.                    Exit Do
1409.                End If
1410.
1411.                If kr = MaxIter Then
1412.                    Debug.Print "Maximale Iterationen erreicht in
   pvonhTWas"
1413.                    Startwert = "Zu viele Iterationen"
1414.
1415.                    Exit Do
1416.                End If
1417.            'Debug.Print "j:"; j; "Q:"; Q; "Iteration"; kr; ":";
   Startwert; "L:"; L
1418.            Loop
1419.
1420.            pvonhTWas = Startwert
1421.
1422.        End Function
1423.
1424.        Function TvonhpWas(h, p1)
1425.        'Diese Function berechnet die Temperatur iterativ für
   gegebenen Druck
1426.        'und Enthalpie im Wassergebiet
1427.        'Optional kann man auch den Startwert der Temperatur und die
   Schrittweite bei der Iteration beeinflussen.
1428.
1429.            If IsNumeric(h) = False Or IsNumeric(p1) = False Then
1430.            'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1431.                TvonhpWas = "Eingabewerte nicht numerisch"
1432.                Exit Function
1433.            End If
1434.
1435.            const1rh    'Einlesen der Konstanten für diese Gleichung
1436.
1437.        p = 0.1 * p1  'Umwandlung [bar] -> [Mpa]
```

```
1438.
1439.       pStern = 1    '[K]
1440.       hStern = 2500 '[kJ/kgK]
1441.
1442.       pie = p / pStern
1443.       Nuu = h / hStern
1444.
1445.       For I = 1 To 20
1446.           a = n(I) * pie ^ II(I) * (Nuu + 1) ^ J(I)
1447.           y = y + a
1448.       Next I
1449.
1450.       TvonhpWas = y - 273.15
1451.
1452.       End Function
1453.
1454.
1455.
1456.       Function TvonspDam(S, p1)
1457.       'Diese Function berechnet die Temperatur iterativ für
   gegebenen Druck
1458.       'und Entropie im Dampfgebiet
1459.       'Optional kann man auch den Startwert der Temperatur und die
   Schrittweite bei der Iteration beeinflussen.
1460.
1461.       Dim StartS As Double
1462.
1463.           If IsNumeric(S) = False Or IsNumeric(p1) = False Then
1464.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1465.               TvonspDam = "Eingabewerte nicht numerisch"
1466.               Exit Function
1467.           End If
1468.
1469.           const2rsa    'Einlesen der Konstanten für diese
   Gleichung
1470.
1471.       p = 0.1 * p1  'Umwandlung [bar] -> [Mpa]
1472.
1473.       pStern = 1 '[Mpa]
1474.       sStern = 2 '[kJ/kgK]
1475.
1476.       pie = p / pStern
1477.       sig = S / sStern
1478.
1479.       y = 0
1480.
1481.       For I = 1 To 46
1482.           a = n(I) * pie ^ II(I) * (sig - 2) ^ J(I)
1483.           y = y + a
1484.       Next I
1485.
1486.       TvonspDam = y - 273.15
1487.       End Function
```

```
1488.
1489.
1490.       Function pvonsTDam(S, T, Optional Startwert = 30, Optional
   Schrittweite = 1)
1491.
1492.           If IsNumeric(S) = False Or IsNumeric(T) = False Then
1493.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1494.               pvonsTDam = "Eingabewerte nicht numerisch"
1495.               Exit Function
1496.           End If
1497.
1498.           Dim StartS As Double
1499.
1500.           StartS = sDam(T, Startwert)     'StartS ist der
   Startwert der Enthalpie für die Iteration
1501.           kr = 0  'Zähler für Iterationsschritte
1502.           J = 0   'Zähler für Oszillationen
1503.           L = 1   'Dieser Parameter ermöglicht eine dynamische
   Anpassung der Schrittweite
1504.           Q = 0   'Überwacht, ob eine Annäherung vorhanden ist
1505.
1506.           Do While Abs(StartS - S) >= S * AbweichungSp / 100 'Die
   Schleife wird solange durchlaufen, bis die errechnete Enthalpie nah
   genug an der gegebenen Enthalpie ist
1507.               kn = k
1508.               Startwert = Startwert + Schrittweite * SchrittWp * 4
   * L * (StartS - S) / (S) 'Der Wert für den Druck wird angepasst
1509.
1510.                   If Startwert < 0 Then
1511.                   'Korrigiert die Schrittweite und fängt von vorne
   an, falls
1512.                   'der Startwert negativ werden sollte
1513.                       L = L * 0.25
1514.                       Startwert = 1
1515.                   End If
1516.
1517.               k1 = StartS
1518.               StartS = sDam(T, Startwert)     'Neue Enthalpie zur
   Kontrolle, ob weiterer Schleifendurchlauf nötig
1519.               k2 = StartS
1520.               k = k1 - k2
1521.
1522.               If Abs(kn + k) < 0.1 * Abs(k) Then 'And (K < 0 And
   kn > 0 Or K > 0 And kn < 0) Then
1523.                   J = J + 1
1524.               Else
1525.                   J = 0
1526.               End If
1527.
1528.               If Abs(k) > Abs(kn) Then
1529.                   Q = Q + 1
1530.               Else
1531.                   Q = 0
```

```
1532.                End If
1533.
1534.                If J > 4 Then
1535.                'Nach 4 aufeinander folgenden Oszillierenden
   Iterationen soll die Schrittweite halbiert werden
1536.                    L = L * 0.5
1537.                End If
1538.
1539.                If Q > 3 Then
1540.                'Nach 3 aufeinander folgenden Iterationen ohne
   Annäherung soll die Schrittweite reduziert werden
1541.                    L = L * 0.5
1542.                    'Startwert = 1
1543.                End If
1544.
1545.                kr = kr + 1
1546.
1547.                If Abs(StartS) > 5000 Then
1548.                    Debug.Print "Iteration pvonsTDam divergiert"
1549.
1550.                    Startwert = "Divergenz"
1551.
1552.                    Exit Do
1553.                End If
1554.
1555.                If kr = MaxIter Then
1556.                    Debug.Print "Maximale Iterationen erreicht in
   pvonsTDam"
1557.                    Startwert = "Zu viele Iterationen"
1558.
1559.                    Exit Do
1560.                End If
1561.            'Debug.Print "j:"; j; "Q:"; Q; "Iteration"; kr; ":";
   Startwert; "L:"; L
1562.            Loop
1563.
1564.            pvonsTDam = Startwert    'Der Wert wird an die Funktion
   übergeben
1565.
1566.        End Function
1567.        Function xvonhT(h, T)
1568.        'Diese Funktion berechnet den Dampfgehalt aus der Enthalpie
   [kJ/kg] bei einer bestimmten Temperatur [°C].
1569.
1570.            If IsNumeric(x) = False Or IsNumeric(T) = False Then
1571.            'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1572.                xvonhT = "Eingabewerte nicht numerisch"
1573.                Exit Function
1574.            End If
1575.
1576.        Dim h1, h2 As Double
1577.
1578.            h1 = hWasKon(T)
```

```
1579.           h2 = hWasDam(T)
1580.
1581.           xvonhT = (h - h1) / (h2 - h1)
1582.
1583.
1584.       End Function
1585.
1586.       Function xvonsT(S, T)
1587.       'Diese Funktion berechnet den Dampfgehalt aus der Entropie
   [kJ/kg] bei einer bestimmten Temperatur [°C].
1588.
1589.           If IsNumeric(S) = False Or IsNumeric(T) = False Then
1590.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1591.               xvonsT = "Eingabewerte nicht numerisch"
1592.               Exit Function
1593.           End If
1594.
1595.           Dim s1, s2 As Double
1596.
1597.           s1 = sWasKon(T)
1598.           s2 = sWasDam(T)
1599.
1600.           xvonsT = (S - s1) / (s2 - s1)
1601.
1602.       End Function
1603.
1604.       Function hvonxT(x, T)
1605.       'Diese Funktion berechnet die Enthalpie [kJ/kg] aus dem
   Dampfgehalt und der Temperatur [°C]
1606.
1607.           If IsNumeric(x) = False Or IsNumeric(T) = False Then
1608.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1609.               hvonxT = "Eingabewerte nicht numerisch"
1610.               Exit Function
1611.           End If
1612.
1613.       hvonxT = x * (hWasDam(T) - hWasKon(T)) + hWasKon(T)
1614.
1615.       End Function
1616.
1617.       Function svonxT(x, T)
1618.       'Diese Funktion berechnet die Entropie [kJ/kg] aus dem
   Dampfgehalt und der Temperatur [°C]
1619.
1620.           If IsNumeric(x) = False Or IsNumeric(T) = False Then
1621.           'Überprüft, ob die Eingabewerte numerisch sind und
   beendet die Funktion wenn nicht
1622.               svonxT = "Eingabewerte nicht numerisch"
1623.               Exit Function
1624.           End If
1625.
1626.       svonxT = x * (sWasDam(T) - sWasKon(T)) + sWasKon(T)
```

