Python data two

Daten einlesen, anschauen lassen, ...

engelbert gruber

16.12.2020
Rev. 0.3

INTRODUCTION

EKG-Daten während der Erfassung aus- bewerten.

Damit ich nicht vorher ein EKG-Messgerät bauen muss, nehme ich Datensätze aus dem INternet. TELE ECG Database: 250 telehealth ECG records , ...

MATERIALS

1. Ein Dataset von TELE ECG
2. PC mit python3
3. Interesse

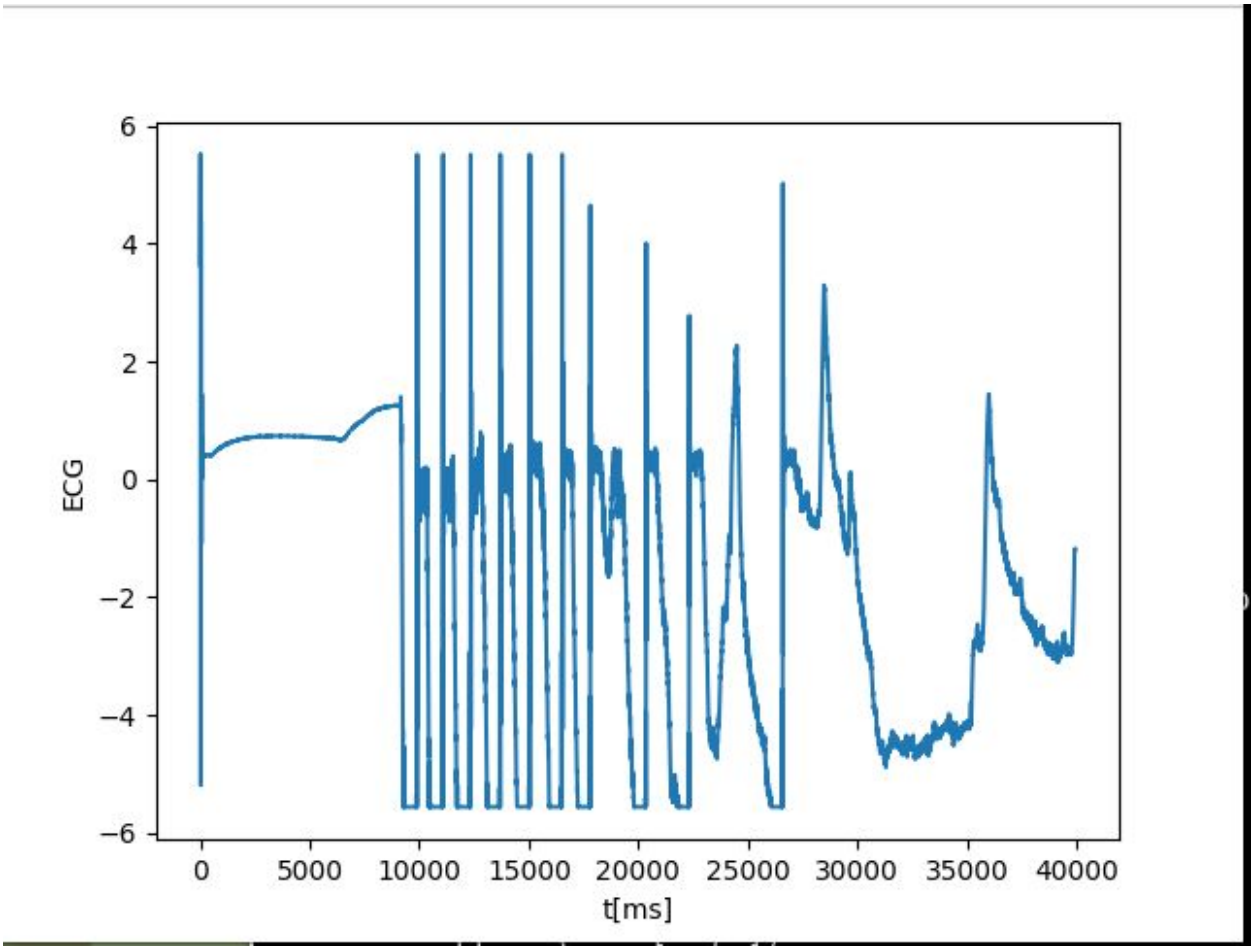
DIE DATEN ...

... kommen vom Analog Digital Converter, der wandelt den analogen Spannungswert an seinem Eingang in einen digitalen Wert um.

Für die Analysesoftware ist aber dasselbe ob die Daten vom ADC oder aus der Datei

kommen. Das Lesen der Daten aus einer Datei beschleunigt die Softwareentwicklung aber sehr, weil wir mehrere Testfälle simulieren können.

Die Daten aus der Datei “100_120.dat” sehen folgendermaßen aus



Zeit von Maxima zu Maxima

Der Maximalwert ist : 5.510786

```
import csv
max_ecg = -10000 # der nächste echte Messwert ist groesser
# die csv-Datei "100_120.dat" zeilenweise durchgehen
for line in csv.reader( open("100_120.dat") ):
    ecg = float(line[0]) # die erste Spalte ist der EKG-Wert
    if ecg > max_ecg:
        max_ecg = ecg
```

```
print(max_ecg)
```

Das nützt uns nichts, wir brauchen die Zeitpunkte der Umkehrpunkte.
Bei 500Hz Samplefreuqenz ist der Abstand zwischen zwei Messpunkten 2ms.

```
import csv
max_ecg = -10000 # der nächste echte Messwert ist groesser
time_in_ms = 0
# die csv-Datei "100_120.dat" zeilenweise durchgehen
for line in csv.reader( open("100_120.dat") ):
    ecg = float(line[0]) # die erste Spalte ist der EKG-Wert
    if ecg > max_ecg:
        max_ecg = ecg
        print(time_in_ms, ecg)
    time_in_ms += 2      # 500Hz
print(max_ecg)
```

Wir haben Glück (oder auch nicht), das Ergebnis ist kurz

```
0 3.63587
2 3.70913
4 4.007597
6 4.498711
8 5.236739
10 5.510786
5.510786
```

Noch einmal überlegen: **Wir wollen die Zeit zwischen zwei Maxima.**

1. Ein Maxima ist erreicht, wenn der aktuelle Wert kleiner als der vorhergehende ist.
2. Erst wenn dir danach ein Minima erreicht haben, suchen wir wieder ein Maxima.
3. Das Minima ist erreicht, wenn der aktuelle Wert größer ist als der vorhergehende.

```
import csv
rising = True
prev_ecg = -10000 # der nächste echte Messwert ist groesser
time_in_ms = 0
# die csv-Datei "100_120.dat" zeilenweise durchgehen
for line in csv.reader( open("100_120.dat") ):
```

```
    ecg = float(line[0]) # die erste Spalte ist der EKG-Wert
    if rising:
        if ecg < prev_ecg:  # Wendepunkt oben
            print(time_in_ms, ecg)
            rising = False
    else:
        if ecg > prev_ecg:  # Wendepunkt unten
            rising = True
    prev_ecg = ecg
    time_in_ms += 2   # 500Hz, 2ms pro Schritt
```

Das sind dann leider 1758 Punkte.

