Departamento de Engenharia Eletrônica
SISTEMAS DISTRIBUÍDOS PARA AUTOMAÇÃO – 2020/1

Trabalho Prático sobre OPC e *Sockets* TCP/IP – Valor: 20 pontos

Descrição

O transporte de minério de ferro das jazidas de extração até as usinas siderúrgicas ou as instalações portuárias para exportação é tradicionalmente feito por vias ferroviárias. Tipicamente, o minério extraído da lavra passa pelos processos de britagem e moagem até ficar com uma granularidade suficientemente baixa. Este minério (designado desde então como "finos de minério") é então depositado em silos, sob os quais os vagões da composição ferroviária vão sendo sucessivamente posicionados para que o minério seja escoado para os mesmos.

Figura 1: Sistema de carregamento de minério de ferro em vagões. Crédito da imagem: Vale

Figura 2: Transporte de minério de ferro por via ferroviária. Crédito da imagem: Vale

O carregamento de minério em vagões a partir de silos é um processo muito sensível, por razões de segurança, de dificuldade de controle e de otimização de logística [1], [2]. Em primeiro lugar, o posicionamento deve ser muito preciso pois o vagão desloca-se sob o silo para que a carga de minério seja igualmente distribuída ao longo de seu comprimento. Em segundo lugar, a operação de descarga do silo deve ser feita em condições controladas para que sua comporta de escoamento seja aberta e fechada em instantes precisos, de forma a evitar que os finos de minério caiam sobre os trilhos ferroviários no interstício entre os vagões, o que prejudicaria a via ferroviária além do desperdício de minério em si. E, finalmente, a carga de minério em cada vagão deve ser precisamente dosada para evitar tanto subcarga (diminuindo a eficiência do transporte) quanto sobrecarga (que poderia comprometer o transporte, provocando descarrilhamento da composição ferroviária). Atualmente, sofisticados sistemas de controle e automação utilizando medições por radar e *laser* são empregados nas indústrias de mineração com o objetivo de maximizar a segurança e a eficiência do processo de carregamento de vagões a partir de silos de minérios [1].

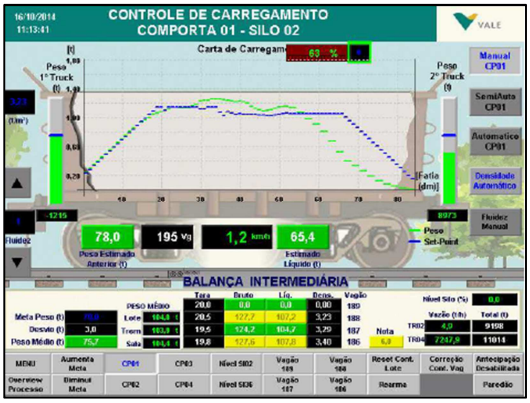

Figura 3: Sistema supervisório para carregamento de minério de ferro em vagões. Fonte: [1]

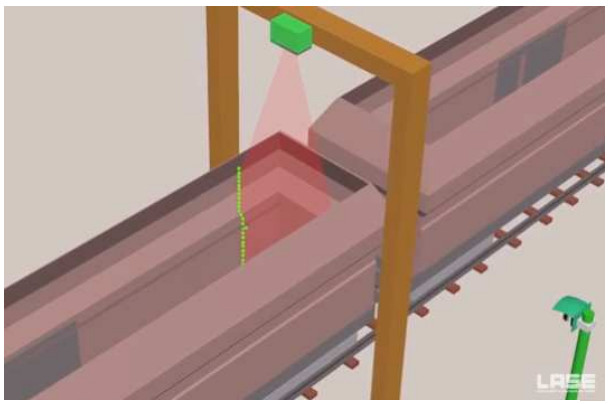

Figura 4: Sistema de medição a *laser* para vagões de carga. Fonte: [3]

