Trabalho Prático Final de SDA

Controle de um Drone (Quadcopter) via Integração com Servidor OPC UA

Professor: Armando Alves Neto
Departamento de Engenharia Eletrônica – Escola de Engenharia
Universidade Federal de Minas Gerais

Instruções:

1. Este trabalho prático tem por objetivo auxiliar a consolidação de conceitos vistos em classe.
2. Pode ser realizado em **dupla**.
3. Data de entrega: **04/12/2025 (impreterivelmente)**
4. **Trabalhos copiados terão nota zero.**

1 Descrição do problema

Na Indústria 4.0, drones atuam como sensores móveis e agentes autônomos para inspeção visual, inventário, monitoramento de segurança e apoio à manutenção, integrando-se a sistemas ciberfísicos e ao IIoT. Em vez de operarem isolados, passam a compor ecossistemas conectados, nos quais dados e comandos circulam entre chão de fábrica, supervisórios, analytics e sistemas corporativos.

Figura 1: Drone de inspeção industrial.

O OPC UA (*Open Platform Communications Unified Architecture*) é o elo dessa integração: oferece um modelo de informação padronizado e seguro que torna os dados do drone — como setpoints e telemetria — acessíveis a qualquer cliente autorizado, independentemente de fornecedor ou plataforma. Com isso, decisões sobre onde o drone deve ir podem ser tomadas por aplicações externas, enquanto múltiplos consumidores leem a mesma telemetria para rastreabilidade, KPIs e automação de mais alto nível, viabilizando escalabilidade e desacoplamento típicos da Indústria 4.0.

Nesse trabalho, iremos desenvolver uma aplicação que permita o controle de um drone simulado no *CoppeliaSim* através da troca de dados com um servidor OPC UA fornecido pela *Prosys*, explorando técnicas de comunicação industrial e automação.

2 Preparando o ambiente

Para começar, antes precisamos preparar o ambiente de simulação.

2.1 Instalar o *CoppeliaSim*

O *CoppeliaSim* (antigo V-REP) é um ambiente de simulação 3D usado para projetar, simular e testar robôs e sistemas automatizados. Ele permite que se crie cenários complexos com robôs, sensores, atuadores e lógica de controle, tudo em um ambiente visual. Seu principal diferencial é a flexibilidade e interatividade, sendo muito utilizado em pesquisa, ensino e desenvolvimento de aplicações em robótica. Para usar o simulador nesse trabalho, faça o seguinte:

- Acesse o site oficial: `https://www.coppeliarobotics.com`
- Selecione a versão mais recente compatível com seu sistema operacional (Windows, Linux ou macOS). Para a maioria dos projetos, a versão EDU (gratuita) já é suficiente.
- Faça o download e extraia o conteúdo. Depois execute o CoppeliaSim. No Windows: abra o arquivo *coppeliaSim.exe*. No Linux/macOS: execute o script *coppeliaSim.sh* pelo terminal (pode precisar dar permissão com chmod +x).
- Por fim, abra o arquivo *drone.ttt* disponível no Moodle (vide figura 2). Ele contém um drone e quatro bandejas dispostas no solo que precisarão ser periodicamente inspecionadas. Uma câmera acoplada ao robô será utilizada para essa missão.