```
1627.
1628.       End Function
```

Zur Abschätzung des Druckes in den noch unbekannten Turbinenabschnitten, z.B. nach der Düsengruppenregelung, oder nach der Entnahme, wird die Theorie von Stodola (Gesetz des Dampfkegels) [2] herangezogen. Die Definition der beiden Funktionen zeigt folgender Quellcode:

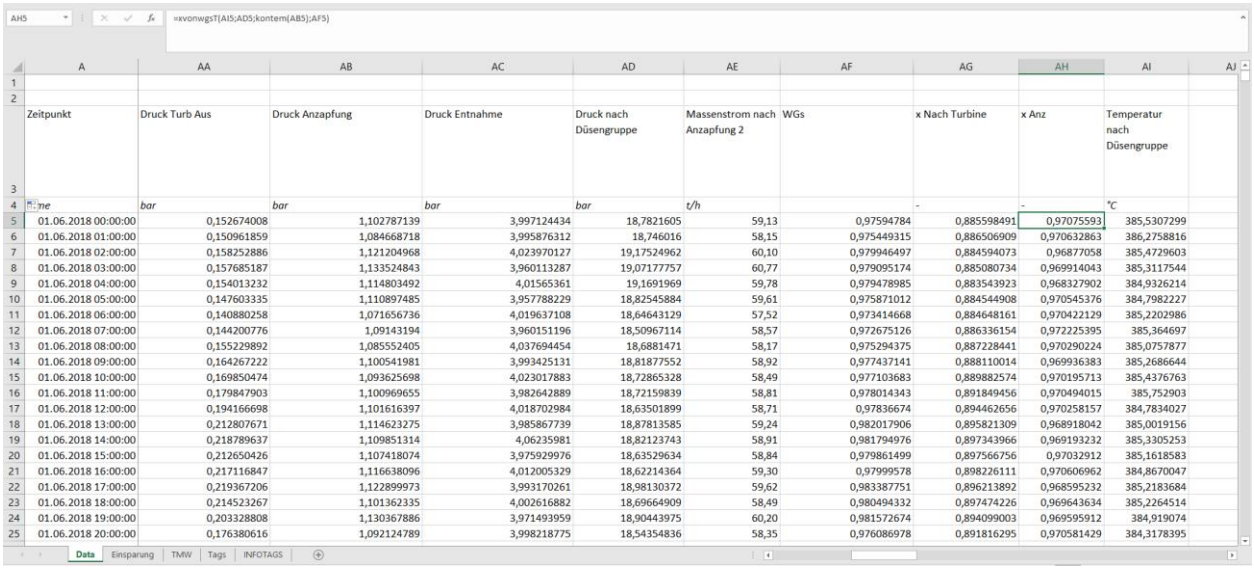
```
1. 'Copyright <2020> <mycon GmbH>
2. '<author: P. Modler>
3.
4. 'Permission is hereby granted, free of charge, to any person
   obtaining a copy of this software and associated documentation files
   (the "Software"), to deal in the Software without restriction,
   including without limitation the rights to use, copy, modify, merge,
   publish, distribute, sublicense, and/or sell copies of the Software,
   and to permit persons to whom the Software is furnished to do so,
   subject to the following conditions:
5.
6. 'The above copyright notice and this permission notice shall be
   included in all copies or substantial portions of the Software.
7.
8. 'THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
   EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
   MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
   NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
   BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
   ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
   CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
   SOFTWARE.
9.
10.
11.  Function stodop1(p2, m, p1a, p2a, ma, Optional expo = 2)
12.  'Berechnet den Druck am Eintritt eines Turbinenabschnitts abhängig
13.  'vom Druck am Austritt dieses Abschnittes und des Massenstroms.
14.  'p1: Druck am Eintritt des Abschnittes
15.  'p2: Druck am Austritt des Abschnittes
16.  'm : Dampfmassenstrom durch den Abschnitt
17.  'p1a, p2a und ma sind die jeweiligen Größen in einem
   Referenzzustand.
18.  stodop1 = (p2 ^ expo + (p1a ^ expo - p2a ^ expo) * m ^ expo / ma ^
   expo) ^ (1 / expo)
19.  End Function
20.
21.  Function stodop2(p1, m, p1a, p2a, ma)
22.  'Berechnet den Druck am Austritt eines Turbinenabschnitts abhängig
23.  'vom Druck am Eintritt dieses Abschnittes und des Massenstroms.
24.  'p1: Druck am Eintritt des Abschnittes
25.  'p2: Druck am Austritt des Abschnittes
26.  'm : Dampfmassenstrom durch den Abschnitt
```

```
27.  'p1a, p2a und ma sind die jeweiligen Größen in einem
   Referenzzustand.
28.  expo = 2
29.  stodop2 = (p1 ^ expo - (p1a ^ expo - p2a ^ expo) * m ^ expo / ma ^
   expo) ^ (0.5)
30.  End Function
```

Ein mögliches Anwendungsbeispiel zeigt der folgende Code, bei dem der Druck in der Anzapfung der Turbine berechnet wird aus dem Druck des Turbinen-Austritts.

```
1.  'Copyright <2020> <mycon GmbH>
2.  '<author: P. Modler>
3.
4.  'Permission is hereby granted, free of charge, to any person
   obtaining a copy of this software and associated documentation files
   (the "Software"), to deal in the Software without restriction,
   including without limitation the rights to use, copy, modify, merge,
   publish, distribute, sublicense, and/or sell copies of the Software,
   and to permit persons to whom the Software is furnished to do so,
   subject to the following conditions:
5.
6.  'The above copyright notice and this permission notice shall be
   included in all copies or substantial portions of the Software.
7.
8.  'THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
   EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
   MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
   NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
   BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
   ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
   CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
   SOFTWARE.
9.
10.
11.  Function pAnzEmp(pTurbAus, mLuKo)
12.
13.  p10 = 1.104564
14.  p20 = 0.152512
15.  m0 = 59.34322
16.  expo = 2
17.
18.  pAnzEmp = stodop1(pTurbAus, mLuKo, p10, p20, m0, expo)
19.
20.  End Function
```

Damit ist es möglich alle relevanten Drücke in der Turbine abzuschätzen. Eine mögliche Implementierung in Excel kann folgendermaßen aussehen:



Zeitpunkt	Druck Turb Aus	Druck Anzapfung	Druck Entnahme	Druck nach Düsengruppe	Massenstrom nach Anzapfung 2	WGs	x Nach Turbine	x Anz	Temperatur nach Düsengruppe
ne	bar	bar	bar	bar	t/h	-	-	-	°C
01.06.2018 00:00:00	0,152674008	1,102787139	3,997124434	18,7821605	59,13	0,97594784	0,885598491	0,97075593	385,5307299
01.06.2018 01:00:00	0,150961859	1,084668718	3,995876312	18,746016	58,15	0,975449315	0,886506909	0,970632863	386,2758816
01.06.2018 02:00:00	0,158252886	1,121204968	4,023970127	19,17524962	60,10	0,979946497	0,884594073	0,96877058	385,4729603
01.06.2018 03:00:00	0,157685187	1,133524843	3,960113287	19,07177757	60,77	0,979095174	0,885080734	0,969914043	385,3117544
01.06.2018 04:00:00	0,154013232	1,114803492	4,01565361	19,1691969	59,78	0,979478985	0,883543923	0,968327902	384,9326214
01.06.2018 05:00:00	0,147603335	1,110897485	3,957788229	18,82545884	59,61	0,975871012	0,884544908	0,970545376	384,7982227
01.06.2018 06:00:00	0,140880258	1,071656736	4,019637108	18,64643129	57,52	0,973414668	0,884648161	0,970422129	385,2202986
01.06.2018 07:00:00	0,144200776	1,09143194	3,960151196	18,50967114	58,57	0,972675126	0,886336154	0,972225395	385,364697
01.06.2018 08:00:00	0,155229892	1,085552405	4,037694454	18,6881471	58,17	0,975294375	0,887228441	0,970290224	385,0757877
01.06.2018 09:00:00	0,164267222	1,100541981	3,993425131	18,81877552	58,92	0,977437141	0,888110014	0,969936383	385,2686644
01.06.2018 10:00:00	0,169850474	1,093625698	4,023017883	18,72865328	58,49	0,977103683	0,889882574	0,970195713	385,4376763
01.06.2018 11:00:00	0,179847903	1,100969655	3,982642889	18,72159839	58,81	0,978014343	0,891849456	0,970494015	385,752903
01.06.2018 12:00:00	0,194166698	1,101616397	4,018702984	18,63501899	58,71	0,97836674	0,894462656	0,970258157	384,7834027
01.06.2018 13:00:00	0,212807671	1,114623275	3,985867739	18,87813585	59,24	0,982017906	0,895821309	0,968918042	385,0019156
01.06.2018 14:00:00	0,218789637	1,109851314	4,06235981	18,82123743	58,91	0,981794976	0,897343966	0,969193232	385,3305253
01.06.2018 15:00:00	0,212650426	1,107418074	3,975929976	18,63529634	58,84	0,979861499	0,897566756	0,97032912	385,1618583
01.06.2018 16:00:00	0,217116847	1,116638096	4,012005329	18,62214364	59,30	0,97999578	0,898226111	0,970606962	384,8670047
01.06.2018 17:00:00	0,219367206	1,122899973	3,993170261	18,98130372	59,62	0,983387751	0,896213892	0,968595232	385,2183684
01.06.2018 18:00:00	0,214523267	1,101362335	4,002616882	18,69664909	58,49	0,980494332	0,897474226	0,969643634	385,2264514
01.06.2018 19:00:00	0,203328808	1,130367886	3,971493959	18,90443975	60,20	0,981572674	0,894099003	0,969595912	384,919074
01.06.2018 20:00:00	0,176380616	1,092124789	3,998218775	18,54354836	58,35	0,976086978	0,891816295	0,970581429	384,3178395

Hier wurden die Drücke, Dampfgehalte und Temperaturen in den verschiedenen Turbinenstufen spaltenweise für Stunden-Mittelwerte berechnet.

Nun werden noch die Temperaturen und die Massenströme aus der Turbine und durch den Luft-Kondensator (LuKo) benötigt. Diese werden mit Hilfe folgender Funktionen bestimmt:

```
1.  'Copyright <2020> <mycon GmbH>
2. '<author: P. Modler>
3.
4. 'Permission is hereby granted, free of charge, to any person
   obtaining a copy of this software and associated documentation files
   (the "Software"), to deal in the Software without restriction,
   including without limitation the rights to use, copy, modify, merge,
   publish, distribute, sublicense, and/or sell copies of the Software,
   and to permit persons to whom the Software is furnished to do so,
   subject to the following conditions:
5.
6. 'The above copyright notice and this permission notice shall be
   included in all copies or substantial portions of the Software.
7.
8. 'THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
   EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
   MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
   NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
   BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
   ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
```

```
   CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
   SOFTWARE.
9.
10.
11.  Function xvonWGsT(T1, p1, T2, WGS)
12.  'Berechnung des Dampfgehalts an einem Punkt mit bekannter
   Temperatur in der Turbine,
13.  'abhängig vom isentropen Wirkungsgrad.
14.
15.  'T1: Temperatur nach der Düsengruppenregelung [°C]
16.  'p1: Druck nach der Düsengruppenregelung [bar]
17.  'T2: Temperatur am Punkt der Berechnung des Dampfgehalts [°C]
18.  'WGS: Isentroper Wirkungsgrad zwischen Düsengruppenregelung und
   Punkt der Berechnung [-]
19.
20.  xvonWGsT = xvonhT(hDam(T1, p1) - WGS * (hDam(T1, p1) -
   hvonxT(xvonsT(sDam(T1, p1), T2), T2)), T2)
21.
22.  End Function
23.
24.  Function Tisent(T1, p1, p2)
25.  'Temperatur bei isenthalpem Prozess
26.
27.  'T1: Temperatur am Referenzpunkt [°C]
28.  'p1: Druck am Referenzpunkt [bar]
29.  'p2: Druck am Punkt der Berechnung [bar]
30.
31.  h1 = hDam(T1, p1)
32.  Tisent = TvonhpDam(h1, p2)
33.
34.  End Function
35.
36.  Function manzapf(mKonVW, TKonVW, pKonVW, TKA, xAnz, TAnz)
37.  'Hilfsfunktion für Massenstrom aus der Anzapfung
38.
39.  'mKonVW: Massenstrom durch den Kondensatvorwärmer [t/h]
40.  'TKonVW: Temperatur im Kondensatvorwärmer [°C]
41.  'pKonVW: Druck im Kondensatvorwärmer [bar]
42.  'TKA: Temperatur im Kondensator [°C]
43.  'xAnz: Dampfgehalt in der Anzapfung [-]
44.  'TAnz: Temperatur in der Anzapfung [°C]
45.
46.  dhLuKo = hWas(TKonVW, pKonVW) - hWas(TKA, pKonVW)
47.  dhanzapf = hvonxT(xAnz, TAnz) - hvonxT(0, TAnz)
48.
49.  manzapf = mKonVW * (dhLuKo / dhanzapf) / (1 + dhLuKo / dhanzapf)
50.
51.  End Function
52.
53.  Function mAnzIte(mFD, mEnt, tnachKonVW, pKonVW, TK, xAnz, TAnz)
54.  'Berechnung des Massenstroms aus der Anzapfung.
55.
56.  'mFD: Frischdampf-Massenstrom [t/h]
57.  'mEnt: Entnahme-Massenstrom [t/h]
```

```
58.  'tnachKonVW: Temperatur des Kondensats nach dem Kondensatvorwärmer
   [°C]
59.  'pKonVW: Druck im Kondensatvorwärmer [bar]
60.  'TKA: Temperatur im Kondensator [°C]
61.  'xAnz: Dampfgehalt in der Anzapfung [-]
62.  'TAnz: Temperatur in der Anzapfung [°C]
63.
64.  mAnzIte = manzapf(mFD - mEnt - manzapf(mFD - mEnt, tnachKonVW,
   pKonVW, TK, xAnz, TAnz), tnachKonVW, pKonVW, TK, xAnz, TAnz)
65.
66.  End Function
67.
68.  Function mEntnahme(mD, TSWBA, pSWB, Tent, pEnt, TKonVW, pKonVW)
69.  'Berechnung des Massenstroms aus der Entnahme
70.
71.  'mD: Dampfmassenstrom [t/h]
72.  'TSWBA: Temperatur nach dem Speisewasserbehälter (Speisewasser)
   [°C]
73.  'pSWB: Druck im Speisewasserbehälter [bar]
74.  'Tent: Temperatur in der Entnahme [°C]
75.  'pEnt: Druck in der Entnahme [bar]
76.  'TKonVW: Temperatur im Kondensatvorwärmer [°C]
77.  'pkonVW: Druck im Kondensatvorwärmer [bar]
78.
79.  mEntnahme = mD * (hWas(TSWBA, pSWB) - hWas(TKonVW, pKonVW)) /
   (hDam(Tent, pEnt) - hWas(TKonVW, pKonVW))
80.
81.  End Function
82.
83.  Function EntTurb(TtEin, ptEin, MEin, TAnz2, pAnz2, mAnz2, pAnz1,
   xAnz1, mAnz1, ptAus, xtAus)
84.  'Berechnung der Enthalpieabgabe des Dampfes in der Turbine
85.
86.  'Ttein: Temperatur am Turbinen-Eintritt [°C]
87.  'ptein: Druck am Turbinen-Eintritt [bar]
88.  'MEin: Frischdampfmassenstrom [t/h]
89.  'Tanz2: Temperatur an der zweiten Anzapfung [°C]
90.  'pAnz2: Druck an der zweiten Anzapfung [bar]
91.  'mAnz2: Massenstrom aus der zweiten Anzapfung [t/h]
92.  'panz1: Druck an der ersten Anzapfung (Entnahme) [bar]
93.  'xAnz1: Dampfgehalt an der ersten Anzapfung (Entnahme) [-]
94.  'mAnz1: Massenstrom aus der ersten Anzapfung (Entnahme) [t/h]
95.  'pTaus: Druck am Turbinen-Austritt (bar)
96.  'xTaus: Dampfgehalt am Turbinen-Austritt (-)
97.
98.  htEin = hDam(TtEin, ptEin)
99.  hAnz2 = hDam(TAnz2, pAnz2)
100. hAnz1 = hvonxT(xAnz1, KonTem(pAnz1))
101. htAus = hvonxT(xtAus, KonTem(ptAus))
102.
103. EntTurb = (MEin * (htEin - hAnz2) + (MEin - mAnz2) * (hAnz2 -
   hAnz1) + (MEin - mAnz2 - mAnz1) * (hAnz1 - htAus)) / 3.6
104.
105. End Function
```

