Departamento de Engenharia Eletrônica
SISTEMAS DISTRIBUÍDOS PARA AUTOMAÇÃO – 2016/1

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

Descrição:

Uma indústria de cimento dispõe de um ERP (*Enterprise Resource Planning*) para lidar com seus procedimentos corporativos. Uma das funções deste ERP é processar as ordens de compra dos clientes e preparar o *setup* da fábrica em função do tipo de cimento encomendado. As informações de cada *setup* de produção são impressas em papel e digitadas pelo operador da sala de controle em um sistema supervisório, que por sua vez as envia para um Controlador Lógico Programável (CLP). Este procedimento manual é lento e sujeito a erros de digitação, o que provoca paradas operacionais indesejadas. Com o objetivo de aumentar a eficiência da produção, esta indústria deseja desenvolver uma aplicação de software que transfira automaticamente as informações de cada *setup* de produção diretamente para o CLP, sem a intermediação dos empregados. O ERP executa em um servidor IBM com sistema operacional AIX, e o acesso ao CLP é executado através de um servidor OPC "clássico" instalado em um computador com S.O. Windows.

Tendo em vista os diferentes S.O. envolvidos, a indústria contratou uma empresa de engenharia para desenvolver esta aplicação de software. Tal empresa projetou a aplicação de software na forma de uma arquitetura composta de um "cliente OPC" e de um "servidor de *sockets*". A aplicação executará no mesmo computador em que reside o servidor OPC. O ERP agirá então como um "cliente de *sockets*", enviando periodicamente mensagens de *setup* de produção a esta aplicação de software que, por sua vez, enviará estes dados ao CLP por meio de seu módulo "cliente OPC". Em adição, o ERP pode solicitar ainda dados de processo sobre a ordem de produção atualmente em fabricação. A figura a seguir exemplifica a estrutura desta aplicação.

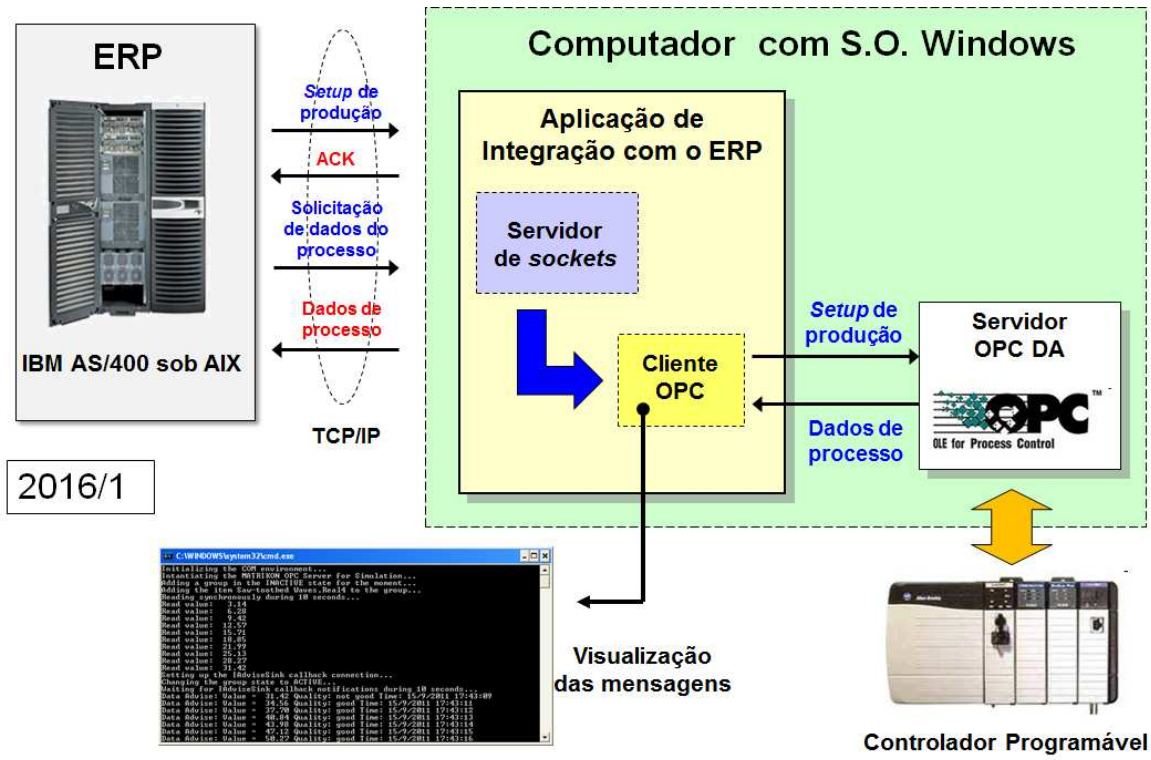

Figura 1 – Arquitetura do sistema

Seu trabalho consiste do projeto e desenvolvimento desta aplicação de software. A aplicação deve desenvolvida no ambiente *Microsoft Visual Studio C++* apenas em suas versões 2008 ou 2010. 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* deverá aguardar mensagens do ERP que correspondem a dois tipos: (1) setup de produção e (2) solicitação de envio de dados de produção. No primeiro caso, o servidor de *sockets* responde com uma mensagem de confirmação (*acknowledge,* ou ACK); no segundo caso, responde com uma mensagem contendo os dados solicitados. Todas as mensagens são estruturadas como cadeias de caracteres compostas de campos fixos de oito (8) caracteres ASCII, separados pelo delimitador "|" (barra vertical), como detalhado a seguir.

