Fundação Escola Técnica Liberato Salzano Vieira da Cunha
Curso Técnico de Eletrônica

MANUAL DE SERVIÇO DO SIMULADOR DE CÉLULA FOTOVOLTAICA

Disciplina de Sistemas de Comunicações II & Eletrônica de Potência

Bernardo Rodrigues da Silva
Mariana Stefani Irigaray

Turma: 4411

Novo Hamburgo
14 de dezembro de 2023

1. DISCLAIMER

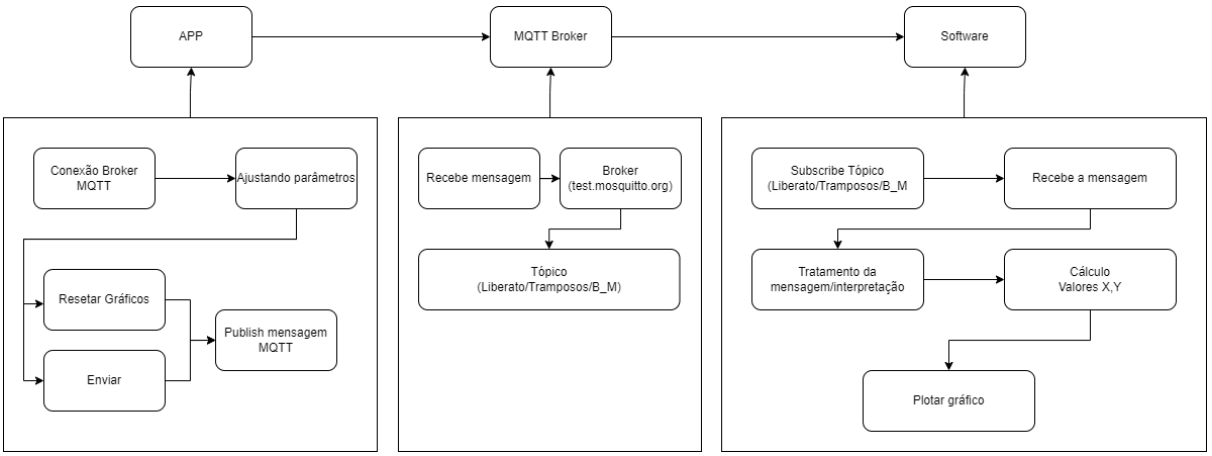
TODA ALTERAÇÃO FEITA NO CÓDIGO E NO APLICATIVO QUE NÃO SEJA REALIZADA PELA INTERFACE DE USUÁRIO, NÃO É RECOMENDADA E DEVE SER FEITA APENAS POR UM PROFISSIONAL QUALIFICADO, DEVE SER FEITA POR SUA CONTA E RISCO.

2. INTRODUÇÃO

O simulador desenvolvido funciona de maneira extremamente simples, ele vem com um projeto desenvolvido em Python (é necessário ter um interpretador de Python instalado no dispositivo que for executar o software), e um aplicativo de Celular no formato .apk, apenas executar o arquivo para instalar e aplicativo.

O aplicativo conta com as seguintes funcionalidades para o usuário enviar dados de radiação solar (W/m²) e de temperatura (ºC), o aplicativo então transmite através de protocolo MQTT para o dispositivo em que está rodando o software. O software pode calcular as curvas de potência e IV em função tanto da temperatura, quanto da radiação, quanto de ambos, cabendo ao usuário decidir qual será a opção mais adequada.

3. DIAGRAMA DE BLOCOS DO SISTEMA



4. ESPECIFICAÇÕES

4.1 Software

O software já vem com diversos parâmetros para a simulação já preenchidos deixando para o usuário preencher apenas as medidas de temperatura e de radiação solar, entretanto, no ínicio do arquivo fotovoltaica.py diversos parâmetros são inicializados ali, caso o usuário

necessite ele pode alterar os parâmetros para uma célula voltaica diferente do padrão que utilizamos.

```
# Parâmetros do painel
Voc_ref = 22.65
Isc_ref = 3.8
Vm_ref = 18.53
Im_ref = 3.592
Pm = 66.6
FF = 0.773
Tref = 25
N = 36
```

Nessa etapa é possível alterar os parâmetros e as especificações de fábrica do painel solar. Os valores utilizados são de um painel desconhecido, foram valores arbitrários.