Ich will das sehen

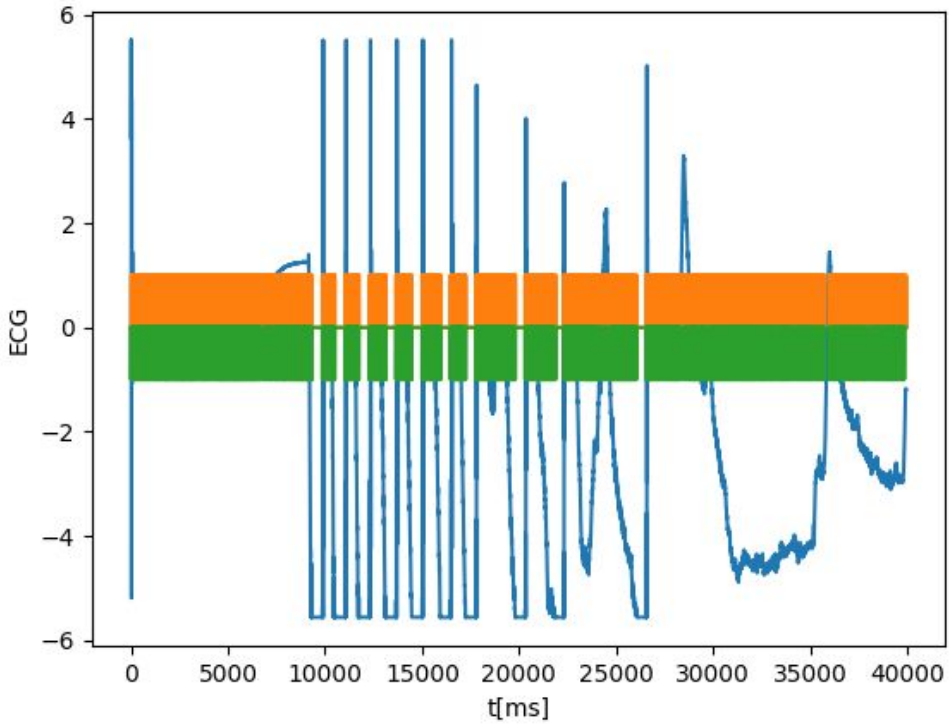
```
import csv

import matplotlib.pyplot as plt
data = []
time = []

rising = True
prev_ecg = -10000 # der nächste echte Messwert ist groesser
time_in_ms = 0
# die csv-Datei "100_120.dat" zeilenweise durchgehen
for line in csv.reader( open("100_120.dat") ):
    ecg = float(line[0]) # die erste Spalte ist der EKG-Wert
    row = [ecg]  # ecg, max, min
    if rising:
        if ecg < prev_ecg:  # Wendepunkt oben
            print(time_in_ms, ecg)
            rising = False
            row.extend([1,0]) # Maxima, kein Minima
        else:
            row.extend([0,0]) # kein Maxima, kein Minima
    else:
        if ecg > prev_ecg:  # Wendepunkt unten
            rising = True
            row.extend([0,-1]) # kein Maxima, Minima
```

```
            else:
                row.extend([0,0]) # kein Maxima, kein Minima
        prev_ecg = ecg
        data.append(row)
        time.append(time_in_ms)
        time_in_ms += 2   # 500Hz, 2ms pro Schritt

plt.plot(time, data)
plt.ylabel("ECG")
plt.xlabel("t[ms]")
plt.show()
```

“`plt.plot(time, data)`” : Wir schauen uns die Kurve und die Zeitpunkte der Maxima und Minima an.

“`row = [ecg]  # ecg, max, min`” schreibt den Messwert in das Zeilenarray.

“`row.extend([max,min])`” erweitert um die max-, min-Marker. max ist 1 wenn ein Maxima gefunden wurde, “min” ist -1 wenn ein Minima gefunden



Damit finden wir hauptsächlich die kleinen Störungen.