   - Mensagem de *setup* de produção (ERP → aplicação de software):

Campo	Tipo	Item de simulação no *Matrikon OPC Simulation Server*
1. Código da mensagem	Inteiro (sempre "00000001")	(não aplicável)
2. Número sequencial da mensagem	Inteiro	(não aplicável)
3. Tipo de cimento a produzir	Inteiro (1 a 10)	*Bucket Brigade.Int2*
4. Tonelagem a produzir	Inteiro (1 a 100)	*Bucket Brigade.Int4*
5. Hora prevista de início da produção	Cadeia de caracteres ("HH:MM:SS")	*Bucket Brigade.String*

   Exemplo: "`00000001|00000132|00000005|00000010|07:45:00`"

   - Mensagem de confirmação (ACK) (aplicação de software → ERP):

Campo	Tipo
1. Código da mensagem	Inteiro (sempre "00000002")
2. Número sequencial da mensagem	Inteiro

   Exemplo: "`00000002|00000084`"

   - Mensagem de solicitação de dados de produção (ERP → aplicação de software):

Campo	Tipo
1. Código da mensagem	Inteiro (sempre "00000005")
2. Número sequencial da mensagem	Inteiro

   Exemplo: "`00000005|00000019`"

   - Mensagem de dados de produção (aplicação de software → ERP):

Campo	Tipo	Item de simulação no *Matrikon OPC Simulation Server*
1. Código da mensagem	Inteiro (sempre "00000010")	(não aplicável)
2. Número sequencial da mensagem	Inteiro	(não aplicável)
3. Produção acumulada até o momento (t)	Inteiro	*Random.Int2*
5. OEE[1]computado no momento	Real	*Random.Real4*
6. Hora prevista de término da produção	Real	*Random.Time*

   Exemplo: "`00000010|00045897|00000005|14.56780|22:30:00`"

   Em todas as mensagens, os números sequenciais de mensagens são individuais para cada tipo de mensagem, devendo ser continuamente incrementados a cada mensagem trocada e recomeçando de zero quando a contagem máxima for alcançada. A porta TCP a ser usada na comunicação via *sockets* deverá ser a **4660**. Todas as mensagens que trafegam entre o ERP e a aplicação de software deverão ser exibidas na janela de console (ou, opcionalmente, interface gráfica) associada à aplicação.

   As mensagens de *setup* da produção serão disparadas arbitrariamente mediante o acionamento da tecla "s" pelo usuário, na aplicação que simulará o "cliente de *sockets*" (ou seja, o sistema ERP). Por outro lado, este disparará mensagens de solicitação de dados regularmente a cada 2 segundos.

[1] OEE (*Overall Equipment Efficiency*): Trata-se de um dos principais indicadores de desempenho (*Key Performance Indicator*, ou KPI) das indústrias modernas. Pesquise no Google a respeito!

Caso haja perda de comunicação entre o sistema ERP 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 solicitação de dados, de forma que as mesmas sempre estejam com valores atualizados a serem enviados ao ERP. A taxa de atualização das leituras deverá ser de 500 ms.

   b) Escrever no servidor OPC, de forma síncrona, os valores recebidos na mensagem de *setup* de produção enviada pelo sistema ERP.

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

3. Interface Gráfica (opcional)

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 do módulo "servidor de *sockets*" deverá ser testado através de programa específico a ser fornecido pelo professor, que simula o funcionamento do sistema ERP (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 cliente 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.
2. Para desenvolver os programas do trabalho, deve ser utilizada **somente** a ferramenta *Microsoft Visual C++ 2008* ou *2010* (versão completa ou a versão *Express Edition*[2]). Atenção: se você utilizar uma versão mais recente do *Visual Studio*, certifique-se que a solução gerada seja compatível com uma das versões acima, pois será com essas que o professor testará a aplicação. Eventuais incompatibilidades de versões serão tratadas como trabalhos não-entregues.
3. 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 entregues por email, com o seguinte conteúdo:
   - Documentação do trabalho, na forma de arquivo-texto, documento Word ou PDF contendo:
     - Indicação precisa do **nome e sobrenome** dos autores;
     - Versão empregada do *Visual Studio C++*;
     - 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 poderá interferir na 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. (*DICA: Você pode remover os arquivos de extensão .ipch e .sdf de sua solução, para diminuir o tamanho do arquivo ZIP ou RAR. Os mesmos serão automaticamente re-gerados 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

[2] A versão *Express Edition* do Visual C++ 2010 ainda pode ser obtida da Microsoft: http://go.microsoft.com/?linkid=9709969

necessário à execução do aplicativo. Programas que não executem, parcial ou integralmente, devido à necessidade destes componentes serão considerados não-entregues.

4. **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.
5. Envie seu trabalho apenas para o email `luizt.smendes@gmail.com`
6. Data de entrega: até **23h59min** do dia **31/05/2016**. Será usada a data/hora de envio do email para fins de certificação. Não serão consideradas exceções a este prazo, sob qualquer pretexto (problema com provedor de Internet, emails recusados por filtros de *spam*, etc.). Trabalhos enviados após esta data, até **07/06/2016**, terão pontuação reduzida em 50%. Trabalhos enviados após 07/06/2016 serão desconsiderados.
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 ou eventuais inconsistências deste enunciado deverão ser submetidas ao professor preferencialmente por email. Tais dúvidas e respectivas respostas serão publicadas na página *web* da disciplina, de forma que todos os alunos possam inteirar-se das mesmas.

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.