Sistemas Distribuidos para Automacao
Documentação do TP (Parte 1)

Vitor Holler Teixeira de Barros (2019021832)

2024

UFMG, Belo Horizonte

Sumário

1 Arquitetura do Trabalho

O projeto é composto por quatro principais arquivos de código Python:

- `populate.py`: Arquivo para criar dados iniciais no servidor OPC/UA.
- `process.py`: Simulação do processo do alto-forno e controle ON/OFF de temperatura.
- `CLP.py`: Controlador Lógico Programável (CLP) que gerencia a comunicação entre o servidor OPC UA e os clientes TCP/IP.
- `client_tcp_ip.py`: Cliente TCP/IP para envio de dados de controle.
- `MES.py`: Sistema de Execução de Manufatura (MES) que coleta e salva dados do processo.

1.1 populate.py

Script que adiciona valores iniciais para as 3 variáveis que estão no servidor OPC/UA, "calor", "temperatura de referencia"e "temperatura atual".

1.2 process.py

O função de controle é ON/OFF, ou seja, quando o auto forno atinge a temperatura desejada, o processo para de jogar calor no sistema. O calor inserido no sistema tem o valor fixo de 10000. Além disso, foi implementado a integração pelo método Runge Kutta

1.3 CLP.py

Uma função client_opc_ua() que faz o envio de um novo temperatura de referencia e faz a leitura da temperatura atual e o calor para o servidor OPC/UA. Uma função server_tcp_ip() que faz a gestão das conexões TCP/IP. Além disso, essa função inicia dois novos processos, send_messages() e receive_messages(), estes que são responsáveis por enviar os dados de temperatura atual e calor para o cliente TCP/IP conectado e que escuta os dados enviados por esse mesmo cliente TCP/IP para temperatura de referencia, respectivamente. A comunicação entres os processos client_opc_ua(), send_messages() e receive_messages() é feita usando multiprocessing.Queue().

1.4 client_tcp_ip.py

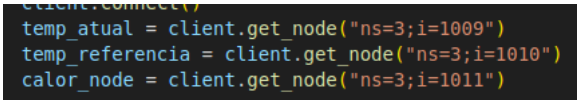
Contem um processo para a escrita em um arquivo chamado "historiador.txt", nesse arquivo é escrito toda alteração ocorrida após a conexão do cliente TCP/IP. Um outro processo para receber as mensagens enviadas pelo servidor TCP/IP. Um sub-processo para o usuário enviar a nova temperatura de referencia para o servidor TCP. Além disso, este arquivo também inicia um processo que inicializa um servidor IPC, responsável por receber a temperatura de referencia enviada pelo usuário a partir do sub-processo detalhado anteriormente e enviar esse dado para uma fila que o processo de escrita no arquivo "historiador.txt"consome.

1.5 MES.py

Um processo client_opc_ua() e outro de escrita num arquivo chamado "MES.txt".

2 Como rodar

Primeiramente é necessário inicializar as variaveis no servidor OPC/UA. Para facilitar existe um arquivo chamado config.py, no qual você pode colocar os respectivos valores de ns e i de cada uma das variáveis do seu servidor. Por padrão, os valores são os contidos na imagem abaixo.



Além disso, é necessário instalar as dependencias necessárias para este projeto. Eu recomendo iniciar um virtual env. Se estiver rodando no linux, executar códigos abaixo.

```
virtualenv TP_SDA_VITOR
source TP_SDA_VITOR/bin/activate
```

Após isso, você deve conferir se está na mesma pasta em que baixou os arquivos em python do moodle. Aqui está o link do github facilitar o clone. Link do GitHub

```
git clone git@github.com:vitorHoller/TP_SDA.git
cd TP_SDA
python3 -m pip install requirements.txt
```

Agora abra 5 terminais diferentes, confira se os terminais estão dentro da pasta clonada do git

```
python3 populate.py
python3 process.py
python3 MES.py
python3 CLP.py
python3 server_tcp_ip.py
```

Após fazer isso, basta colocar diferentes temperaturas de referencia e ver como o sistema atua. No github existe um arquivo .mp4 da gravação da minha tela rodando o projeto.