Considere uma empresa de mineração em vias de automatizar o processo de carregamento de vagões em uma de suas minas. A empresa dispõe de um servidor de alta capacidade de processamento, com S.O. Linux, no qual uma aplicação de otimização de carregamento de vagões é executada. Esta aplicação necessita receber dados em tempo real do posicionamento dos vagões sob o silo e, em função destas informações, calcula a carga de minério a ser depositada no vagão e determina os instantes precisos de abertura e fechamento da comporta do silo, informações estas enviadas para um Controlador Lógico Programável (CLP) que comanda tanto o acionamento da comporta do silo quanto a velocidade de movimentação dos vagões. O acesso às variáveis de processo deste CLP é feito por meio de um computador com S.O. Windows no qual executa um servidor OPC "clássico". A disparidade dos sistemas envolvidos requer que uma aplicação de software seja desenvolvida com o propósito de integrar o sistema de otimização ao CLP via *sockets* TCP/IP. A aplicação, assim, funcionará simultaneamente como "cliente OPC" e "servidor de *sockets*". A figura 2 exemplifica a estrutura desta aplicação.

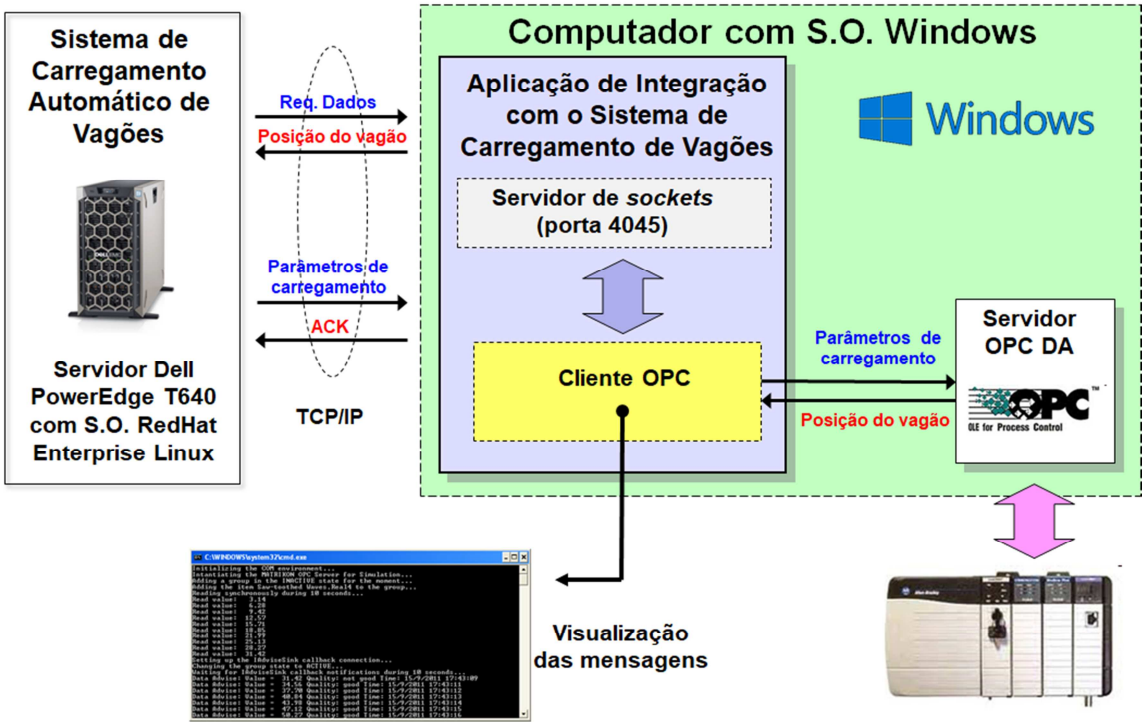

Figura 5: Arquitetura da aplicação de software a ser desenvolvida

A aplicação deve ser desenvolvida nas linguagens C ou C++ com o ambiente *Microsoft Visual Studio Community Edition* (você deverá instalar e utilizar no mesmo a "carga de trabalho" *Desktop development with C++*). O código referente ao cliente OPC poderá ser baseado em qualquer *toolkit* de código aberto existente, incluindo o *Simple OPC Client* apresentado pelo professor em classe.