```
106.
107. Function EntLuKo(xTurb, TK, mLuKo)
108. 'Berechnung der Enthalpieabgabe des Dampfes im LuKo
109.
110. 'xTurb: Dampfgehalt am Turbinen-Austritt [-]
111. 'TK: Temperatur im LuKo [°C]
112. 'mLuKo: Massenstrom im LuKo [t/h]
113.
114. EntLuKo = (hvonxT(xTurb, TK) - hvonxT(0, TK)) * mLuKo / 3.6
115.
116. End Function
117.
118. Function mLuft(TL, I1, I2, I3, I4, I5, I6, U)
119. 'Berechnung des Luftmassenstroms durch den Luft-Kondensator (LuKo)
120.
121. 'TL: Temperatur der Luft am Eintritt (Umgebungstemperatur) [°C]
122. 'I1: Stromverbrauch des 1. Ventilators [A]
123. 'I2: Stromverbrauch des 2. Ventilators [A]
124. 'I3: Stromverbrauch des 3. Ventilators [A]
125. 'I4: Stromverbrauch des 4. Ventilators [A]
126. 'I5: Stromverbrauch des 5. Ventilators [A]
127. 'I6: Stromverbrauch des 6. Ventilators [A]
128. 'U: Spannung [V]
129.
130. AL = 6 ^ 2 * 3.14 / 4
131. RL = 287.058
132. T = TL + 273.15
133. pL = 100000
134. rohL = pL / RL / T
135.     p = I1 * U
136.     mLuft1 = (p * 2 * AL ^ 2 * rohL ^ 2) ^ (1 / 3)
137.     p = I2 * U
138.     mLuft2 = (p * 2 * AL ^ 2 * rohL ^ 2) ^ (1 / 3)
139.     p = I3 * U
140.     mLuft3 = (p * 2 * AL ^ 2 * rohL ^ 2) ^ (1 / 3)
141.     p = I4 * U
142.     mLuft4 = (p * 2 * AL ^ 2 * rohL ^ 2) ^ (1 / 3)
143.     p = I5 * U
144.     mLuft5 = (p * 2 * AL ^ 2 * rohL ^ 2) ^ (1 / 3)
145.     p = I6 * U
146.     mLuft6 = (p * 2 * AL ^ 2 * rohL ^ 2) ^ (1 / 3)
147.
148. mLuft = mLuft1 + mLuft2 + mLuft3 + mLuft4 + mLuft5 + mLuft6
149.
150. End Function
151.
152. Function MitLogTDiff(TK, TA, Tu)
153. 'Berechnung der idealen mittleren logarithmischen
   Temperaturdifferenz im LuKo
154.
155. 'TK: Temperatur im LuKo [°C]
156. 'TA: Temperatur der Abluft [°C]
157. 'Tu: Temperatur der Zuluft [°C]
158.
```

```
159. MitLogTDiff = (TA - Tu) / Application.WorksheetFunction.Ln((TK -
   Tu) / (TK - TA))
160.
161. End Function
```

Damit sind alle Größen für den Gütefaktor bestimmt. Er wird berechnet aus:

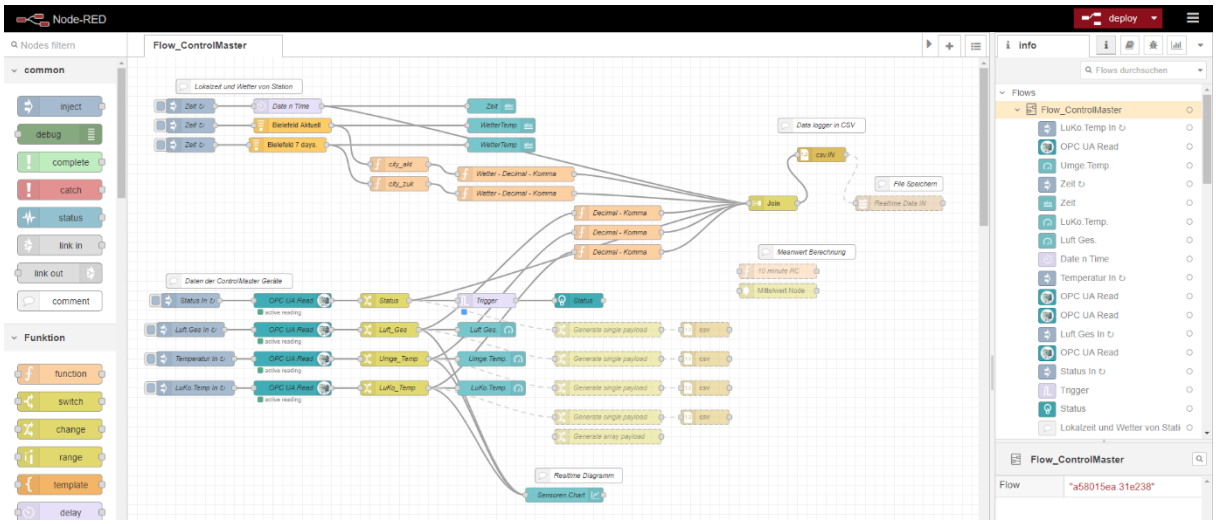
$$G = \frac{\dot{Q}}{\Delta T_{m,log,ideal}}$$

Mit:
G: Gütefaktor
$\dot{Q}$: Abgeführte Wärme durch LuKo (berechnet durch Funktion „*EntLuKo*“)
$\Delta T_{m,log,ideal}$: Ideale mittlere logarithmische Temperaturdifferenz (berechnet durch Funktion „*MitLogTDiff*“)

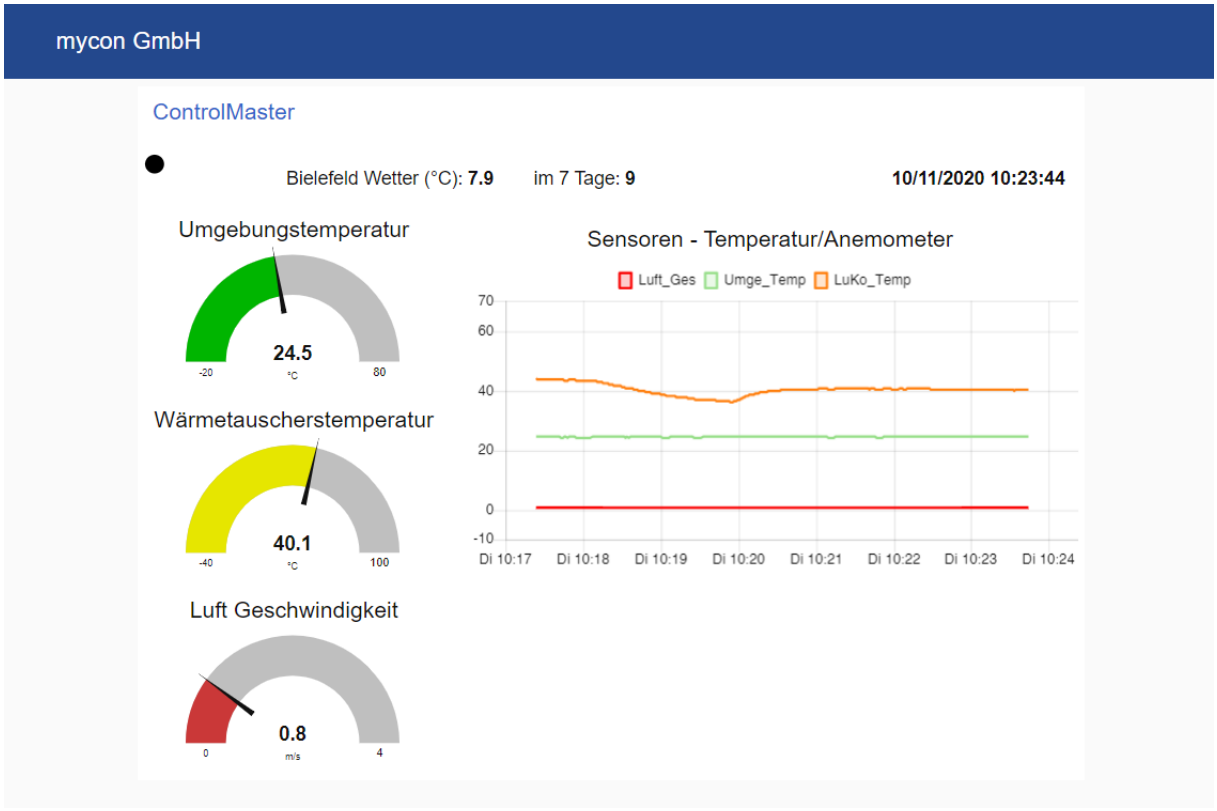
Schnittstelle zwischen ControlMaster und CloudSpeicher

Node-RED ist ein Programmierwerkzeug, mit dem Hardware-Geräte, APIs und Online-Dienste auf neue und interessante Weise miteinander verdrahtet werden können. Es stellt einen browserbasierten Editor zur Verfügung, mit dem es einfach ist, Flüsse zu verkabeln, indem die breite Palette von Knoten in der Palette verwendet wird, die mit einem einzigen Mausklick zur Laufzeit bereitgestellt werden können. [3]

Die folgende Abbildung zeigt ein Beispielsbild für den Node-red ControlMaster Flow. In diesem Flow werden Variablen mit Hilfe der Siemens Steuerung gemessen und über ein Cloud-System in einer CSV Datei gespeichert. Zusätzliche Daten werden aus einer CSV Datei ausgelesen und mit den gemessenen Daten verknüpft. Eine Auswertung dieser Daten durch den Algorithmus liefert dann das Ergebnis.



Eine Überwachung der Ergebnisse und der Messwerte ist über ein User Interface (UI) möglich. Dies ist beispielhaft in der folgenden Abbildung zu sehen, in der einige Messwerte veranschaulicht werden. Die Umgebungstemperatur beschreibt die Temperatur der Luft am Luft-Eintritt des Kondensators. Die Wärmetauschertemperatur ist die Temperatur der Abluft des Kondensators. Die Luftgeschwindigkeit entspricht der Geschwindigkeit, mit der die Luft den Kondensator verlässt. Zusätzlich ist eine Überwachung der Werte von nahegelegenen Wetterstationen möglich. Dies bietet die Möglichkeit zum Abgleich aktueller Werte und zur Einbindung von Prognose-Daten zur Festlegung zukünftiger Reinigungstermine.