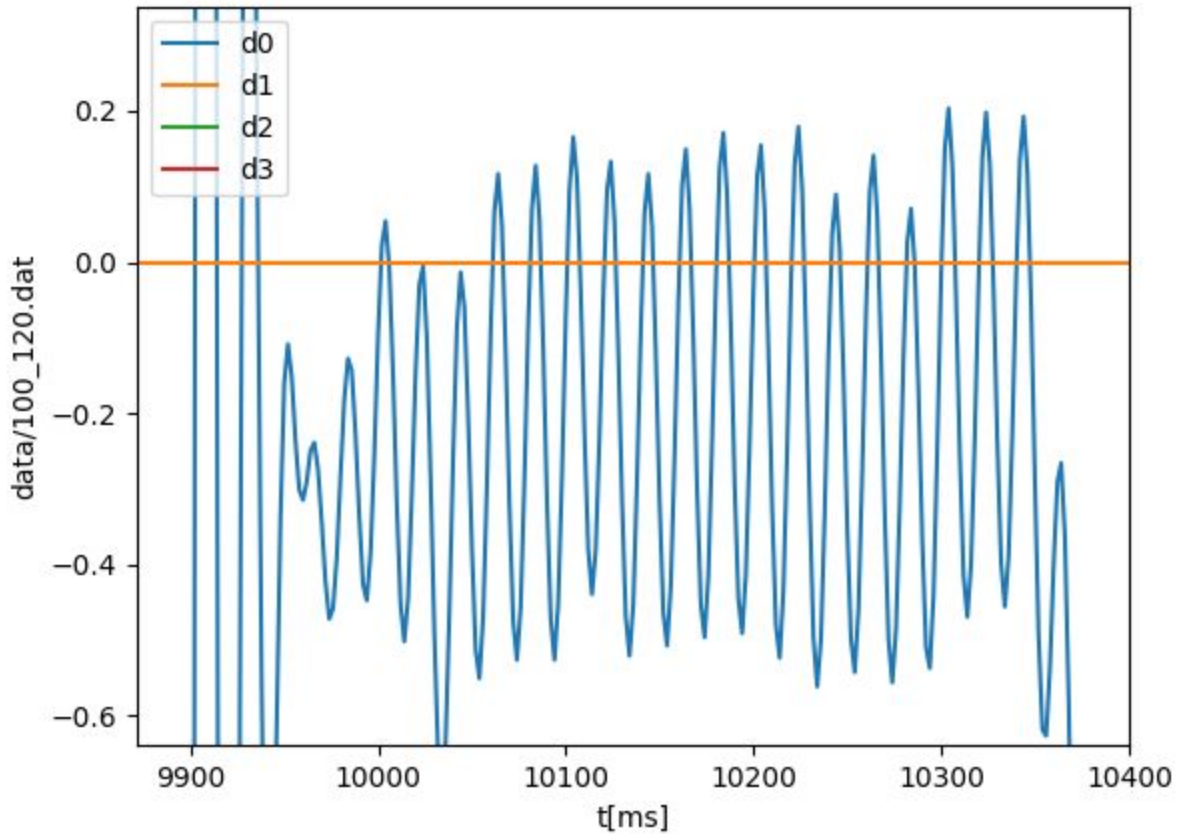
4 arduino data two

Ansatz im Detail

“`rising = True`” wir sind entweder in einer monoton steigenden oder fallenden Flanke.

“
```
if rising:
    if ecg < prev_ecg:  # Wendepunkt oben
        rising = False
```
” rising bedeutet die Werte sind monoton angestiegen, wenn der aktuelle Wert kleiner ist als der vorige hat sich das Vorzeichen der Steigung geändert.
Wir schalten auf falling.

Problem: digitalisierte Werte haben eine Auflösung. Zum Beispiel 10-bit Analog Digital Wandler können von 0 bis 1023 darstellen. Änderungen unter 0.2 % sind zufällig, wenn nicht das Rauschen davor alles zufällig macht.



Wir haben ein kleines, höherfrequentes Signal auf unserem EKG-Signal.

Änderungen unter … 0.8 V sind noch kein EKG-Signal, interessieren uns nicht.

Entwurf

```
# if rising
#   if ecg > max_ecg
#     max_ecg = ecg
#     max_t = time
#   else if ecg < (max_ecg - threshold):
#     print( max_t, max_ecg )
#     rising = false
#     min_ecg = ecg
#     min_t = time
# else
#   if ecg < min_ecg
#     min_ecg = ecg
#     min_t = time
#   else if ecg > (min_ecg + threshold):
#     print( min_t, min_ecg )
#     rising = true
#     max_ecg = ecg
#     max_t = time
```

Umbau

Erst wenn der Wert um einiges kleiner ist als das bisherige Maximum schalten wir um auf fallend. Dasselbe für Minimasuchen.

```
    rising = True
```

Wir müssen uns das Maximum/Minimum merken und nicht nur mit dem vorigen Wert vergleichen.

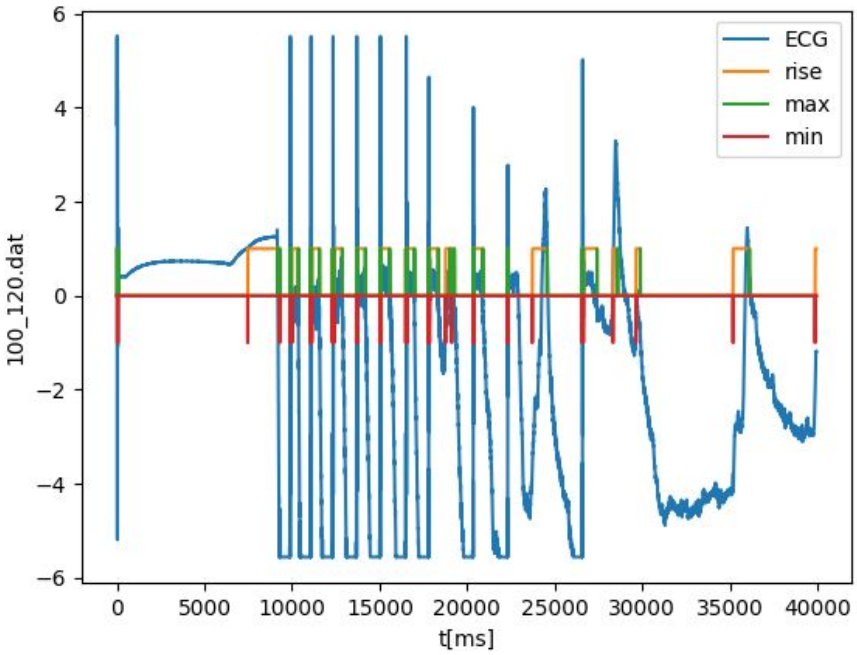
```
    prev_ecg = -10000 # der nächste echte Messwert ist groesser
    max_ecg = 0
    min_ecg = 0

    time_in_ms = 0
```

der erforderliche Unterschied zwischen Maxima/Minima und dem aktuellen, damit es als