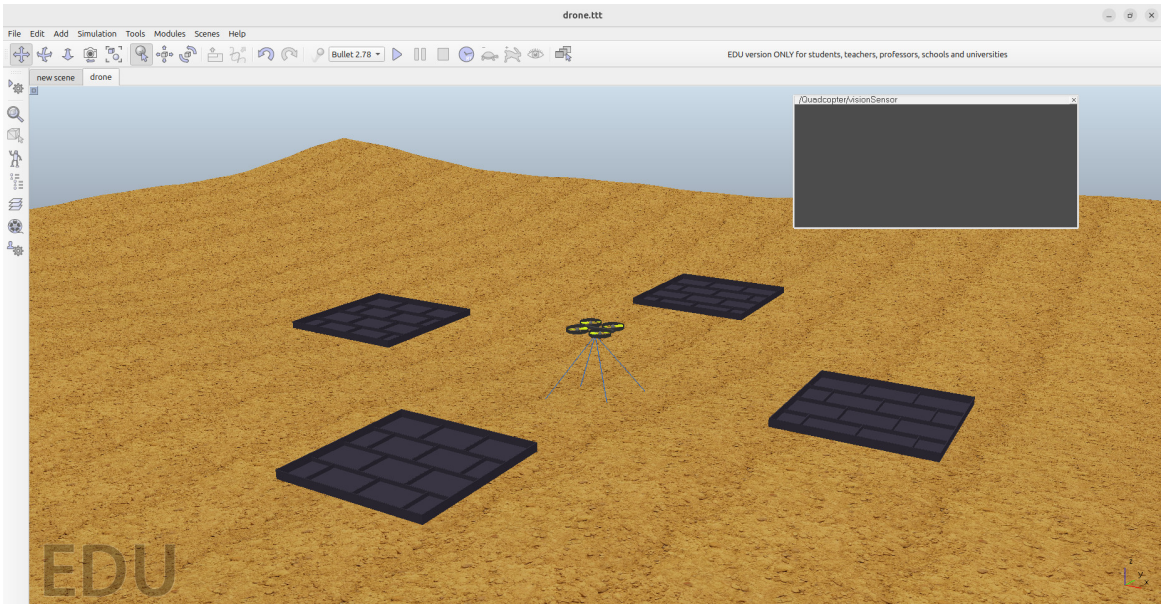

Figura 2: Drone genérico no *CoppeliaSim*.

2.2 Instalar o Prosys OPC UA Simulation Server

O *Prosys OPC UA Simulation Server* é um software gratuito desenvolvido pela *Prosys OPC* que simula um servidor OPC UA completo. É amplamente usado para testes, desenvolvimento e ensino de aplicações baseadas em OPC UA, sem a necessidade de hardware real (como CLPs ou sensores). Para usá-lo, faça o seguinte:

- Acesse o site oficial `https://www.prosysopc.com/products/opc-ua-simulation-server/` e cadastre-se (é necessário preencher um formulário com seu nome, email e instituição para baixar a versão gratuita). Após o cadastro, você receberá um link de download por email.
- Baixe o instalador correspondente ao seu sistema operacional e instale o servidor seguindo as instruções padrão.

- Abra o *Prosys OPC UA Simulation Server*. Quando o servidor iniciar, ele já carrega algumas variáveis simuladas (como temperaturas, níveis, motores, etc.). Você pode navegar pelo espaço de endereçamento (*Address Space*) e visualizar os dados simulados.

2.3 Integrando os dois simuladores

Tendo realizado com sucesso os passos anteriores, é hora de configurar o servidor OPC e integrá-lo ao simulador de robôs.

2.3.1 Criando um *Object* com 6 variáveis no Prosys

- Na aba *Objects*, dentro da pasta *Objects::FolderType*, clique em *Add* > *New Object*. Crie um objeto chamado *Drone* para emular as variáveis do robô.
- Selecione esse novo objeto e acrescente 3 variáveis do tipo *Numeric* para a posição do drone (*DroneX*, *DroneY*, *DroneZ*). Não se esqueça de atribuir valores iniciais a cada uma das juntas.
- Repita o procedimento, agora para criar 3 variáveis para a posição-referência de controle (*TargetX*, *TargetY*, *TargetZ*).
- Verifique se sua configuração é semelhante à figura 3.

