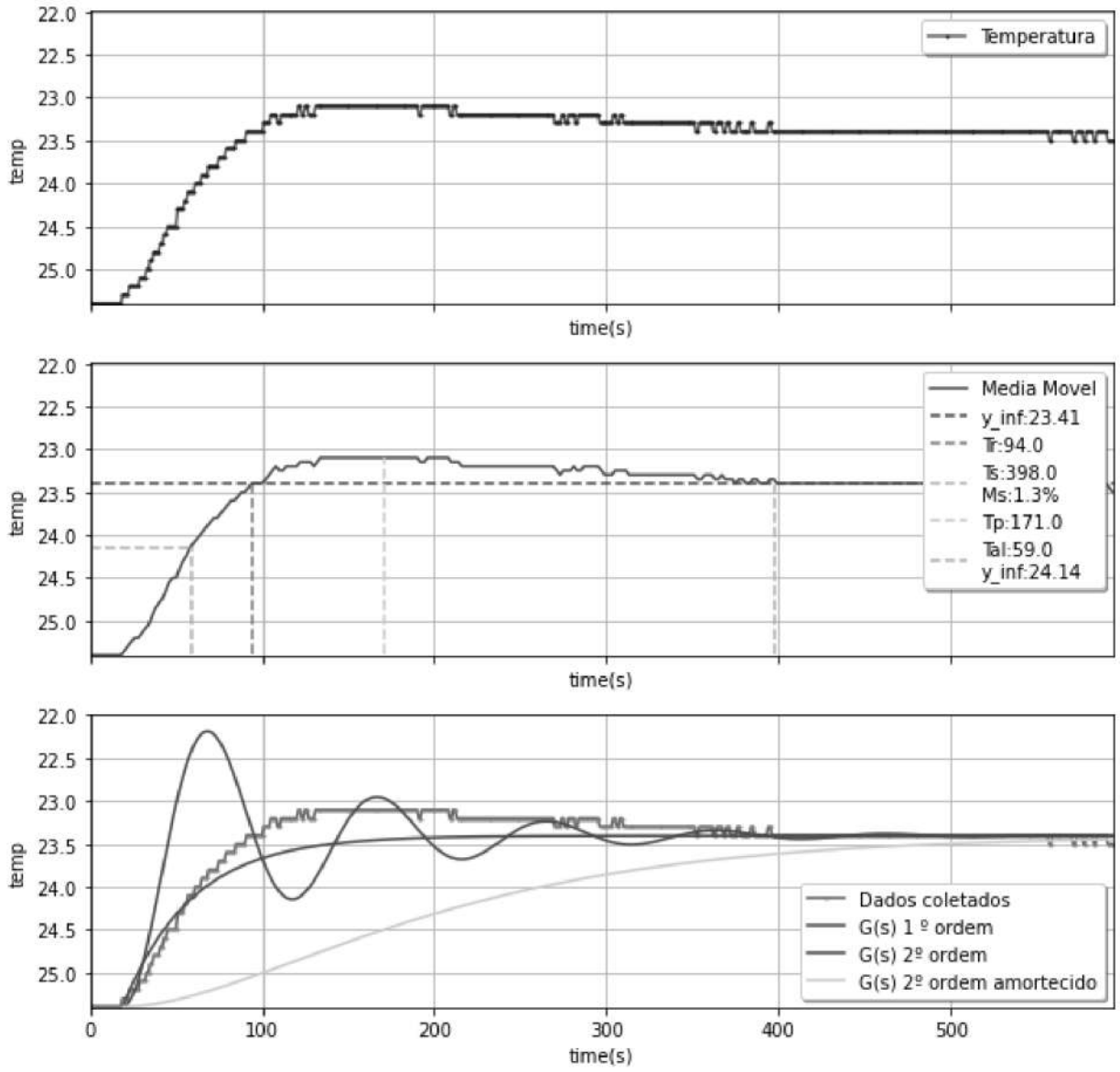
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
import control as control



from asyncio.windows_events import NULL


path = '../dgrau'
DEGRAU = 70 # 70 POSCENTO DE PWM
s = control.tf('s')
for diretorio, subpastas, arquivos in os.walk(path):
    for arquivo in arquivos:
        G1 = NULL
        G2 = NULL
        csi_cal = NULL
        G2CritiAmo = NULL
        fig, (ax1, ax2,ax3) = plt.subplots(3, 1, sharey=True, sharex= True)
        bd = pd.read_csv(os.path.join(diretorio, arquivo), delimiter=';')
        temp  = np.array(bd['temp'].values)
        time = np.array(bd['time(s)'].values)
        ax1.plot(time, temp,marker="o", markersize=1, markeredgecolor="blue", label='Temperatura')
        ax3.plot(time, temp,marker="o", markersize=1, markeredgecolor="grey", label='Dados coletados')
        # print(np.where(tmm < max(tmm)))
        #
        ## Media Movel
        temp_init = max(temp)
        tmm = bd['temp'].rolling(4).mean().replace(np.nan, temp_init) # temperatura media movel
        time_delay =  time[np.where(tmm < temp_init)[0][0]]
        ax2.plot(time, tmm, color='r', label='Media Movel')
        #
        ## Estabilização y_inf
        init_est = np.where(time >= 400)[0][0]
        y_inf = np.mean(tmm[init_est:])
```