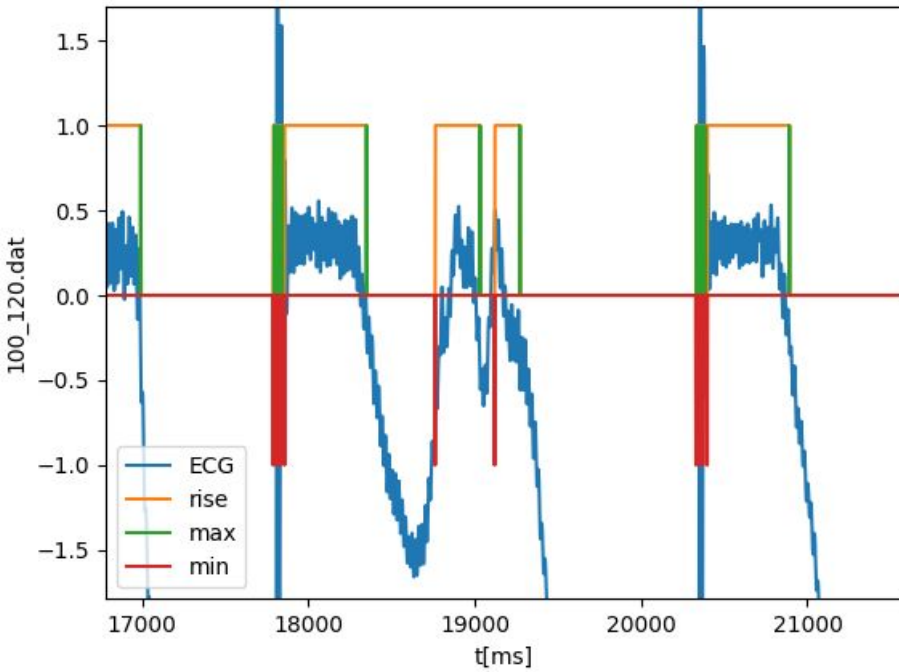
Umkehrpunkt gilt.

```
requ_diff = 1

for line in csv.reader( open(fname) ):
    ecg = float(line[0]) # die erste Spalte ist der EKG-Wert
    row = [ecg, rising]  # ecg, rising, max, min
    if rising:
        if ecg > max_ecg:
            max_ecg = ecg # ansteigend
            row.extend([0,0]) # kein Maxima, kein Minima
        if ecg < prev_ecg:  # Wendepunkt oben
        elif ecg < (max_ecg - requ_diff):  # Wendepunkt oben
            print(time_in_ms, ecg)
            rising = False
            row.extend([1,0]) # Maxima, kein Minima
            min_ecg = ecg     # Startwert für Minima
        else: # zuwenig Aenderung
            row.extend([0,0]) # kein Maxima, kein Minima
    else: # falling
        if ecg < min_ecg:
            min_ecg = ecg # fallend
            row.extend([0,0]) # kein Maxima, kein Minima
        if ecg > prev_ecg:  # Wendepunkt unten
        elif ecg > (min_ecg + requ_diff):  # Wendepunkt unten
            rising = True
            row.extend([0,-1]) # kein Maxima, Minima
            max_ecg = ecg # Startwert Maximasuche
        else: # zu wenig Aenderung
            row.extend([0,0]) # kein Maxima, kein Minima
    prev_ecg = ecg
```

Besser



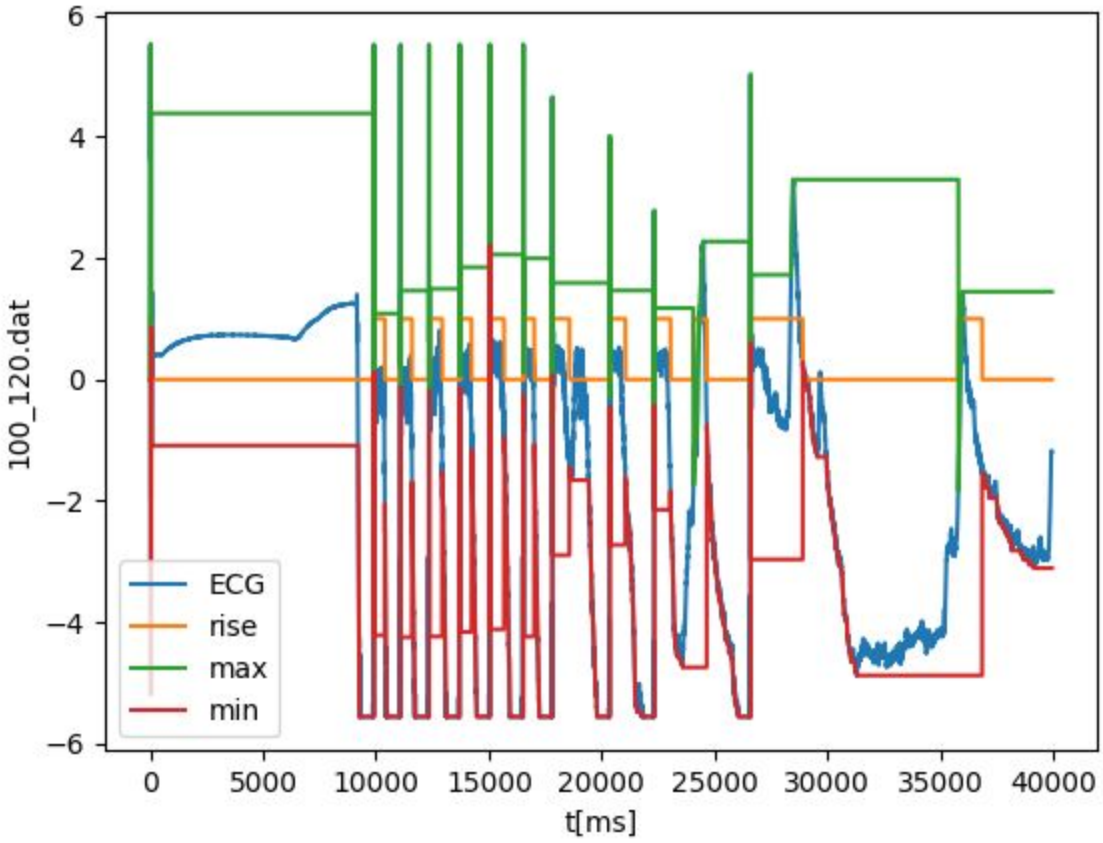
Es werden immer noch viele Umkehrpunkte gefunden wo wir keine sehen