Características da aplicação a ser desenvolvida

1. Módulo "servidor de *sockets*"

   O módulo referente ao servidor de *sockets* recebe dois tipos de mensagens provenientes do sistema de otimização do carregamento de vagões: (1) mensagens periódicas (a cada 2 segundos)[1] de solicitação de posicionamento dos vagões e (2) mensagens aperiódicas com os parâmetros de carregamento do vagão a serem enviados ao CLP. No primeiro caso, o servidor de *sockets* responde imediatamente com uma mensagem contendo os dados da posição do vagão; e, no segundo caso, responde com uma mensagem de confirmação (ACK). Todas as mensagens são estruturadas como

[1] Numa situação real, estas mensagens seriam recebidas do sistema de otimização em uma escala de tempo muito menor, usualmente a cada 100 ms ou menos. Aqui estamos exagerando grandemente esta escala para fins didáticos.

cadeias de caracteres ASCII compostas de campos variáveis de um a dez caracteres, separados pelo delimitador "|" (barra vertical), como detalhado a seguir.

- Mensagem de solicitação de dados (Sistema de otimização → Aplicação de software):

Campo	Tipo
1. Código da mensagem	Inteiro (sempre "55")
2. Número sequencial da mensagem	Inteiro (NNNNN)

Exemplo: "`55|009852`"

- Mensagem de posicionamento do vagão (Aplicação de software → Sistema de otimização):

Campo	Tipo	Item correspondente no ***Matrikon OPC Simulation Server***
1. Código da mensagem	Inteiro (sempre "55")	----------
2. Número sequencial da mensagem	Inteiro (NNNNN)	----------
3. Velocidade do vagão (cm/s)	Inteiro (NNNNN)	*Random.UInt1*
4. Estado do sensor de início de carga (L/D)*	Inteiro (NNNNN)	*Random.UInt2*
5. Estado do sensor de final de carga (L/D)	Inteiro (NNNNN)	*Random.UInt4*
6. Leitura da balança de carga no vagão (kg)	Real (NNNN.NN)	*Random.Saw-toothed Waves*

*Ligado/Desligado

Exemplo: "`55|12345|00001|00000|00001|1233.55`"

- Mensagem com parâmetros de carregamento do vagão (Sistema de otimização → Aplicação de software):

Campo	Tipo	Item correspondente no ***Matrikon OPC Simulation Server***
1. Código da mensagem	Inteiro (sempre "99")	----------
2. Número sequencial da mensagem	Inteiro (NNNNN)	----------
3. Tempo de abertura da comporta do silo (seg.)	Inteiro (NNNNN)	*Bucket Brigade.UInt1*
4. Quantidade de minério a carregar (kg)	Real (NNNN.NN)	*Bucket Brigade.Real4*

Exemplo: "`99|98765|00300|1250.00`"

IMPORTANTE: Para fins de simulação da aplicação, esta mensagem será disparada pelo cliente de *sockets* sempre que o usuário digitar a tecla "c" no computador.

- Mensagem de confirmação (ACK) (Aplicação de software → Sistema de otimização):

Campo	Tipo
1. Código da mensagem	Inteiro (sempre "00")
2. Número sequencial da mensagem	Inteiro (NNNNN)

Exemplo: "`00/55432`"

Em todas as mensagens, os números sequenciais são consecutivos, devendo ser continuamente incrementados pelo emissor da mensagem e recomeçando de zero quando a contagem máxima for alcançada. Assim, por exemplo, uma possível sequência de mensagens trocadas teria os seguintes números sequenciais:

Mensagem	Origem	Número sequencial
Requisição de dados	Sistema de Otimização	000001
Mensagem de dados de processo	Servidor de *sockets*	000002
Requisição de dados	Sistema de Otimização	000003
Mensagem de dados de processo	Servidor de *sockets*	000004
Envio de parâmetros de carregamento	Sistema de Otimização	000005
ACK	Servidor de *sockets*	000006
...	...	000007

A porta TCP a ser usada na comunicação via *sockets* deverá ser a **4045**. Todas as mensagens que trafegam entre o sistema de otimização e a aplicação de software deverão ser exibidas na janela de console (ou, opcionalmente, interface gráfica) associada à aplicação.

Caso haja perda de comunicação entre o sistema de otimização e a aplicação de software (p. ex. devido à desconexão do cabo de rede), esta última deve descartar qualquer mensagem pendente (a receber ou a enviar) e reconectar-se automaticamente com o primeiro.

2. Módulo "cliente OPC"

O módulo correspondente ao cliente OPC deve executar as seguintes ações:

a) Ler do servidor OPC, de forma assíncrona (ou seja, via notificações por *callback*), as variáveis que consistem da mensagem de posicionamento de vagão, de forma que as mesmas sempre estejam com valores atualizados a serem enviados ao sistema de otimização. A taxa de atualização das leituras deverá ser de 1000 ms.

b) Escrever no servidor OPC, de forma síncrona, os valores correspondentes aos parâmetros de carregamento de vagão recebidos do Sistema de otimização.

Todos os valores lidos e escritos, pelo cliente OPC, do/no servidor OPC também deverão ser exibidos na janela de console associada à aplicação.

3. OPCIONAL: módulo VBA para *o Microsoft Excel®*

A especificação OPC DA oferece duas formas de interação entre clientes e servidores OPC: a forma *custom* e a forma *automation*. A primeira é a que usamos para desenvolver clientes e servidores OPC com base em linguagens que suportam apontadores, como C e C++, ao passo que a segunda é específica para linguagens que não têm este suporte como o *Visual Basic* e o *Visual Basic for Applications* (VBA). A forma *automation* é particularmente apropriada para que certas aplicações, como p. ex. o *Microsoft Excel®,* possam executar código VBA que faça acesso a um servidor OPC, permitindo que células específicas de uma planilha possam ser mapeadas em itens de processo presentes neste servidor. Tal acesso, contudo, requer um componente de software referido como *automation wrapper*, em geral fornecido na forma de uma *Dynamic Linking Library* (arquivo DLL) como parte de um produto OPC comercial. Felizmente, o *Matrikon OPC Simulation Server* a ser utilizado neste trabalho (vide tópico sobre "observações importantes" adiante) já inclui um *automation wrapper* que possibilita este tipo de interação.

No presente trabalho, poderá ser desenvolvido pelos alunos um módulo opcional que receberá pontuação adicional de 5 pontos, e corresponderá a um programa VBA contido em uma planilha *Excel®* que possibilitará a um operador:

a) Ler do servidor de otimização, de forma assíncrona, as variáveis de processo que constam dos campos 3 a 6 da mensagem de posicionamento de vagão enviada pela aplicação de software ao sistema de otimização, cujos valores deverão ser visualizados em células específicas da planilha;

b) Escrever no servidor OPC, de forma síncrona, os parâmetros de carregamento do vagão (campos 3 e 4 da respectiva mensagem enviada pelo sistema de otmização à aplicação de software), os quais deverão ser digitados em células específicas da planilha. Este recurso, numa situação real, permitiria ao operador acessar ou alterar diretamente estes parâmetros.