```
1. /*Copyright <2020> <mycon GmbH>*/
2. /*<author: H. Gandhi>*/
3.
4. /*Permission is hereby granted, free of charge, to any person
   obtaining a copy of this software and associated documentation files
   (the "Software"), to deal in the Software without restriction,
   including without limitation the rights to use, copy, modify, merge,
   publish, distribute, sublicense, and/or sell copies of the Software,
   and to permit persons to whom the Software is furnished to do so,
   subject to the following conditions:*/
5.
6. /*The above copyright notice and this permission notice shall be
   included in all copies or substantial portions of the Software.*/
7.
8. /*THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
   EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
   MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
   NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
   BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
   ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
   CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
   SOFTWARE.*/
9.
10.
```

Senner Str. 156
D-33659 Bielefeld
Tel. +49 521 40 30 90
Fax +49 521 40 24 82
www.mycon-germany.com
info@mycon-germany.com


```
[
    {
        "id": "a58015ea.31e238",
        "type": "tab",
        "label": "Flow_ControlMaster",
        "disabled": false,
        "info": ""
    },
    {
        "id": "417c05f9.aa0dcc",
        "type": "inject",
        "z": "a58015ea.31e238",
        "name": "LuKo.Temp In",
        "props": [
            {
                "p": "payload"
            },
            {
                "p": "topic",
                "vt": "str"
            }
        ],
        "repeat": "1",
        "crontab": "",
        "once": true,
        "onceDelay": "1",
        "topic": "ns=4;i=3;datatype=Real",
        "payload": "",
        "payloadType": "date",
        "x": 140,
        "y": 680,
        "wires": [
            [
                "55ffcb16.643ee4"
            ]
        ]
    },
    {
        "id": "55ffcb16.643ee4",
        "type": "OpcUa-Client",
        "z": "a58015ea.31e238",
        "endpoint": "23572188.8c679e",
        "action": "read",
        "deadbandtype": "a",
        "deadbandvalue": 1,
        "time": 10,
        "timeUnit": "s",
        "certificate": "n",
        "localfile": "",
        "securitymode": "None",
        "securitypolicy": "None",
        "name": "OPC UA Read",
        "x": 340,
        "y": 680,
```

```
55.          "wires": [
56.              [
57.                  "7e90b1bf.3d5ef"
58.              ]
59.          ]
60.      },
61.      {
62.          "id": "18968819.1202b8",
63.          "type": "ui_gauge",
64.          "z": "a58015ea.31e238",
65.          "name": "Umge.Temp.",
66.          "group": "9e5ebe31.7533b",
67.          "order": 6,
68.          "width": 5,
69.          "height": 3,
70.          "gtype": "gage",
71.          "title": "Umgebungstemperatur",
72.          "label": "°C",
73.          "format": "{{msg.payload.toFixed(1)}}",
74.          "min": "-20",
75.          "max": "80",
76.          "colors": [
77.              "#00b500",
78.              "#e6e600",
79.              "#ca3838"
80.          ],
81.          "seg1": "27",
82.          "seg2": "50",
83.          "x": 750,
84.          "y": 620,
85.          "wires": []
86.      },
87.      {
88.          "id": "ddb8882.57fee78",
89.          "type": "inject",
90.          "z": "a58015ea.31e238",
91.          "name": "Zeit",
92.          "props": [
93.              {
94.                  "p": "payload"
95.              }
96.          ],
97.          "repeat": "1",
98.          "crontab": "",
99.          "once": true,
100.         "onceDelay": "1",
101.         "topic": "",
102.         "payload": "",
103.         "payloadType": "date",
104.         "x": 130,
105.         "y": 100,
106.         "wires": [
107.             [
108.                 "67d7333.18d24cc"
```

```
109.                ]
110.            ],
111.            "outputLabels": [
112.                "Zeit"
113.            ]
114.        },
115.        {
116.            "id": "db50959c.5feea8",
117.            "type": "ui_text",
118.            "z": "a58015ea.31e238",
119.            "group": "9e5ebe31.7533b",
120.            "order": 5,
121.            "width": 5,
122.            "height": 1,
123.            "name": "Zeit",
124.            "label": "",
125.            "format": "{{value}}",
126.            "layout": "row-right",
127.            "x": 750,
128.            "y": 100,
129.            "wires": []
130.        },
131.        {
132.            "id": "311c0dcf.fddb32",
133.            "type": "ui_gauge",
134.            "z": "a58015ea.31e238",
135.            "name": "LuKo.Temp.",
136.            "group": "9e5ebe31.7533b",
137.            "order": 8,
138.            "width": 5,
139.            "height": 3,
140.            "gtype": "gage",
141.            "title": "Wärmetauscherstemperatur",
142.            "label": "°C",
143.            "format": "{{msg.payload.toFixed(1)}}",
144.            "min": "-40",
145.            "max": "100",
146.            "colors": [
147.                "#00b500",
148.                "#e6e600",
149.                "#ca3838"
150.            ],
151.            "seg1": "35",
152.            "seg2": "60",
153.            "x": 750,
154.            "y": 680,
155.            "wires": []
156.        },
157.        {
158.            "id": "ae6e9688.968948",
159.            "type": "ui_gauge",
160.            "z": "a58015ea.31e238",
161.            "name": "Luft Ges.",
162.            "group": "9e5ebe31.7533b",
```

```
        "order": 9,
        "width": 5,
        "height": 3,
        "gtype": "gage",
        "title": "Luft Geschwindigkeit",
        "label": "m/s",
        "format": "{{msg.payload.toFixed(1)}}",
        "min": "0",
        "max": "4",
        "colors": [
            "#ca3838",
            "#e6e600",
            "#00b500"
        ],
        "seg1": "1,2",
        "seg2": "2",
        "x": 740,
        "y": 560,
        "wires": []
    },
    {
        "id": "67d7333.18d24cc",
        "type": "moment",
        "z": "a58015ea.31e238",
        "name": "Date n Time",
        "topic": "Date",
        "input": "",
        "inputType": "date",
        "inTz": "Europe/Berlin",
        "adjAmount": 0,
        "adjType": "days",
        "adjDir": "add",
        "format": "DD/MM/yyyy HH:mm:ss",
        "locale": "de-DE",
        "output": "",
        "outputType": "msg",
        "outTz": "Europe/Berlin",
        "x": 330,
        "y": 100,
        "wires": [
            [
                "db50959c.5feea8",
                "e4aecf4b.c6642"
            ]
        ]
    },
    {
        "id": "784638d4.94b568",
        "type": "inject",
        "z": "a58015ea.31e238",
        "name": "Temperatur In",
        "props": [
            {
                "p": "payload"
```

```
            },
            {
                "p": "topic",
                "vt": "str"
            }
        ],
        "repeat": "1",
        "crontab": "",
        "once": true,
        "onceDelay": "1",
        "topic": "ns=4;i=4;datatype=Real",
        "payload": "",
        "payloadType": "date",
        "x": 140,
        "y": 620,
        "wires": [
            [
                "c4a9eccb.766cd"
            ]
        ]
    },
    {
        "id": "c4a9eccb.766cd",
        "type": "OpcUa-Client",
        "z": "a58015ea.31e238",
        "endpoint": "23572188.8c679e",
        "action": "read",
        "deadbandtype": "a",
        "deadbandvalue": 1,
        "time": 10,
        "timeUnit": "s",
        "certificate": "n",
        "localfile": "",
        "securitymode": "None",
        "securitypolicy": "None",
        "name": "OPC UA Read",
        "x": 340,
        "y": 620,
        "wires": [
            [
                "947138ca.da8658"
            ]
        ]
    },
    {
        "id": "3fd1d707.3ce8d8",
        "type": "OpcUa-Client",
        "z": "a58015ea.31e238",
        "endpoint": "23572188.8c679e",
        "action": "read",
        "deadbandtype": "a",
        "deadbandvalue": 1,
        "time": 10,
        "timeUnit": "s",
```