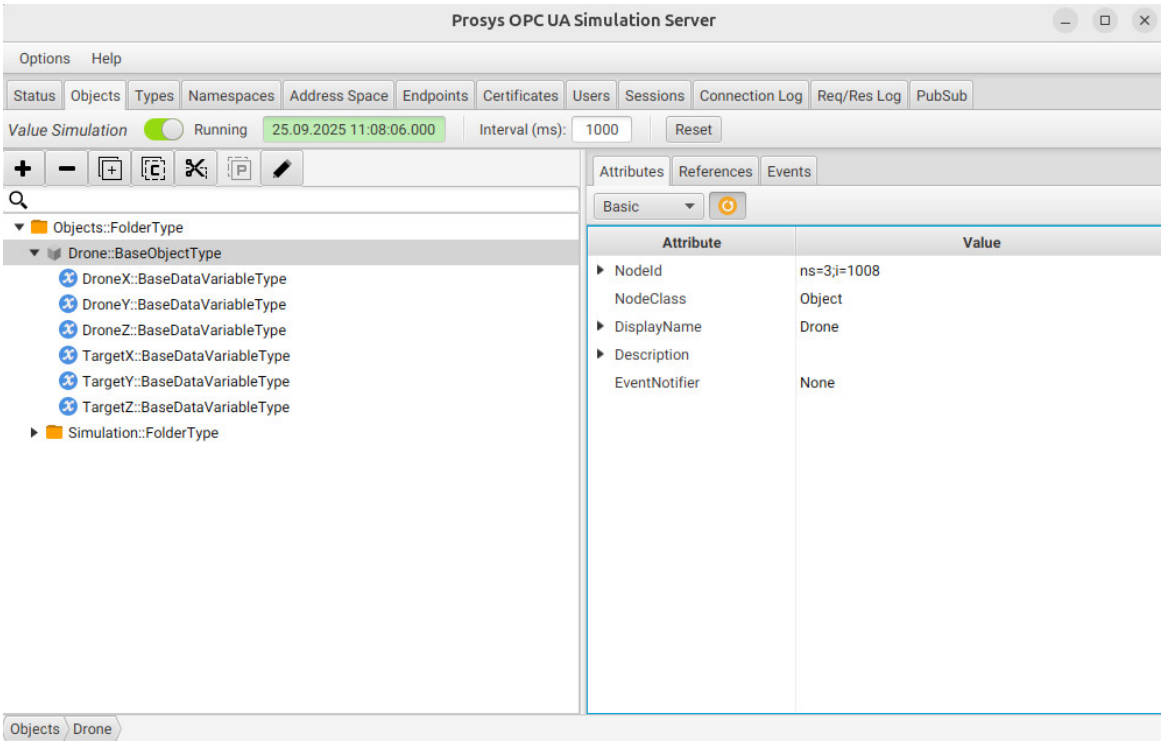

Figura 3: Configuração do Prosys OPC UA Simulation Server.

2.3.2 Executando a ponte de comunicação

No Moodle há um arquivo chamado *bridge.py*, que cria um cliente OPC UA para se comunicar com o servidor. Ele também acessa o *CoppeliaSim* via API ZMQ, fazendo a ponte entre as variáveis do servidor OPC e o drone simulado.

- Para rodar o cliente, é necessário instalar a biblioteca *opcua* para Python (*pip install opcua*).

- Como vamos controlar um robô externamente via Python, será necessário instalar a API remota ZMQ (*coppeliasim_zmqremoteapi_client*). Para isso, acesse `https://manual.coppeliarobotics.com/en/zmqRemoteApiOverview.htm`.
- Pronto! Após executar o script, o servidor estará apto a receber comandos de movimentação para o robô.