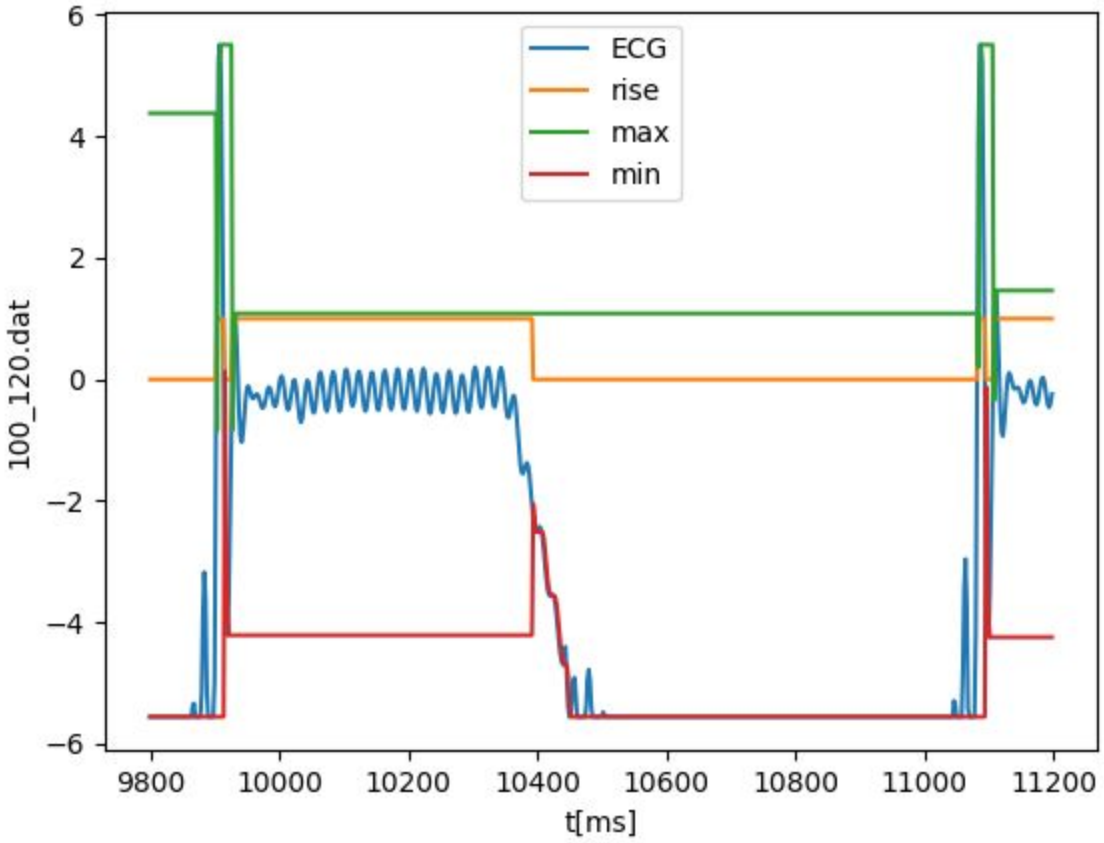
Der Schlenker zwischen 18000 und 19000 ist zwar ein Messfehler, aber wir probieren mit

“**requ_diff = 3**” und geben die Werte “**max_ecg**” und “**min_ecg**” aus anstatt der Wendepunkte

```
rising = True

max_ecg = 0
min_ecg = 0
time_in_ms = 0

requ_diff = 3

for line in csv.reader( open(fname) ):
    ecg = float(line[0]) # die erste Spalte ist der EKG-Wert
    row = [ecg, rising, max_ecg, min_ecg]  # ecg, max, min
    if rising:
        if ecg > max_ecg:
            max_ecg = ecg
        elif ecg < (max_ecg - requ_diff):  # Wendepunkt oben
            print(time_in_ms, ecg)
            rising = False
            min_ecg = ecg
    else:
        if ecg < min_ecg:
            min_ecg = ecg
        elif ecg > (min_ecg + requ_diff):  # Wendepunkt unten
            rising = True
            max_ecg = ecg
    data.append(row)
    time.append(time_in_ms)
    time_in_ms += 2   # 500Hz, 2ms pro Schritt
plt.plot(time, data)
plt.ylabel(pathlib.Path(fname).parts[-1])
plt.legend(["ECG", "rise", "max", "min"])
plt.xlabel("t[ms]")
plt.show()
```

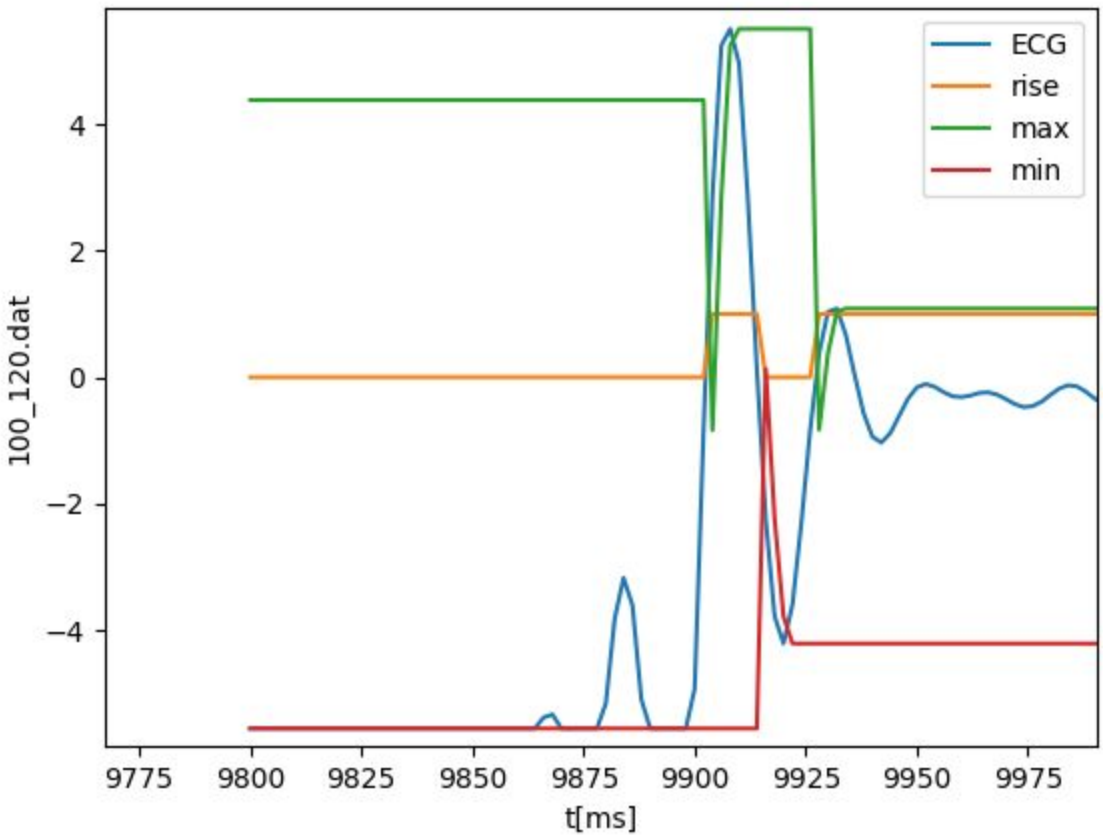

100_120.dat
t[ms]
ECG
rise
max
min
6
4
2
0
−2
−4
−6
0
5000
10000
15000
20000
25000
30000
35000
40000

