Documentação funcionamento do software leitura de manômetro

Status	In Progress
Assign	
Due	
Tags	Mackenzie

Documentação código leitura do manômetro de pressão

- A documentação a seguir tem a intenção de explicar o script escrito em python para leitura de manômetros através do processamento de imagem.
- As configurações do manômetro são feitas através das seguintes variáveis que estão das linhas 68 as linhas 72

```
  min_angle = 45
  max_angle = 315
  min_value = 0
  max_value = 100
  units = "BAR"
```

- Com uma simples implementação de um frontend é possível evitar de que os valores fiquem chumbados no código de maneira(hardcoded)
- Com o cálculo do resultado variável final nesse caso chamada de val, são feitas as condições desejadas para que haja as tomadas de decisão

```
img = cv2.imread(folder_path + "/" + filename + ".jpg")
  val = get_current_value(img, min_angle, max_angle, min_value, max_value, x, y, r, filename,folder_path)
  if(val < 5 ):
    print("Risco!")
  if(5 < val < 11):
    print("Subpressão")
  if(12 < val < 17 | 24 < val < 29):
    print("Alerta")
  if(18 < val < 23):
    print("Condicao de trabalho")
  if( val > 30):
    print("Superpressão")
  print ("Current reading: %s %s" %(("%.2f" % val), units))
```

- A pasta contendo a imagem deve estar dentro de uma pasta do projeto, nesse caso dentro da pasta imagens, caso haja necessidade de mudança a pasta deve ser renomeada e os caminhos relativos (relatives path) devem ser respeitados.
- No envio da imagem basta coloca-la na pasta desejada fazer a importação de maneira correta no código e tudo funcionará

```
folder_path = 'images'
  filename = input("Filename without extension .jpg: ")
```

- O formato e extensão da imagem importam portanto o formato correto das imagens deve ser colocado em .jpg, caso isso não seja respeitado coloca-se uma condicional que converte a foto no formato desejado como mostrado a seguir

```
from PIL import Image
import os

def convert_to_jpeg(input_path, output_path):
    try:
        with Image.open(input_path) as im:
            rgb_im = im.convert('RGB')
            rgb_im.save(output_path, 'JPEG', quality=95)
        return True
    except Exception as e:
        print(f"Error: {e}")
        return False
input_path = '/path/to/input/image.png'
output_path = '/path/to/output/image.jpg'
convert_to_jpeg(input_path, output_path)
```

- também para melhor performace uma foto com resolução quadrada do exemplo 960x960 ajuda na calibração e trás calibragens mais precisas. Para isso basta um simples script de crop da imagem enviada como demonstrado a seguir

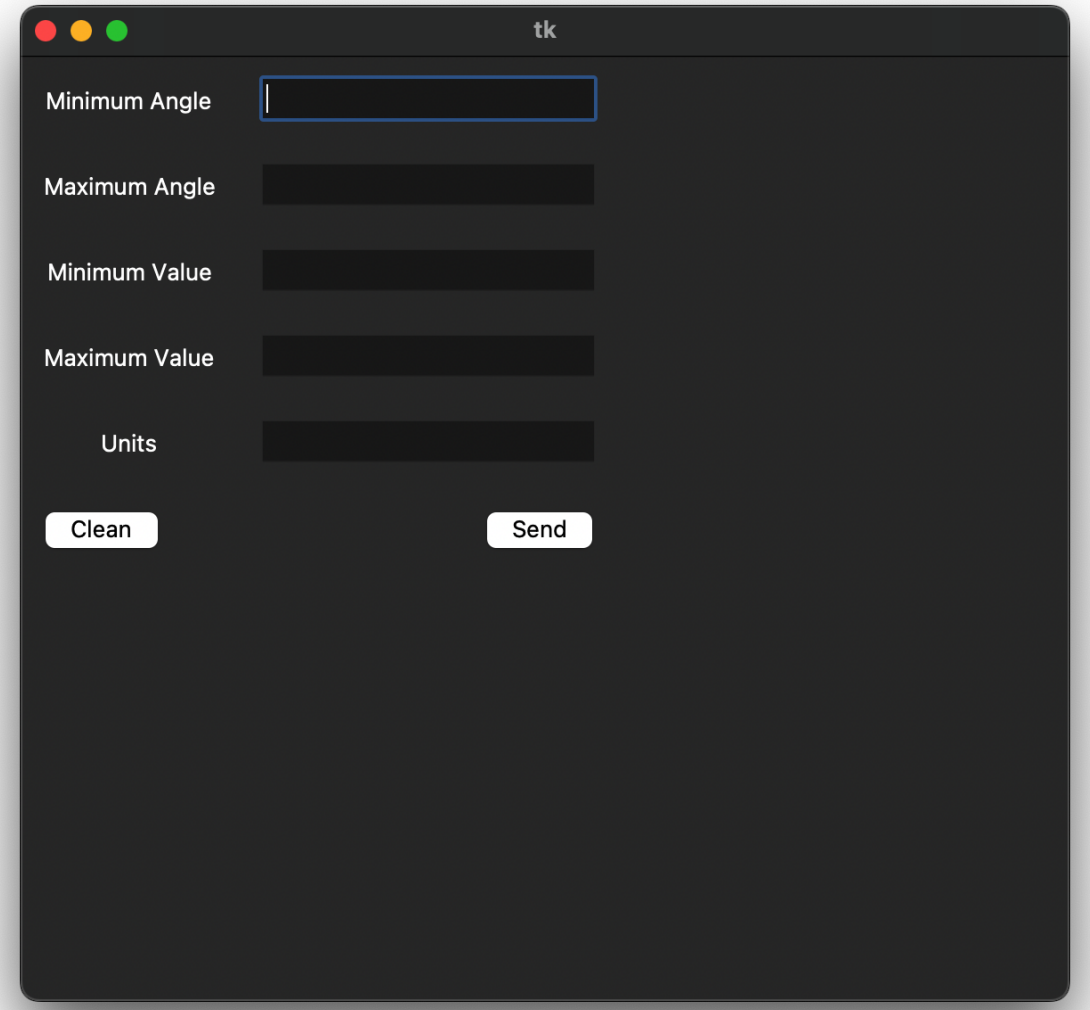
```
from PIL import Image

def crop_and_save_to_square(input_path, output_path):
    # Open the input image
    with Image.open(input_path) as image:
        # Get the size of the input image
        width, height = image.size

        # Find the minimum dimension
        min_dim = min(width, height)

        # Calculate the coordinates for the crop
        left = (width - min_dim) // 2
        top = (height - min_dim) // 2
        right = (width + min_dim) // 2
        bottom = (height + min_dim) // 2
```

```
        # Crop the image to the square size
        square_image = image.crop((left, top, right, bottom))

        # Resize the image if it's larger than 1200x1200
        max_size = (1200, 1200)
        square_image.thumbnail(max_size, Image.ANTIALIAS)

        # Save the cropped and resized image as a new file
        square_image.save(output_path)

# Example usage:
crop_and_save_to_square("path/to/input/image.jpg", "path/to/output/image.jpg")
```