3 O que deve ser feito

- Crie um programa chamado *CLP* que contenha um cliente OPC UA (para ler e atuar no drone via servidor) e um servidor TCP/IP (responsável por receber conexões de clientes TCP/IP que desejem controlar o processo). *Sugestão*: crie duas threads, uma para cada finalidade. O cliente OPC irá ler a posição do drone ($Drone_{X,Y,Z}$) e enviar comandos de referência de posição ($Target_{X,Y,Z}$), conforme ilustrado na figura 4.
- Crie um programa que opere como cliente TCP/IP para ler e enviar dados remotamente ao processo (para fins práticos, esse cliente pode rodar na mesmo máquina do servidor). Dados de controle e supervisão devem ser exibidos ao operador via um sinótipo gráfico ou exibidas na própria tela do terminal (a parte gráfica é meramente opcional, mas vale pontos-extra). Todas essas informações devem ser registradas em um arquivo denominado "historiador.txt". O operador deverá, via supervisório, escolher pontos de referência para inspeção visual do drone, recebendo de volta os valores de posição da robô e o *timestamp* associado.
- Por fim, crie um segundo cliente OPCUA em outro programa, encapsulado dentro de outro servidor OPCUA, numa arquitetura que chamamos de *chained server*. O servidor fornecerá as mesmas informações do drone para outro cliente, chamado *MES*, que irá ler dados do processo para salvar em um arquivo chamado "mes.txt".

A figura 4 ilustra a estrutura da rede distribuída a ser implementada.

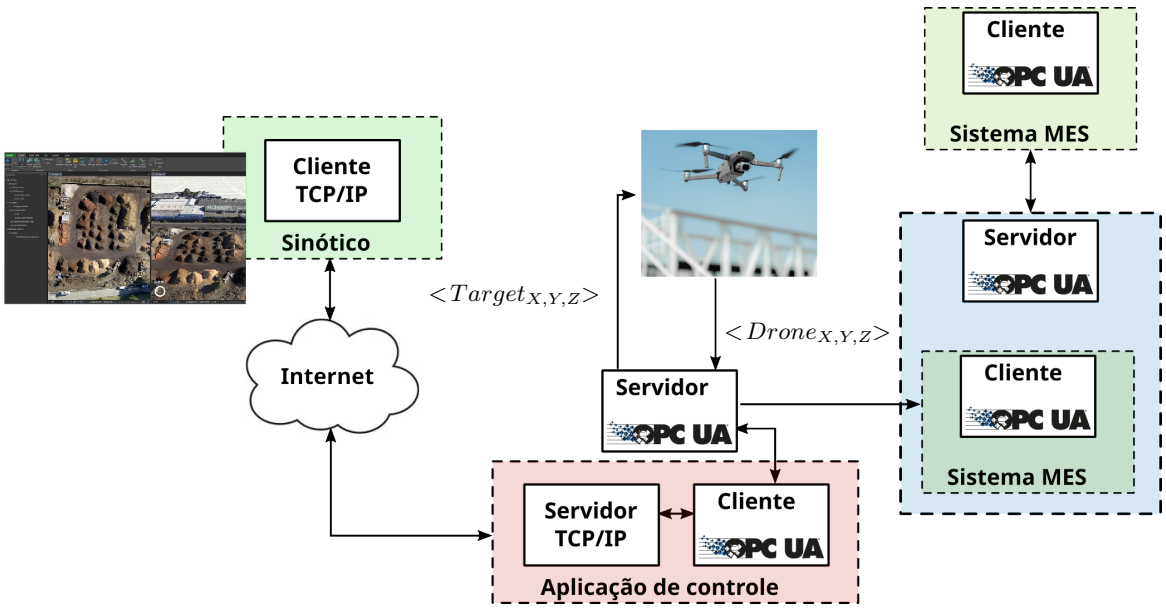

Figura 4: Sistema distribuído para operação do processo.

4 O que deve ser entregue:

- (50% da nota) Códigos contendo os arquivos dos programas. Será permitido o uso das linguagens python, c ou c++.

- (10% da nota) Um documento “readme.txt” com instruções de compilação e operação do sistema.
- (20% da nota) Relatório de descrição do trabalho em “.pdf”.
- (20% da nota) Um vídeo de até 5 minutos apresentando o grupo e o funcionamento do trabalho (link para o Youtube deve estar no relatório e no readme).
- Tudo deverá ser entregue dentro de um arquivo .zip (ou .rar, .tar, etc) com o nome dos membros do grupo. Ex. <fulano_e_siclano.zip>.