Im Detail



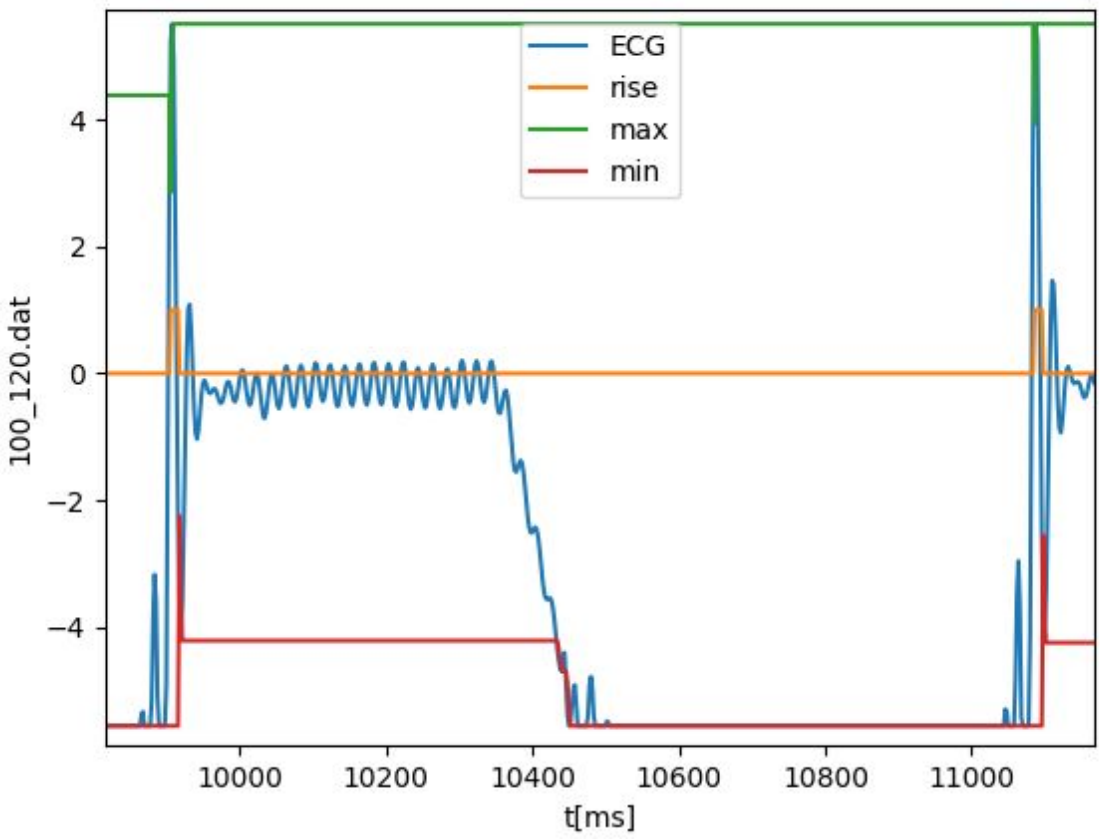
Das funktioniert noch nicht ganz, es werden zwei Maxima/Minima gefunden.

- Bei 9900 ein hoher oberer Umkehrpunkt über 5,
- kurz darauf ein Minimum etwas unter 4,
- danach wird bis 10400 ein Maximum gesucht (grün ist cirka 1)
- nach 10400 fällt das ECG unter max-threshold (-2) und es wird ein Minima gesucht.



- Dieser Teil beginnt Minimasuche (rise=0). ECG am Minimum: rot und blau. Der Anstiege des ECG nach 9875 ist noch zu klein um ein Umschalten auf Maximasuche auszulösen.
- Bei 9900 überschreitet das ECG den Wert min_ecg+threshold (3), es wird auf Maximasuche umgeschalten (rise=1, max wird auf den ECG-Wert gesetzt und folgt ihm. Bei 9910 etwas über 5.
- Kurz danach ist ECG kleiner max-threshold und es wird auf Minimasuche umgeschalten.
- Das ECG-Signal fällt bis -4
- steigt aber noch einmal über -1 (-4 + threshold 3), dadurch wird ein weiteres Maximum gefunden.

Wenn wir den threshold auf 6 setzen müsste das System ab 9910 (max - threshold) im Zustand Minimasuche bleiben.



... funktioniert ... **AAABER** der Wert "6" ist nur für diese Kurve gut, sobald die Amplituden kleiner werden (größer wahrscheinlich nicht weil unten clippt das SIgnal bereits) finden wir vielleicht gar nichts mehr.

Berechnen der Pulsfrequenz.

Wir müssen uns den Zeitpunkt des Maximas merken:

```
max_t = 0
```

und in setzen.

```
if ecg > max_ecg:
    max_ecg = ecg
    max_t = time_in_ms
```

wenn der ECG-Wert danach unter den Wert max_ecg - 6 gefallen ist, berechnen wir die Pulsfrequenz “`60000. / (max_t - max_t_prev)`”. Dazu brauchen wir auch den Zeitpunkt des vorigen Maximum “`max_t_prev`”.

```
if rising:
    if ecg > max_ecg:
        max_ecg = ecg
        max_t = time_in_ms
    elif ecg < (max_ecg - requ_diff):  # Wendepunkt oben
        rising = False
        min_ecg = ecg
        pulse_max = 60000. / (max_t - max_t_prev)
        max_t_prev = max_t
```