```
271.         "certificate": "n",
272.         "localfile": "",
273.         "securitymode": "None",
274.         "securitypolicy": "None",
275.         "name": "OPC UA Read",
276.         "x": 340,
277.         "y": 560,
278.         "wires": [
279.             [
280.                 "59e8052b.27843c"
281.             ]
282.         ]
283.     },
284.     {
285.         "id": "7ac93503.e81dbc",
286.         "type": "inject",
287.         "z": "a58015ea.31e238",
288.         "name": "Luft.Ges In",
289.         "props": [
290.             {
291.                 "p": "payload"
292.             },
293.             {
294.                 "p": "topic",
295.                 "vt": "str"
296.             }
297.         ],
298.         "repeat": "1",
299.         "crontab": "",
300.         "once": true,
301.         "onceDelay": "1",
302.         "topic": "ns=4;i=2;datatype=Real",
303.         "payload": "",
304.         "payloadType": "date",
305.         "x": 130,
306.         "y": 560,
307.         "wires": [
308.             [
309.                 "3fd1d707.3ce8d8"
310.             ]
311.         ]
312.     },
313.     {
314.         "id": "b668ade7.2a723",
315.         "type": "OpcUa-Client",
316.         "z": "a58015ea.31e238",
317.         "endpoint": "23572188.8c679e",
318.         "action": "read",
319.         "deadbandtype": "a",
320.         "deadbandvalue": 1,
321.         "time": 10,
322.         "timeUnit": "s",
323.         "certificate": "n",
324.         "localfile": "",
```

```
        "securitymode": "None",
        "securitypolicy": "None",
        "name": "OPC UA Read",
        "x": 340,
        "y": 500,
        "wires": [
            [
                "fc64b7e2.7c5378"
            ]
        ]
    },
    {
        "id": "44385311.ef22bc",
        "type": "inject",
        "z": "a58015ea.31e238",
        "name": "Status In",
        "props": [
            {
                "p": "payload"
            },
            {
                "p": "topic",
                "vt": "str"
            }
        ],
        "repeat": "1",
        "crontab": "",
        "once": true,
        "onceDelay": "0.1",
        "topic": "ns=4;i=5;datatype=Boolean",
        "payload": "",
        "payloadType": "date",
        "x": 130,
        "y": 500,
        "wires": [
            [
                "b668ade7.2a723"
            ]
        ]
    },
    {
        "id": "83533825.942c28",
        "type": "trigger",
        "z": "a58015ea.31e238",
        "name": "Trigger",
        "op1": "lime",
        "op2": "black",
        "op1type": "str",
        "op2type": "str",
        "duration": "500",
        "extend": true,
        "units": "ms",
        "reset": "",
        "bytopic": "all",
```

```
379.         "topic": "topic",
380.         "outputs": 1,
381.         "x": 740,
382.         "y": 500,
383.         "wires": [
384.             [
385.                 "967f7a4e.08ead8"
386.             ]
387.         ]
388.     },
389.     {
390.         "id": "967f7a4e.08ead8",
391.         "type": "ui_template",
392.         "z": "a58015ea.31e238",
393.         "group": "9e5ebe31.7533b",
394.         "name": "Status",
395.         "order": 1,
396.         "width": 1,
397.         "height": 1,
398.         "format": "<svg>\n<circle cx=\"8\" cy=\"8\" r=\"8\"
   style=\"stroke: none; fill: {{msg.payload}};\"/>\n</svg>",
399.         "storeOutMessages": true,
400.         "fwdInMessages": true,
401.         "resendOnRefresh": false,
402.         "templateScope": "local",
403.         "x": 930,
404.         "y": 500,
405.         "wires": [
406.             []
407.         ],
408.         "outputLabels": [
409.             "0"
410.         ],
411.         "icon": "node-red/light.svg"
412.     },
413.     {
414.         "id": "2786320.87d8cce",
415.         "type": "comment",
416.         "z": "a58015ea.31e238",
417.         "name": "Lokalzeit und Wetter von Station",
418.         "info": "",
419.         "x": 230,
420.         "y": 60,
421.         "wires": []
422.     },
423.     {
424.         "id": "9366d066.3d039",
425.         "type": "comment",
426.         "z": "a58015ea.31e238",
427.         "name": "Daten der ControlMaster Geräte",
428.         "info": "",
429.         "x": 210,
430.         "y": 460,
431.         "wires": []
```

```
    },
    {
        "id": "3375bd06.022b12",
        "type": "dwdweather",
        "z": "a58015ea.31e238",
        "name": "Bielefeld Aktuell",
        "mosmixStation": "10326",
        "lookAheadHours": "0",
        "additionalFields": "",
        "repeat": "0",
        "x": 340,
        "y": 140,
        "wires": [
            [
                "966bf7bc.a1cd98",
                "2a47ccf3.a5dd14"
            ]
        ]
    },
    {
        "id": "2a47ccf3.a5dd14",
        "type": "ui_text",
        "z": "a58015ea.31e238",
        "group": "9e5ebe31.7533b",
        "order": 2,
        "width": 5,
        "height": 1,
        "name": "WetterTemp",
        "label": "Bielefeld Wetter (°C): ",
        "format": "{{msg.payload.tempc}}",
        "layout": "row-right",
        "x": 770,
        "y": 140,
        "wires": []
    },
    {
        "id": "dd77af84.76f8b",
        "type": "dwdweather",
        "z": "a58015ea.31e238",
        "name": "Bielefeld 7 days.",
        "mosmixStation": "10326",
        "lookAheadHours": "168",
        "additionalFields": "",
        "repeat": "0",
        "x": 330,
        "y": 180,
        "wires": [
            [
                "73d97025.09a64",
                "3703af50.c34e4"
            ]
        ]
    },
    {
```

```
        "id": "73d97025.09a64",
        "type": "ui_text",
        "z": "a58015ea.31e238",
        "group": "9e5ebe31.7533b",
        "order": 3,
        "width": 3,
        "height": 1,
        "name": "WetterTemp",
        "label": "im 7 Tage: ",
        "format": "{{msg.payload.tempc}}",
        "layout": "row-left",
        "x": 770,
        "y": 180,
        "wires": []
    },
    {
        "id": "9a87614b.6deba",
        "type": "inject",
        "z": "a58015ea.31e238",
        "name": "Zeit",
        "props": [
            {
                "p": "payload"
            }
        ],
        "repeat": "1",
        "crontab": "",
        "once": true,
        "onceDelay": "1",
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "x": 130,
        "y": 140,
        "wires": [
            [
                "3375bd06.022b12"
            ]
        ],
        "outputLabels": [
            "Zeit"
        ]
    },
    {
        "id": "5ac0e613.43cb98",
        "type": "inject",
        "z": "a58015ea.31e238",
        "name": "Zeit",
        "props": [
            {
                "p": "payload"
            }
        ],
        "repeat": "1",
```

```
        "crontab": "",
        "once": true,
        "onceDelay": "1",
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "x": 130,
        "y": 180,
        "wires": [
            [
                "dd77af84.76f8b"
            ]
        ],
        "outputLabels": [
            "Zeit"
        ]
    },
    {
        "id": "3f83f7e1.2c0f98",
        "type": "comment",
        "z": "a58015ea.31e238",
        "name": "Data logger in CSV ",
        "info": "",
        "x": 1430,
        "y": 140,
        "wires": []
    },
    {
        "id": "5a5acd3f.d4aa24",
        "type": "comment",
        "z": "a58015ea.31e238",
        "name": "Realtime Diagramm",
        "info": "",
        "x": 930,
        "y": 860,
        "wires": []
    },
    {
        "id": "11f7e350.9daafd",
        "type": "ui_chart",
        "z": "a58015ea.31e238",
        "name": "Sensoren.Chart",
        "group": "9e5ebe31.7533b",
        "order": 7,
        "width": 10,
        "height": 6,
        "label": "Sensoren - Temperatur/Anemometer",
        "chartType": "line",
        "legend": "true",
        "xformat": "dd HH:mm",
        "interpolate": "linear",
        "nodata": "",
        "dot": false,
        "ymin": "-10",
```

```
        "ymax": "70",
        "removeOlder": "1",
        "removeOlderPoints": "",
        "removeOlderUnit": "3600",
        "cutout": 0,
        "useOneColor": false,
        "useUTC": false,
        "colors": [
            "#fa0000",
            "#98df8a",
            "#ff7f0e",
            "#2ca02c",
            "#acc7e8",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "useOldStyle": false,
        "outputs": 1,
        "x": 900,
        "y": 900,
        "wires": [
            []
        ]
    },
    {
        "id": "59e8052b.27843c",
        "type": "change",
        "z": "a58015ea.31e238",
        "name": "Luft_Ges",
        "rules": [
            {
                "t": "set",
                "p": "topic",
                "pt": "msg",
                "to": "Luft_Ges",
                "tot": "str"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 540,
        "y": 560,
        "wires": [
            [
                "ae6e9688.968948",
                "11f7e350.9daafd",
                "bc85953e.394f18",
                "ab439537.4dffc8"
            ]
```

```
            ]
        },
        {
            "id": "947138ca.da8658",
            "type": "change",
            "z": "a58015ea.31e238",
            "name": "Umge_Temp",
            "rules": [
                {
                    "t": "set",
                    "p": "topic",
                    "pt": "msg",
                    "to": "Umge_Temp",
                    "tot": "str"
                }
            ],
            "action": "",
            "property": "",
            "from": "",
            "to": "",
            "reg": false,
            "x": 550,
            "y": 620,
            "wires": [
                [
                    "18968819.1202b8",
                    "11f7e350.9daafd",
                    "72e08d78.913164",
                    "b878899f.f8de28"
                ]
            ]
        },
        {
            "id": "7e90b1bf.3d5ef",
            "type": "change",
            "z": "a58015ea.31e238",
            "name": "LuKo_Temp",
            "rules": [
                {
                    "t": "set",
                    "p": "topic",
                    "pt": "msg",
                    "to": "LuKo_Temp",
                    "tot": "str"
                }
            ],
            "action": "",
            "property": "",
            "from": "",
            "to": "",
            "reg": false,
            "x": 550,
            "y": 680,
            "wires": [
```