Para tal, a planilha *Excel®* deverá ser construída da seguinte forma:

- Deverá conter quatro botões clicáveis pelo operador, tratados pelo código VBA, que executem respectivamente as ações de (1) conexão ao servidor OPC, (2) solicitação de leitura por notificação (leitura assíncrona), (3) solicitação de escrita síncrona e (4) desconexão com o servidor OPC. O professor disponibilizará uma pequena aplicação de exemplo como ponto de partida para este módulo VBA.
- O *ProgID* do servidor OPC e os nomes dos itens OPC correspondentes aos campos 3 a 6 da mensagem de dados de processo devem ser lidos de células específicas da planilha, permitindo ao operador alterá-las sem necessidade de intervenção no código VBA.
- Todos os valores lidos deverão apresentar também os respectivos *time-stamps* e qualidade em células apropriadas.

No desenvolvimento deste módulo, será necessário registrar no editor VBA do *Excel®* o *automation*

wrapper disponibilizado como parte do *Matrikon OPC Simulation Server*:

1) No *Excel®*, abra o editor VBA (*Alt + F11*) e selecione *Ferramentas → Referências*;
2) Na janela de referências que se abrir, marque a opção "Matrikon OPC Automation 2.0". (Naturalmente, você já deverá ter previamente instalado o *Matrikon OPC Simulation Server*.)

IMPORTANTE:

1) A planilha Excel a ser submetida deverá estar completa, contendo todos os campos acima e a respectiva lógica em VBA. Não serão consideradas planilhas incompletas.
2) Esta atividade opcional é complementar e não pode ser entregue de forma isolada. A mesma será avaliada apenas se o trabalho principal for submetido.

4. OPCIONAL: Interface Gráfica

O desenvolvimento de uma interface gráfica não é requerido neste trabalho, bastando apenas a janela de console. Os melhoramentos descritos no quadro a seguir são opcionais e valerão pontuação adicional ao trabalho, a critério do professor, dentro dos limites estabelecidos.

Características opcionais	Pré-requisito	Pontuação adicional
Interface gráfica, em substituição à janela de console	----------	Até 3 pontos
Capacidade de realizar o *browsing* dos itens existentes no servidor OPC	Interface gráfica	Até 2 pontos
Capacidade de adicionar/remover grupos e itens no servidor OPC, bem como visualizar os valores dos itens	*Browsing* dos itens OPC do servidor	Até 3 pontos
Outros melhoramentos adicionais, a critério dos alunos, e desde que considerados relevantes pelo professor	----------	Até 2 pontos

Atenção: Tenha em mente que o desenvolvimento de interface gráfica associada a esta aplicação é uma tarefa de relativa complexidade e que demandará significativa dedicação de tempo.

Observações importantes

- O funcionamento da aplicação deverá ser testado através de programa específico a ser fornecido pelo professor, que simula o funcionamento do computador de processo (que, por sua vez, atuará como "cliente de *sockets*"). Tal programa será o mesmo com o qual o professor testará as aplicações desenvolvidas pelos alunos. Compile e instale este programa **em outro computador**, de forma que a comunicação entre o cliente e o servidor de *sockets* seja estabelecida entre computadores diferentes. **Atenção:** para que o Windows aceite conexões TCP proveniente de outros computadores, as propriedades de seu *firewall* devem ser alteradas.

- O funcionamento do módulo cliente OPC deverá ser testado com a utilização do software *Matrikon OPC Simulation Server*, gratuito e disponível no site `http://www.matrikon.com`. A instalação do *Simulation Server* inclui a instalação do módulo *Matrikon OPC Explorer*, um cliente OPC também gratuito. Este último poderá ser útil para auxiliar nos testes do módulo cliente OPC da aplicação a ser desenvolvida, como, por exemplo, a verificação da escrita de itens no servidor.

- Os módulos "servidor de *sockets*" e "cliente OPC" deverão ter funcionamento independente entre si, de modo que a eventual paralisação de um deles não impeça o funcionamento do outro. Por exemplo, caso haja desconexão de rede, o cliente OPC deverá continuar a ler dados do servidor OPC. (Sugestão: modele sua aplicação como um programa *multithread*, com uma *thread* correspondente ao servidor de *sockets* e outra ao cliente OPC, e use comunicação inter-processos ou memória compartilhada + sincronização para interação entre as *threads*.)