und nach der Berechung

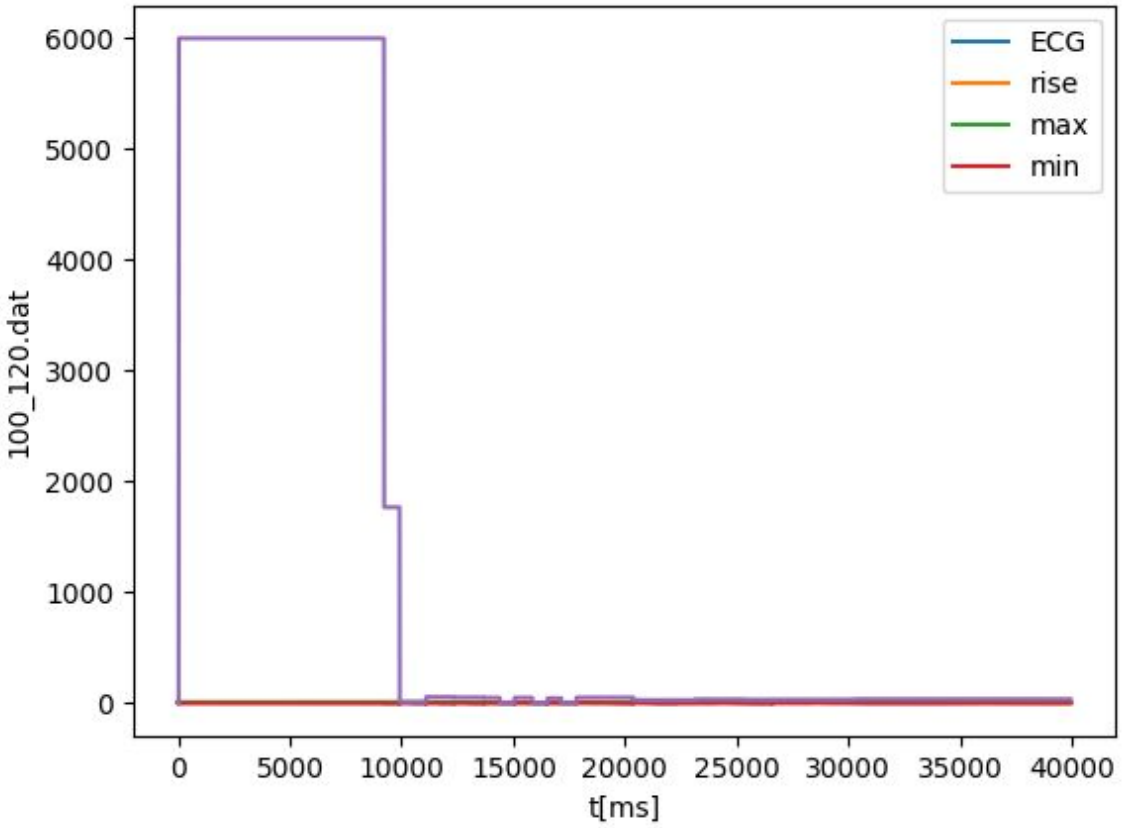
```
row.append( pulse_max )
```

Beim Aufruf kommt die Fehlermeldung

```
     pulse_max = 60000. / (max_t - max_t_prev)

ZeroDivisionError: float division by zero
```

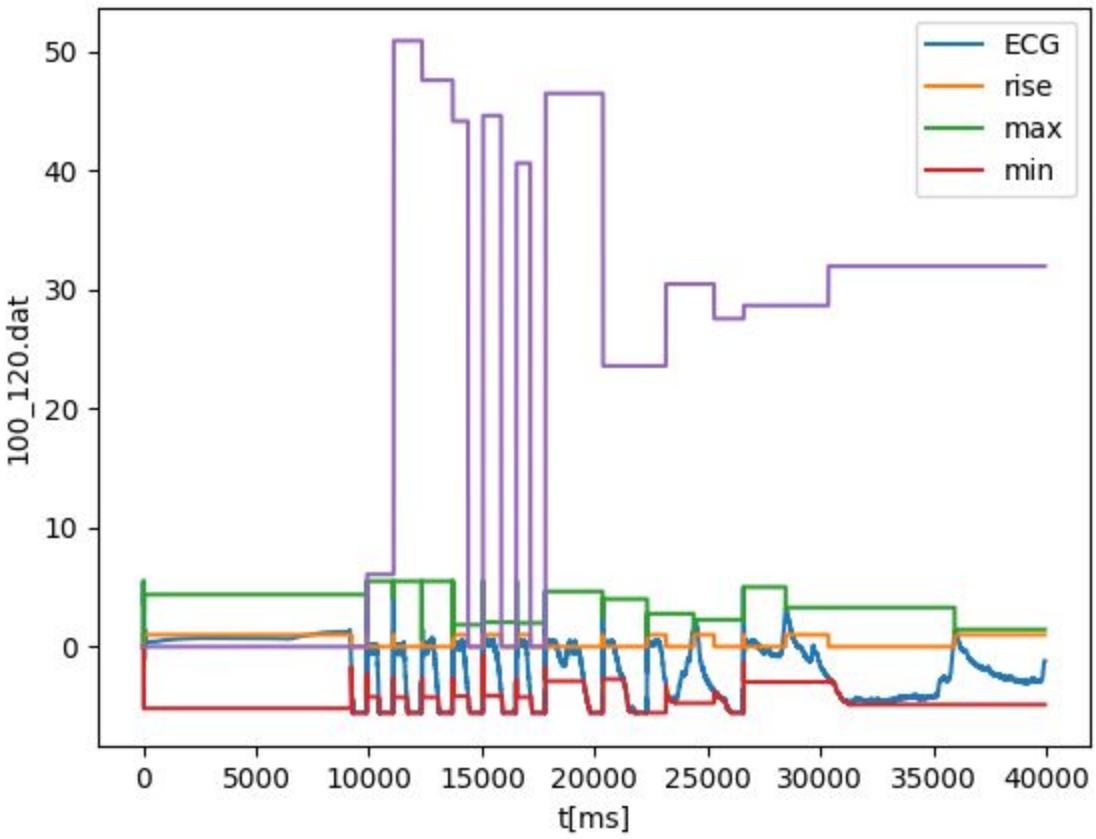
Extrahieren der Pulsberechnung in eine Funktion:

```
def bpm_from_2_ms(t, prev_t):
    dt = t - prev_t
    if dt <= 0:
        return 0
    return 60000. / dt
```