No arquivo graphics.py, também é possível alterar o tempo de permanência do gráfico na tela, inclusive no graphics.py por se tratar de funções mais simples, ainda é possível fazer alguns ajustes caso necessário.

É possível alterar o tempo de permanência do gráfico na tela através da seguinte variável:

```
TEMPO_DE_PAUSE = 10
```

Tempo dado em segundos.

Além disso, é possível fazer alterações na maneira que o gráfico é plotado, como cores, eixos, nomenclaturas e eventualmente se necessário escalas.

Para alterar as cores:

```
def random_color():
    # Gerar três valores aleatórios entre 0 e 255
    R = random.randint( a: 0,  b: 255)
    G = random.randint( a: 0,  b: 255)
    B = random.randint( a: 0,  b: 255)

    # Retornar uma lista com os valores
    return [R, G, B]
```

Para alterar os padrões do gráfico:

```
    plt.plot( *args: eixo_x, eixo_y, label=label,
             color=(cor_da_curva[0] / 255, cor_da_curva[1] / 255, cor_da_curva[2] / 255))

plt.xlabel('Tensão (V)')
plt.ylabel('Potência (W)')
plt.title('Curvas Características do Painel Solar')
plt.legend()

plt.grid(True)
plt.show()
plt.pause(TEMPO_DE_PAUSE)
plt.close()
```

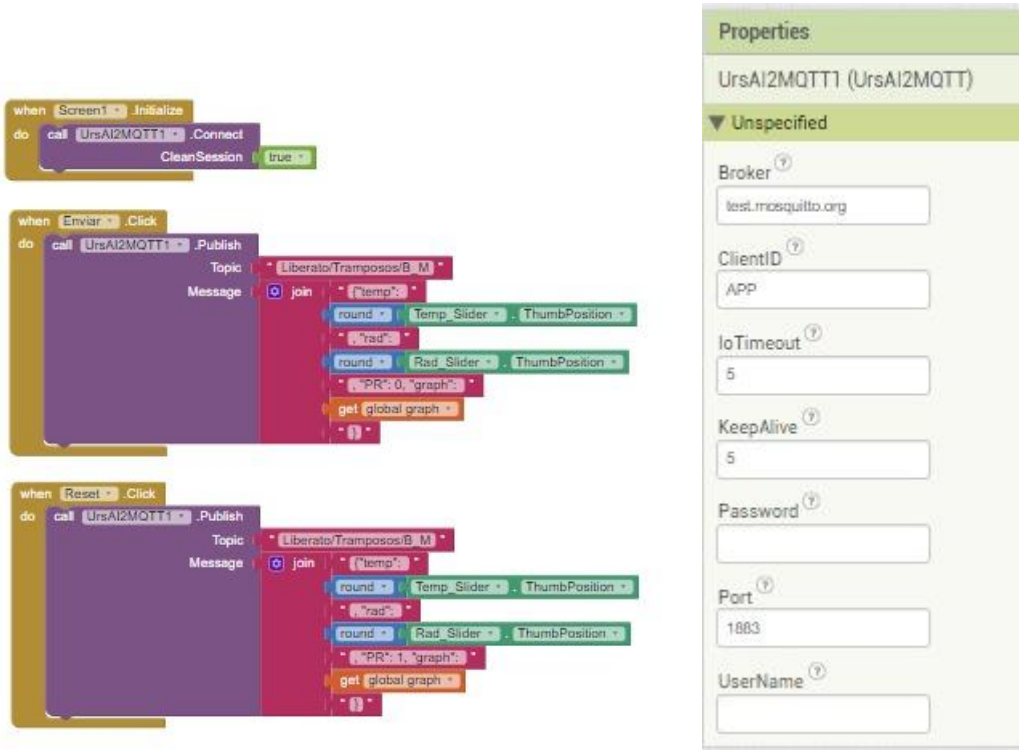
Quanto ao arquivo mqtt.py, deixamos como padrão já tanto no app quanto no software, os padrões de comunicação MQTT utilizados, tais como, broker, tópico, port, client id, etc. Diferentemente dos outros ajustes que caso o usuário necessite é de fácil mudança, os parâmetros MQTT já vem ajustados tanto para o aplicativo quanto para o software, o que dificulta na hora de trocar os devidos parâmetros.