- O desenvolvimento do cliente OPC deverá ser feito pelos próprios alunos, seja integralmente ou com base em algum software de código aberto como, por exemplo, a versão modificada do *Simple OPC Server* apresentada pelo professor. O uso de qualquer software aberto tomado como base do desenvolvimento deverá estar claramente descrito na documentação, e as eventuais alterações e acréscimos feitos pelos alunos em tal código deverão estar indicados na forma de comentários apropriados, tanto no código-fonte quanto na documentação do projeto.

Instruções

1. O trabalho pode ser feito individualmente ou em grupos de 2 alunos. Atenção: a prova sobre este assunto poderá conter perguntas sobre o desenvolvimento deste trabalho.
2. A linguagem de programação deverá ser C ou C++.
3. Para desenvolver os programas do trabalho, utilize **somente** a ferramenta *Microsoft Microsoft Visual Studio Community* com a "carga de trabalho" *(workload) Desktop development with C++*. Crie uma conta gratuita do *Visual Studio Dev* da Microsoft (https://www.visualstudio.com/dev-essentials/) e baixe a ferramenta diretamente de https://www.visualstudio.com/downloads/.
4. O desenvolvimento da aplicação deverá ser feito de modo que, nos programas-fontes, toda referência a arquivos externos seja relativa ao diretório corrente (p. ex. "`..\..\teste.dat`") ao invés de absoluta (p.ex. "`C:\programas\dados\teste.dat`"), de modo que o respectivo projeto possa ser depositado, compilado e testado pelo professor em qualquer diretório de sua escolha.
3. As soluções devem ser submetidas via *Moodle* na forma de três arquivos, sendo os dois primeiros obrigatórios:
   - Arquivo PDF correspondente à documentação do trabalho, contendo:
     - Indicação precisa do **nome e sobrenome** dos autores;
     - Arquitetura de software empregada na solução, com descrição detalhada de aspectos como processos e *threads* empregados, aspectos de sincronização, comunicação inter-processos, estruturas de dados utilizadas, etc.;
     - Descrição de qualquer código aberto empregado como base do cliente OPC bem como as eventuais modificações feitas no mesmo;
     - Instruções de compilação e execução da aplicação;
     - Resultados de testes efetuados com a aplicação;
     - Quaisquer outras informações que auxiliem o professor a entender e testar o programa gerado.

     Atenção! Seja original em sua documentação. A presença de figuras e textos copiados deste enunciado prejudicará a pontuação atribuída a este item.
   - Arquivo .ZIP ou .RAR contendo a solução completa do trabalho, correspondente a:
     - Pasta (diretório) contendo todos os arquivos gerados pelo *Visual Studio C++*, de forma que o professor possa examinar os programas-fonte, compilá-los e testar seu funcionamento em seu próprio computador;
     - Quaisquer bibliotecas, arquivos .DLL, componentes, cabeçalhos (*headers),* etc. de terceiros necessários à compilação e geração do aplicativo. Se existentes, tais componentes devem estar presentes em subdiretórios específicos da pasta acima referida.
   - Arquivo executável correspondente à solução. Se necessário, mude a extensão do executável de .EXE para .DAT para evitar que seu provedor de acesso à Internet barre o envio de emails que contenham tais arquivos, como medida de segurança contra vírus. O tamanho máximo de arquivo passível de ser submetido será de 50Mb. *DICA: Você pode remover os arquivos de extensão .ipch e .db de sua solução, para diminuir o tamanho do arquivo ZIP ou RAR. Os mesmos serão automaticamente recriados pelo Visual Studio quando o projeto for recompilado.*

     ATENÇÃO: O projeto enviado deve ser totalmente auto-contido, ou seja, deve conter todo e qualquer código, bibliotecas externas, etc., necessários à sua compilação e execução. O professor não executará a carga e/ou instalação de nenhum componente externo necessário à execução do aplicativo. Programas que não compilem ou não executem, parcial ou integralmente, devido à ausência destes componentes serão considerados **não-entregues**.
   - Planilha Excel com módulo VBA (opcional).