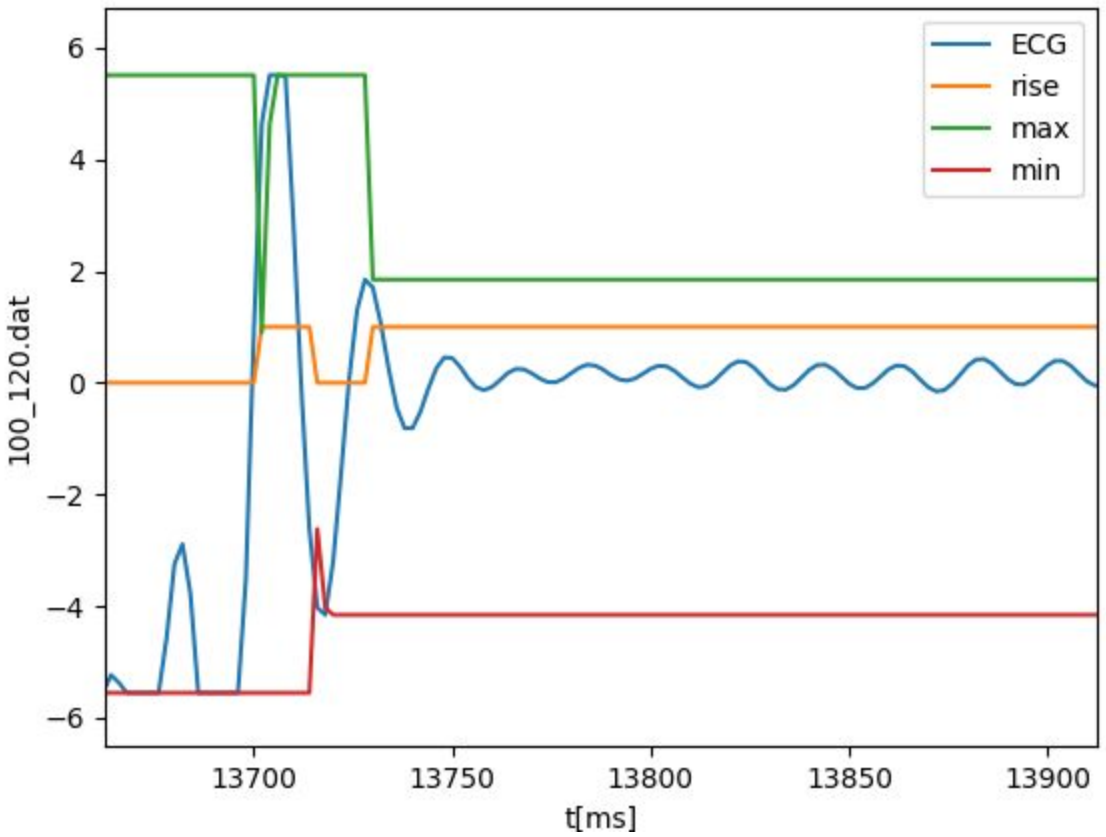


Wir klammern unmögliche Pulsfrequenzen aus.

```
def bpm_from_2_ms(t, prev_t):
    dt = t - prev_t
    if dt < 200: # pulse > 300 ?
        return 0
    return 60000. / dt
```

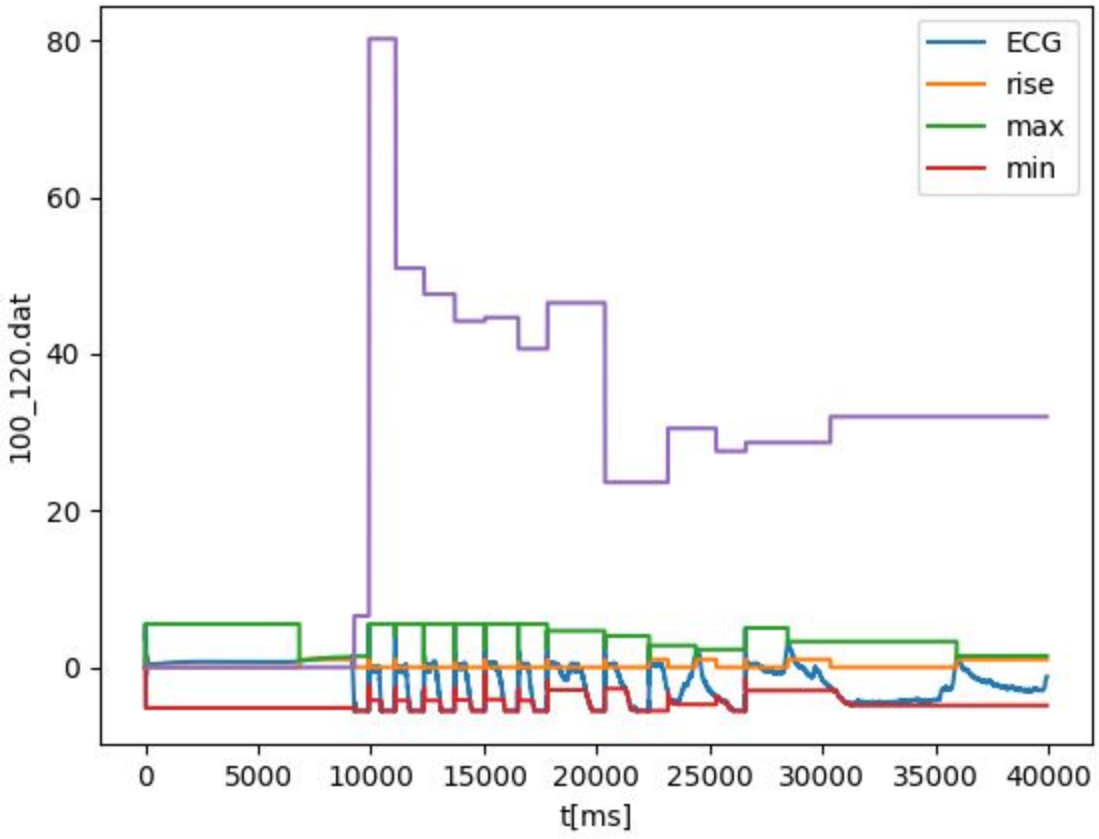


Die Bereiche um 15000 mit Puls = 0 sind die bei denen die negative Spitze nach dem Maximum beinahe -5 war.



Das heisst wir müssen verhindern, dass ein Maximum zu schnell hintereinander auftritt.

```
if ecg < min_ecg:
    min_ecg = ecg
elif ecg > (min_ecg + requ_diff) and (
          time_in_ms - max_t > 100):  # Wendepunkt unten
    rising = True
    max_ecg = ecg
```



REFERENCES

- Matplotlib: https://matplotlib.org/tutorials/introductory/pyplot.html
- python csv: https://docs.python.org/3/library/csv.html
- "Time and tide waits for no man" (Quote from Shakespeare, Julius Caesar)
- TELE ECG Database: 250 telehealth ECG records

REVISION HISTORY

- 0.3 16.12.2020 - noch unter 20 Seiten
- 0.2 15.12.2020 - anderer Versuch
- 0.1 10.12.2020 - NOGO