```
            [
                "311c0dcf.fddb32",
                "11f7e350.9daafd",
                "59cdfee4.980bd",
                "e2b7aaf2.d52488"
            ]
        ]
    },
    {
        "id": "fc64b7e2.7c5378",
        "type": "change",
        "z": "a58015ea.31e238",
        "name": "Status",
        "rules": [
            {
                "t": "set",
                "p": "topic",
                "pt": "msg",
                "to": "Status",
                "tot": "str"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 530,
        "y": 500,
        "wires": [
            [
                "83533825.942c28",
                "960f7cbd.64b7a",
                "e4aecf4b.c6642"
            ]
        ]
    },
    {
        "id": "bc85953e.394f18",
        "type": "change",
        "z": "a58015ea.31e238",
        "d": true,
        "name": "Generate single payload",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "msg",
                "to": "{ \"Luft_Ges\":msg.payload}",
                "tot": "jsonata"
            }
        ],
        "action": "",
        "property": "",
```

```
        "from": "",
        "to": "",
        "reg": false,
        "x": 990,
        "y": 620,
        "wires": [
            [
                "c4580516.492c48"
            ]
        ]
    },
    {
        "id": "c4580516.492c48",
        "type": "csv",
        "z": "a58015ea.31e238",
        "d": true,
        "name": "",
        "sep": ",",
        "hdrin": "",
        "hdrout": "none",
        "multi": "one",
        "ret": "\\n",
        "temp": "Luft_Ges",
        "skip": "0",
        "strings": true,
        "include_empty_strings": false,
        "include_null_values": false,
        "x": 1190,
        "y": 620,
        "wires": [
            []
        ]
    },
    {
        "id": "72e08d78.913164",
        "type": "change",
        "z": "a58015ea.31e238",
        "d": true,
        "name": "Generate single payload",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "msg",
                "to": "{ \"Umge_Temp\":msg.payload}",
                "tot": "jsonata"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 990,
```

```
        "y": 680,
        "wires": [
            [
                "5ad58b7d.415914"
            ]
        ]
    },
    {
        "id": "5ad58b7d.415914",
        "type": "csv",
        "z": "a58015ea.31e238",
        "d": true,
        "name": "",
        "sep": ",",
        "hdrin": "",
        "hdrout": "none",
        "multi": "one",
        "ret": "\\n",
        "temp": "Umge_Temp",
        "skip": "0",
        "strings": true,
        "include_empty_strings": false,
        "include_null_values": false,
        "x": 1190,
        "y": 680,
        "wires": [
            []
        ]
    },
    {
        "id": "59cdfee4.980bd",
        "type": "change",
        "z": "a58015ea.31e238",
        "d": true,
        "name": "Generate single payload",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "msg",
                "to": "{ \"LuKo_Temp\":msg.payload}",
                "tot": "jsonata"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 990,
        "y": 740,
        "wires": [
            [
                "956af12d.4f9f6"
```

```
            ]
        ]
    },
    {
        "id": "956af12d.4f9f6",
        "type": "csv",
        "z": "a58015ea.31e238",
        "d": true,
        "name": "",
        "sep": ",",
        "hdrin": "",
        "hdrout": "none",
        "multi": "one",
        "ret": "\\n",
        "temp": "LuKo_Temp",
        "skip": "0",
        "strings": true,
        "include_empty_strings": false,
        "include_null_values": false,
        "x": 1190,
        "y": 740,
        "wires": [
            []
        ]
    },
    {
        "id": "960f7cbd.64b7a",
        "type": "change",
        "z": "a58015ea.31e238",
        "d": true,
        "name": "Generate single payload",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "msg",
                "to": "{ \"Status\":msg.payload}",
                "tot": "jsonata"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 990,
        "y": 560,
        "wires": [
            [
                "d678b19b.b2de4"
            ]
        ]
    },
    {
```

```
        "id": "d678b19b.b2de4",
        "type": "csv",
        "z": "a58015ea.31e238",
        "d": true,
        "name": "",
        "sep": ",",
        "hdrin": "",
        "hdrout": "none",
        "multi": "one",
        "ret": "\\n",
        "temp": "Status",
        "skip": "0",
        "strings": true,
        "include_empty_strings": false,
        "include_null_values": false,
        "x": 1190,
        "y": 560,
        "wires": [
            []
        ]
    },
    {
        "id": "1c126515.c8c26b",
        "type": "file",
        "z": "a58015ea.31e238",
        "name": "Realtime.Data.IN",
        "filename": "C:\\Users\\User\\InternServer - mycon
  GmbH\\ControlMaster\\Node Red\\realtime_log_data.csv",
        "appendNewline": false,
        "createDir": false,
        "overwriteFile": "false",
        "encoding": "utf8",
        "x": 1590,
        "y": 300,
        "wires": [
            []
        ],
        "icon": "node-red/leveldb.png"
    },
    {
        "id": "e4aecf4b.c6642",
        "type": "join",
        "z": "a58015ea.31e238",
        "name": "",
        "mode": "custom",
        "build": "object",
        "property": "payload",
        "propertyType": "msg",
        "key": "topic",
        "joiner": ";",
        "joinerType": "str",
        "accumulate": false,
        "timeout": "",
        "count": "7",
```

```
971.          "reduceRight": false,
972.          "reduceExp": "",
973.          "reduceInit": "",
974.          "reduceInitType": "",
975.          "reduceFixup": "",
976.          "x": 1330,
977.          "y": 300,
978.          "wires": [
979.              [
980.                  "3e949a39.4f25e6"
981.              ]
982.          ]
983.      },
984.      {
985.          "id": "ab439537.4dffc8",
986.          "type": "function",
987.          "z": "a58015ea.31e238",
988.          "name": "Decimal - Komma",
989.          "func": "msg.payload = msg.payload.toFixed(2); // Decimal
    conversion \n\n// Komma Format \nif(typeof msg.payload != \"string\")
    msg.payload = String(msg.payload);\nif(msg.payload.match(/\\./g)) {\n
    msg.payload = msg.payload.replace(/\\,/g, \".\");\n    msg.payload =
    msg.payload.replace(/\\.(?!.*\\.)/g, \",\");\n    \n} else {\n
    msg.payload = msg.payload.replace(/\\,/g, \".\");\n    \n}\n\nreturn
    msg;",
990.          "outputs": 1,
991.          "noerr": 0,
992.          "initialize": "",
993.          "finalize": "",
994.          "x": 1010,
995.          "y": 320,
996.          "wires": [
997.              [
998.                  "e4aecf4b.c6642"
999.              ]
1000.             ]
1001.         },
1002.         {
1003.             "id": "b7df2362.6b979",
1004.             "type": "function",
1005.             "z": "a58015ea.31e238",
1006.             "name": "Wetter - Decimal - Komma",
1007.             "func": "msg.payload = msg.payload.tempc; // Get
    only Temperature\nmsg.payload = msg.payload.toFixed(2); // Decimal
    conversion \n\n// Komma Format \nif(typeof msg.payload != \"string\")
    msg.payload = String(msg.payload);\nif(msg.payload.match(/\\./g)) {\n
    msg.payload = msg.payload.replace(/\\,/g, \".\");\n    msg.payload =
    msg.payload.replace(/\\.(?!.*\\.)/g, \",\");\n    \n} else {\n
    msg.payload = msg.payload.replace(/\\,/g, \".\");\n    \n}\n\nreturn
    msg;",
1008.             "outputs": 1,
1009.             "noerr": 0,
1010.             "initialize": "",
1011.             "finalize": "",
```

```
1012.                "x": 800,
1013.                "y": 240,
1014.                "wires": [
1015.                    [
1016.                        "e4aecf4b.c6642"
1017.                    ]
1018.                ]
1019.            },
1020.            {
1021.                "id": "3e949a39.4f25e6",
1022.                "type": "csv",
1023.                "z": "a58015ea.31e238",
1024.                "name": "csv.IN",
1025.                "sep": ";",
1026.                "hdrin": false,
1027.                "hdrout": "none",
1028.                "multi": "one",
1029.                "ret": "\\n",
1030.                "temp": "Date, city_akt, city_zuk, Luft_Ges,
   Umge_Temp, LuKo_Temp, Status,",
1031.                "skip": "0",
1032.                "strings": true,
1033.                "include_empty_strings": false,
1034.                "include_null_values": false,
1035.                "x": 1430,
1036.                "y": 200,
1037.                "wires": [
1038.                    [
1039.                        "1c126515.c8c26b"
1040.                    ]
1041.                ]
1042.            },
1043.            {
1044.                "id": "966bf7bc.a1cd98",
1045.                "type": "function",
1046.                "z": "a58015ea.31e238",
1047.                "name": "city_akt",
1048.                "func": "msg.topic=\"city_akt\";\nreturn msg;",
1049.                "outputs": 1,
1050.                "noerr": 0,
1051.                "initialize": "",
1052.                "finalize": "",
1053.                "x": 560,
1054.                "y": 220,
1055.                "wires": [
1056.                    [
1057.                        "b7df2362.6b979"
1058.                    ]
1059.                ]
1060.            },
1061.            {
1062.                "id": "3703af50.c34e4",
1063.                "type": "function",
1064.                "z": "a58015ea.31e238",
```