5. **Verifique a consistência do arquivo ZIP (RAR) antes de enviá-lo.** Muitas avaliações são prejudicadas porque o arquivo ZIP (RAR) enviado encontra-se inconsistente, p. ex. por não conter todos os arquivos necessários à reprodução da pasta original. Teste seu arquivo ZIP (RAR) descompactando-o em outro computador e reproduzindo o projeto a partir do mesmo. Se o arquivo ZIP (RAR) recebido estiver inconsistente ou incompleto, o mesmo será desconsiderado e o trabalho será considerado como **não-entregue**. Não serão enviados, aos alunos, avisos de problemas com os arquivos enviados.
6. Data de entrega: até **23h55min** do dia **11/10/2020** via *Moodle*. Não serão consideradas exceções a este prazo, sob qualquer pretexto. Trabalhos submetidos após esta data, até **18/10/2020**, terão pontuação reduzida em 50%.
7. O professor não dará suporte sobre dúvidas ou problemas de utilização das ferramentas indicadas. O suporte necessário para o aprendizado das mesmas deverá ser obtido pelo aluno por seus próprios meios. A *web* possui farto material de apoio a estas ferramentas.
8. Dúvidas sobre o entendimento do trabalho ou sobre eventuais inconsistências deste enunciado deverão ser submetidas ao professor até o dia **09/10/2020**, de preferência na plataforma *Microsoft Teams* para que todos os alunos tenham acesso às mesmas e suas eventuais resoluções.

Quadro de pontuação do trabalho

A tabela seguinte apresenta os itens de pontuação a serem atribuídos ao trabalho.

Item	Pontuação
Funcionamento do servidor de *sockets*	3.0
Escrita síncrona de itens pelo cliente OPC	3.0
Leitura assíncrona de itens pelo cliente OPC	3.0
Funcionamento concorrente da aplicação	3.0
Apresentação apropriada de informações na console	3.0
Documentação	5.0

Observe, contudo, que alguns itens de avaliação estão entrelaçados entre si, de forma que, dependendo das circunstâncias, a perda de pontos em um item pode acarretar a perda automática em outros. Por exemplo, falhas na leitura de mensagens pelo servidor de *sockets* podem acarretar falhas na escrita de itens pelo cliente OPC, e, neste caso, ambos os itens seriam penalizados.

ATENÇÃO! Caso sejam detectadas evidências de improbidade acadêmica na execução dos trabalhos, os mesmos receberão nota zero e serão encaminhados aos Colegiados de Cursos para as providências disciplinares cabíveis.

Referências

[1] Vicente de Paulo Amâncio, Carlos Cristiano Teixeira, Emerson Klippel & Pedro Faria, "Operação Automatizada de um Sistema de Carregamento de Vagões", 2012, https://abmproceedings.com.br/ptbr/article/operao-automatizada-de-um-sistema-decarregamento-de-vages

[2] Carlos Cristiano Carvalho Teixeira, Vicente de Paulo Amancio, Wallace Costa Carvalho, Wendelling Átila Correia de Andrade, Emerson Klippel, Pedro Henrique Librelon de Fari &; Herbert Ricardo Garcia Viana, 2015, "Assertividade de Peso em Carregamentos Volumétricos de Vagões: Uma Aplicação na Mineração de Ferro", 2015, https://abmproceedings.com.br/ptbr/article/assertividade-de-peso-em-carregamentos-volumetricos-de-vagoes-uma-aplicacao-na-mineracao-de-ferro

[3] Grupo C tecnologia, https://ctecnologia.com.br/produto/mineracao-3/