- para interface front-end depende do que se achar mais interessante, uma simples busca na internet encontra resultados interessantes e simples que resolvem o problema como a seguir.

```
import tkinter as tk
import analog_gauge_reader


def submit():
    # retrieve values from entry widgets
    val1 = min_angle.get()
    val2 = max_angle.get()
    val3 = min_value.get()
    val4 = max_value.get()
    val5 = units.get()

    # print values to console
    print(val1, val2, val3, val4, val5)

def clear():
    # clear values in entry widgets
    min_angle.delete(0, tk.END)
    max_angle.delete(0, tk.END)
    min_value.delete(0, tk.END)
    max_value.delete(0, tk.END)
    units.delete(0, tk.END)

# create a tkinter window with a larger size
window = tk.Tk()
window.geometry("400x400")

# create label widgets for each input field
label1 = tk.Label(window, text="Minimum Angle")
label2 = tk.Label(window, text="Maximum Angle")
label3 = tk.Label(window, text="Minimum Value")
label4 = tk.Label(window, text="Maximum Value")
label5 = tk.Label(window, text="Units")

# create five entry widgets
min_angle = tk.Entry(window)
max_angle = tk.Entry(window)
min_value = tk.Entry(window)
max_value = tk.Entry(window)
units = tk.Entry(window)
```

```
# create two buttons
send_button = tk.Button(window, text="Send", command=submit)
clean_button = tk.Button(window, text="Clean", command=clear)

# arrange widgets in the window using grid layout
label1.grid(row=0, column=0, padx=10, pady=10)
min_angle.grid(row=0, column=1, padx=10, pady=10)
label2.grid(row=1, column=0, padx=10, pady=10)
max_angle.grid(row=1, column=1, padx=10, pady=10)
label3.grid(row=2, column=0, padx=10, pady=10)
min_value.grid(row=2, column=1, padx=10, pady=10)
label4.grid(row=3, column=0, padx=10, pady=10)
max_value.grid(row=3, column=1, padx=10, pady=10)
label5.grid(row=4, column=0, padx=10, pady=10)
units.grid(row=4, column=1, padx=10, pady=10)
send_button.grid(row=5, column=1, padx=10, pady=10, sticky="E")
clean_button.grid(row=5, column=0, padx=10, pady=10, sticky="W")

# start the event loop
window.mainloop()
```



Para rodar o código

- Para usar o projeto basta rodar o arquivo analog_gauge_reader no terminal em python e seguir o fluxo natural.
- Voce precisará inserir o nome da foto que deseja ler, sem a extensão.
- Basta fazer isso caso a imagem esteja na pasta images, caso não esteja é preciso arrumar os caminhos relativos.
- Com o envio da imagem pro programa, haverá um print da resposta final no console e será gerado 2 novas fotos, referentes as calibragens que foram feitas utilizando a imagem.