```
1065.                "name": "city_zuk",
1066.                "func": "msg.topic=\"city_zuk\";\nreturn msg;",
1067.                "outputs": 1,
1068.                "noerr": 0,
1069.                "initialize": "",
1070.                "finalize": "",
1071.                "x": 560,
1072.                "y": 260,
1073.                "wires": [
1074.                    [
1075.                        "eaa9f31e.6cfec"
1076.                    ]
1077.                ]
1078.            },
1079.            {
1080.                "id": "eaa9f31e.6cfec",
1081.                "type": "function",
1082.                "z": "a58015ea.31e238",
1083.                "name": "Wetter - Decimal - Komma",
1084.                "func": "msg.payload = msg.payload.tempc; // Get
   only Temperature\nmsg.payload = msg.payload.toFixed(2); // Decimal
   conversion \n\n// Komma Format \nif(typeof msg.payload != \"string\")
   msg.payload = String(msg.payload);\nif(msg.payload.match(/\\./g)) {\n
   msg.payload = msg.payload.replace(/\\,/g, \".\");\n    msg.payload =
   msg.payload.replace(/\\.(?!.*\\.)/g, \",\");\n    \n} else {\n
   msg.payload = msg.payload.replace(/\\,/g, \".\");\n    \n}\n\nreturn
   msg;",
1085.                "outputs": 1,
1086.                "noerr": 0,
1087.                "initialize": "",
1088.                "finalize": "",
1089.                "x": 800,
1090.                "y": 280,
1091.                "wires": [
1092.                    [
1093.                        "e4aecf4b.c6642"
1094.                    ]
1095.                ]
1096.            },
1097.            {
1098.                "id": "4576d48b.c9bbcc",
1099.                "type": "change",
1100.                "z": "a58015ea.31e238",
1101.                "d": true,
1102.                "name": "Generate array payload",
1103.                "rules": [
1104.                    {
1105.                        "t": "set",
1106.                        "p": "payload",
1107.                        "pt": "msg",
1108.                        "to": "[\t    {
   \"Date\":msg.payload.Date,\"city_akt\":msg.payload.city_akt,\"city_zu
   k\":msg.payload.city_zuk,\"Luft_Ges\":msg.payload.Luft_Ges,\"Umge_Tem
```

```
   p\":msg.payload.Umge_Temp,\"LuKo_Temp\":msg.payload.LuKo_Temp,\"Statu
   s\":msg.payload.Status}\t]",
                        "tot": "jsonata"
                    }
                ],
                "action": "",
                "property": "",
                "from": "",
                "to": "",
                "reg": false,
                "x": 990,
                "y": 780,
                "wires": [
                    []
                ]
            },
            {
                "id": "b878899f.f8de28",
                "type": "function",
                "z": "a58015ea.31e238",
                "name": "Decimal - Komma",
                "func": "msg.payload = msg.payload.toFixed(2); //
   Decimal conversion \n\n// Komma Format \nif(typeof msg.payload !=
   \"string\") msg.payload =
   String(msg.payload);\nif(msg.payload.match(/\\./g)) {\n
   msg.payload = msg.payload.replace(/\\,/g, \".\");\n    msg.payload =
   msg.payload.replace(/\\.(?!.*\\.)/g, \",\");\n    \n} else {\n
   msg.payload = msg.payload.replace(/\\,/g, \".\");\n    \n}\n\nreturn
   msg;",
                "outputs": 1,
                "noerr": 0,
                "initialize": "",
                "finalize": "",
                "x": 1010,
                "y": 360,
                "wires": [
                    [
                        "e4aecf4b.c6642"
                    ]
                ]
            },
            {
                "id": "e2b7aaf2.d52488",
                "type": "function",
                "z": "a58015ea.31e238",
                "name": "Decimal - Komma",
                "func": "msg.payload = msg.payload.toFixed(2); //
   Decimal conversion \n\n// Komma Format \nif(typeof msg.payload !=
   \"string\") msg.payload =
   String(msg.payload);\nif(msg.payload.match(/\\./g)) {\n
   msg.payload = msg.payload.replace(/\\,/g, \".\");\n    msg.payload =
   msg.payload.replace(/\\.(?!.*\\.)/g, \",\");\n    \n} else {\n
   msg.payload = msg.payload.replace(/\\,/g, \".\");\n    \n}\n\nreturn
   msg;",
```

```
                "outputs": 1,
                "noerr": 0,
                "initialize": "",
                "finalize": "",
                "x": 1010,
                "y": 400,
                "wires": [
                    [
                        "e4aecf4b.c6642"
                    ]
                ]
            },
            {
                "id": "77aebcaf.607cc4",
                "type": "function",
                "z": "a58015ea.31e238",
                "d": true,
                "name": "10 minute RC",
                "func": "// Applies a simple RC low pass filter to
   incoming payload values\nvar tc = 10*60*1000;       // time constant
   in milliseconds\n\nvar lastValue = context.get('lastValue');\nif
   (typeof lastValue == \"undefined\") lastValue = msg.payload;\nvar
   lastTime = context.get('lastTime') || null;\nvar now = new
   Date();\nvar currentValue = msg.payload;\nif (lastTime === null) {\n
   // first time through\n    newValue = currentValue;\n} else {\n
   var dt = now - lastTime;\n    var newValue;\n    \n    if (dt > 0)
   {\n        var dtotc = dt / tc;\n        newValue = lastValue * (1 -
   dtotc) + currentValue * dtotc;\n    } else {\n        // no time has
   elapsed leave output the same as last time\n        newValue =
   lastValue;\n    }\n}\ncontext.set('lastValue',
   newValue);\ncontext.set('lastTime', now);\n\nmsg.payload =
   newValue;\nreturn msg;",
                "outputs": 1,
                "noerr": 0,
                "initialize": "",
                "finalize": "",
                "x": 1340,
                "y": 440,
                "wires": [
                    []
                ]
            },
            {
                "id": "290d54e3.23441c",
                "type": "aggregator",
                "z": "a58015ea.31e238",
                "d": true,
                "name": "Mittelwert Node",
                "topic": "",
                "intervalCount": 1,
                "intervalUnits": "m",
                "submitIncompleteInterval": true,
                "submitPerTopic": false,
                "aggregationType": "mean",
```

```
                "x": 1340,
                "y": 480,
                "wires": [
                    []
                ]
            },
            {
                "id": "f3ed0934.87bde8",
                "type": "comment",
                "z": "a58015ea.31e238",
                "name": "Meanwert Berechnung",
                "info": "",
                "x": 1400,
                "y": 400,
                "wires": []
            },
            {
                "id": "50e5aa8f.865864",
                "type": "comment",
                "z": "a58015ea.31e238",
                "name": "File Speichern",
                "info": "",
                "x": 1610,
                "y": 260,
                "wires": []
            },
            {
                "id": "23572188.8c679e",
                "type": "OpcUa-Endpoint",
                "z": "",
                "endpoint": "opc.tcp://192.168.0.155:4840",
                "secpol": "None",
                "secmode": "None",
                "login": false
            },
            {
                "id": "9e5ebe31.7533b",
                "type": "ui_group",
                "z": "",
                "name": "ControlMaster",
                "tab": "b134b5b0.acd8f8",
                "order": 1,
                "disp": true,
                "width": "15",
                "collapse": false
            },
            {
                "id": "b134b5b0.acd8f8",
                "type": "ui_tab",
                "z": "",
                "name": "mycon GmbH",
                "icon": "dashboard",
                "disabled": false,
                "hidden": false
```

```
1242.            }
1243.
1244.
```

Literaturverzeichnis

[1] W. Wagner, W. Wagner, and A. Kruse, *Properties of Water and Steam: The Industrial Standard IAPWS-IF97 for the Thermodynamic Properties and Supplementary Equations for Other Properties : Tables Based on These Equations*. Springer-Verlag, 1998.

[2] G. Cordes, “Die Turbine bei Abweichung vom Auslegungszustand,” in *Strömungstechnik der gasbeaufschlagten Axialturbine: unter besonderer Berücksichtigung der Strahltriebwerksturbine*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1963, pp. 234–274.

[3] “Node-RED Homepage.” www.nodered.org (accessed Nov. 25, 2020).