4.2 Aplicativo

Para trocar os parâmetros de comunicação MQTT no software é pelos seguintes campos:

```
host = 'test.mosquitto.org'
port = 1883
topic = 'Liberato/Tramposos/B_M'  # Tópico MQTT utilizado para a comunicação
client_id = '20000298'  # ID do cliente
```

Para trocar os parâmetros de comunicação MQTT no aplicativo é pelos seguintes campos:



5. MENSAGENS RECEBIDAS

As mensagens recebidas são um ponto crucial para o software funcionar, o exemplo de mensagem que o software espera receber são as seguintes:

{"temp": 34, "rad": 692, "PR": 0, "graph": 0}
{'temp': 34, 'rad': 692, 'PR': 0, 'graph': 0}
{"temp": 34, "rad": 264, "PR": 0, "graph": 0}
{"temp": 61, "rad": 831, "PR": 0, "graph": 1}
{'temp': 61, 'rad': 831, 'PR': 0, 'graph': 1}
Plotando gráfico potência
{"temp": 20, "rad": 104, "PR": 0, "graph": 1}
{'temp': 20, 'rad': 104, 'PR': 0, 'graph': 1}
Plotando gráfico potência
{"temp": 20, "rad": 104, "PR": 1, "graph": 1}
{'temp': 20, 'rad': 104, 'PR': 1, 'graph': 1}
Resetando gráficos

{"temp": 20, "rad": 104, "PR": 0, "graph": 1}

{'temp': 20, 'rad': 104, 'PR': 0, 'graph': 1}

Plotando gráfico potência

Plotando gráfico IV

{"temp": 20, "rad": 104, "PR": 1, "graph": 2}

{'temp': 20, 'rad': 104, 'PR': 1, 'graph': 2}

Resetando gráficos

{"temp": 20, "rad": 104, "PR": 0, "graph": 2}

{'temp': 20, 'rad': 104, 'PR': 0, 'graph': 2}

Plotando gráfico IV

Conexão estabelecida com sucesso!

Conexão estabelecida com sucesso!

Conexão estabelecida com sucesso!

Se for necessário trocar a mensagem que o software irá receber, é necessário alterar o processamento da imagem (no arquivo mqtt.py) e maneira que o software interpreta cada campo da mensagem.

```
def subscribe(client: mqtt_client):
    # Callback para quando uma mensagem é recebida
    def on_message(client, userdata, msg):
        print(msg.payload.decode())  # Exibe a mensagem recebida
        if is_json(msg.payload.decode()):
            print(json.loads(msg.payload.decode()))
            msg_dicpy = json.loads(msg.payload.decode())
            temperatura = msg_dicpy["temp"]
            radiacao = msg_dicpy["rad"]
            reset = msg_dicpy["PR"]
            opc = msg_dicpy["graph"]
            if reset == 1:
                graphics.reset_graphics()
            elif opc == 1:
                fotovoltaica.calculo_pot(temperatura, radiacao)
                graphics.plot_pot(graphics.curvas_pot)
            elif opc == 2:
                fotovoltaica.calculo_iv(temperatura, radiacao)
                graphics.plot_iv(graphics.curvas_iv)
            elif opc == 3:
                fotovoltaica.calculo_iv(temperatura, radiacao)
                graphics.plot_iv(graphics.curvas_iv)
                fotovoltaica.calculo_pot(temperatura, radiacao)
                graphics.plot_pot(graphics.curvas_pot)

    client.subscribe(topic)
    client.on_message = on_message  # Definição do callback a ser utilizado
```