```
    ax2.plot(time, y_inf*np.ones(len(temp)),'--', color='g',label='y_inf:{}'.format(round(y_inf,2)))
    #
    ## Tempo de subida
    util_tr = np.where(tmm <= y_inf )[0]
    indice_tr = util_tr[0]
    tr = time[indice_tr]
    ax2.plot(tr*np.ones(indice_tr), tmm[:indice_tr],'--', color='grey',label='Tr:{}'.format(round(tr,2)))
    #
    ## Maximo sobresinal e tempo de pico
    temp_min = min(tmm[:400])
    max_sobr =abs(((temp_min  - y_inf) / y_inf) *100.0)

    if(max_sobr>0.1):
        #
        ## Tempo de estabilixação
        util_ts = np.where(tmm[:init_est] < (y_inf*0.95 + temp_min*0.05))[0]
        indice_ts = util_ts[len(util_ts)-1]
        ts = time[indice_ts]
        ax2.plot(ts*np.ones(indice_tr), tmm[:indice_tr],'--', color='aqua',label='Ts:{}\nMs:{}%'.format(round(ts,2), rou
        ## Tempo de pico
        util_tp = np.where(tmm[:indice_ts-1] == temp_min )[0]
        indice_tp = round((util_tp[0]+util_tp[len(util_tp)-1])*0.5)
        tp = time[indice_tp]
        #ax2.text(tp,temp_min -0.35, 'Tp:{}'.format(round(tp,2)))
        ax2.plot(tp*np.ones(indice_tp), tmm[:indice_tp],'--', color='gold',label='Tp:{}'.format(round(tp,2)))
        #
        ##calculo da planta de segunda prdem
        csi_cal = math.cos(math.pi*(1- tr/tp))
        Wn = 4/( ts * csi_cal)
        G2 = ((Wn**2)*((y_inf - temp_init)/DEGRAU))/ ((s**2) + (2*csi_cal*Wn*s) + (Wn**2))
    Wn = 4/( ts )
    G2CritiAmo = ((Wn**2)*((y_inf - temp_init)/DEGRAU))/ ((s**2) + (2*Wn*s) + (Wn**2))

    y_inf63p = y_inf + ((temp_init - y_inf )*0.37)
    indice_tal = np.where(tmm <= (y_inf63p))[0][0]
    tal =  time[indice_tal]
    ax2.plot(tal*np.ones(indice_tal), tmm[:indice_tal],'--', color='aqua',label='Tal:{}\ny_inf:{}'.format(round(tal,2),r
    ax2.plot(time[:indice_tal], y_inf63p*np.ones(indice_tal),'--', color='aqua')
```

```
## calculo da planta para primeira ordem
a = 1/(tal- time_delay)
k = (( y_inf - temp_init  )*a)/DEGRAU
G1 =  (k/(s + a))

##propriedades do grafico em s
maxG = 0
minG = 10000
for (G, labelG, cor) in [(G1, 'G(s) 1 º ordem', 'r'),(G2,'G(s) 2º ordem', 'g'), (G2CritiAmo, 'G(s) 2º ordem amortecid
    if (G!=NULL):
        dt = 0.01 # passo da simulação
        nt = int ( max(time) / dt ) + 1 # Number of points of sim time
        t = np.linspace(0,max(time),nt)
        u = DEGRAU * np.ones(nt)
        T,Y= control.forced_response(G,t,u , X0 =0.0)
        Y = Y + temp_init
        T = T + time_delay
        ax3.plot(T, Y, color=cor, label=labelG)
        maxY = max(Y)
        minY = min(Y)
        maxG = maxG if maxG > maxY else maxY
        minG = minG if minG < minY else minY


##propriedades do grafico
for ax in [ax1,ax2,ax3]:
    ax.set_xlabel('time(s)')
    ax.set_ylabel('temp')
    ax.grid(True)
    ax.legend(shadow=True, fancybox=True)
fig.suptitle(arquivo)
plt.xlim([0, time[len(time)-1]])
plt.ylim([maxG + 0.01, minG - 0.2])
plt.rcParams['figure.figsize'] = [10, 10]
plt.show()
for (G, labelG) in [(G1, 'G(s) 1 º ordem'),(G2,'G(s) 2º ordem'), (G2CritiAmo, 'G(s) 2º ordem amortecido')]:
        if (G!=NULL):
            print(labelG,G)
```

d70_976Hz.csv




```
G(s) 1 º ordem
-0.0007124
----------
s + 0.025
```

```
G(s) 2º ordem
        -0.000119
-------------------------
s^2 + 0.0201 s + 0.004176

G(s) 2º ordem amortecido
       -2.878e-06
-------------------------
s^2 + 0.0201 s + 0.000101
```

d70_980Hz.csv