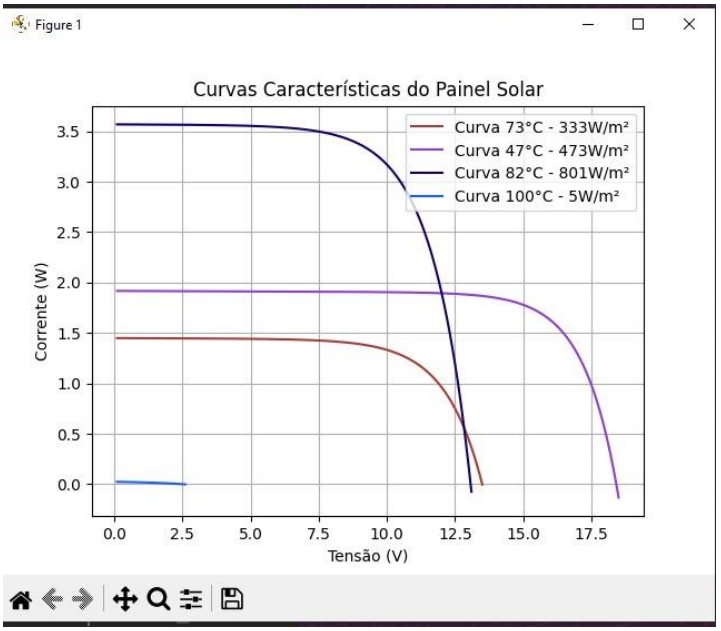
É possível ver na estrutura de *if's* o tratamento que o software faz nas mensagens, para alterar o padrão da mensagem é necessário alterar cada um dos campos do dicionário recebido e também a comparação em cada um dos *if's*.

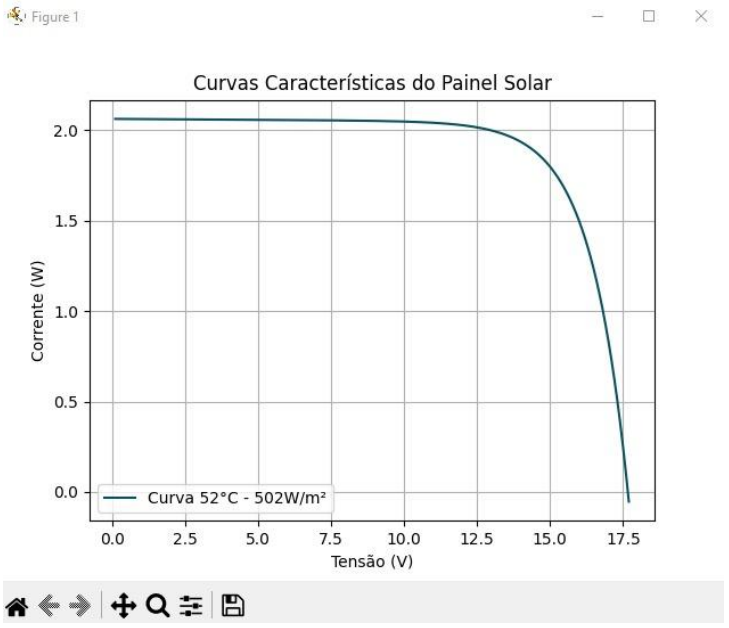
6. POSSÍVEIS PROBLEMAS E SUAS SOLUÇÕES

Os possíveis erros mais prováveis que o sistema pode apresentar são os seguintes.

- Falha na comunicação com o broker ‘test.mosquitto.org’. Nesse caso indicamos a troca de broker para algum outro como o ‘tago.io’ ou algum outro broker que o técnico for mais familiarizado.
- Falha na interpretação da mensagem. Esse erro só irá ocorrer se o técnico quiser fazer a alteração no padrão das mensagens, esse texto é muito característico porquê assim que o software recebe a mensagem, ele printa a mesma, entretanto se a interpretação estiver errada, o código para e reporta um erro na saída do terminal.
- Um problema que pode ocorrer no aplicativo é a desconexão com o broker, para resolver esse problema, basta fechar e abrir o app novamente. Caso não resolva, o problema pode ser por conta do broker ou por conta das funções de comunicação mal parametrizadas no app.

7. RESULTADOS ESPERADOS





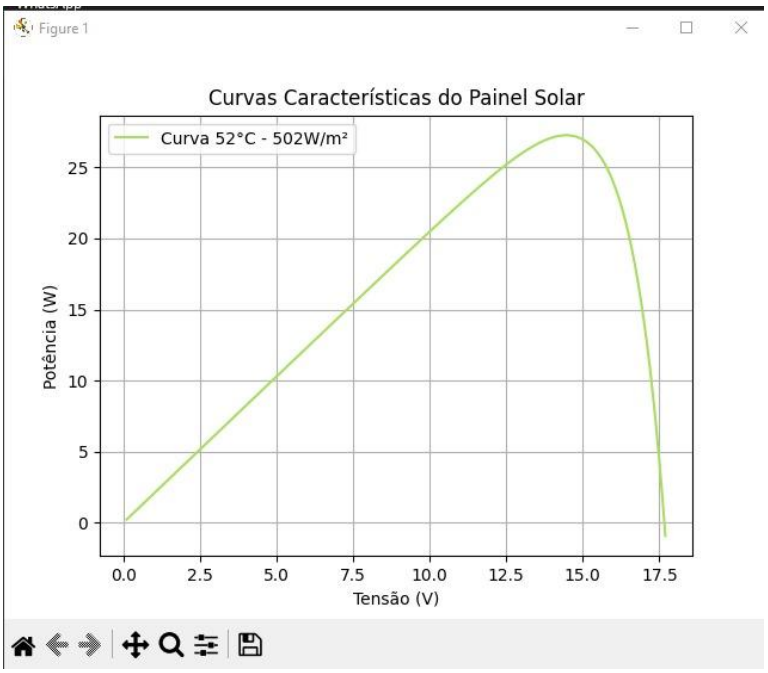


Figure 1
Curvas Características do Painel Solar
Curva 52°C - 502W/m²
Potência (W)
Tensão (V)
0
5
10
15
20
25
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

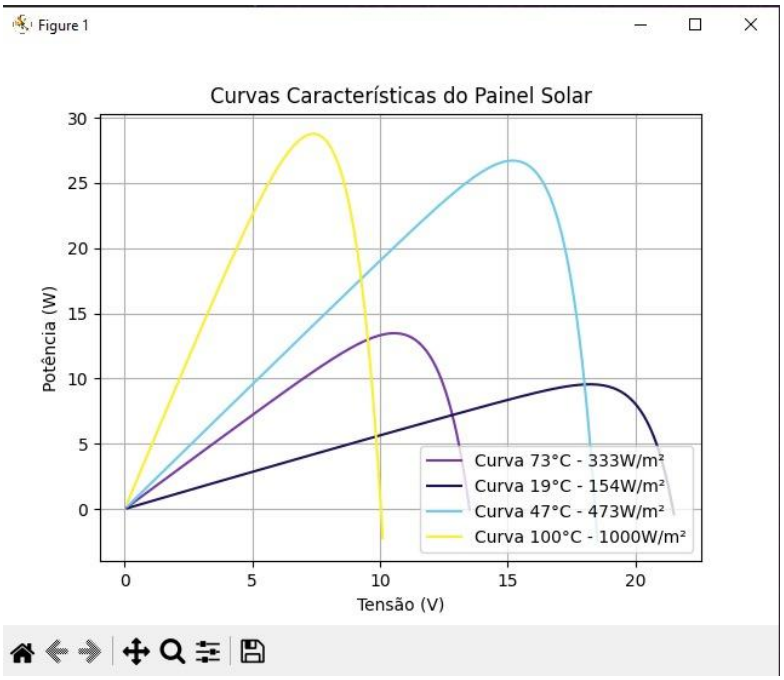


Figure 1
Curvas Características do Painel Solar
Curva 73°C - 333W/m²
Curva 19°C - 154W/m²
Curva 47°C - 473W/m²
Curva 100°C - 1000W/m²
Potência (W)
Tensão (V)
0
5
10
15
20
25
30