UNIVERSIDADE NOVA DE LISBOA
FCT - Departamento de Engenharia Electrotécnica
Secção de Robótica e Manufatura Integrada

Integração de Sistemas

2016 / 2017

Trabalho 1

Programação de DLL's e Integração com JAVA

Utilizando Sockets

Duração: 3 aulas acompanhadas por docente

Entrega: 2 de Abril

1. INTRODUÇÃO

É frequente em integração de sistemas a interligação de aplicações escritas em linguagens distintas. Um dos problemas tradicionais é a integração de bibliotecas já existentes na aplicação que se está a desenvolver.
Assim, neste trabalho estudar-se-ão técnicas de implementação de Dynamic Link Libraries (DLL's) escritas em linguagem C e a sua invocação a partir de aplicações desenvolvidas em linguagem JAVA utilizando a interface JNI (JAVA Native Interface).

2. Apresentação do Problema

Monitorização de Dispositivos de Geração de Energia

Nos últimos tempos tem crescido a utilização de dispositivos para geração de energia, tanto pelas empresas como por particulares. Estes dispositivos permitem gerar energia de forma fácil e sem grande manutenção. Embora a geração de energia e o seu consumo seja fácil, é difícil perceber a rentabilidade destes dispositivos e se estão a operar de forma correta com a mesma facilidade. Cada um dos fabricantes disponibiliza uma interface que permite consultar o desempenho de cada elemento e assim é necessário ao utilizador "saltar" de dispositivo em dispositivo de forma a consultar o desempenho do dispositivo e se está operacional.

Neste trabalho é proposta uma nova abordagem genérica, em que são pedidas DLL's que permitem integrar qualquer dispositivo deste género com a linguagem de programação JAVA. Fazendo com que toda esta informação possa ser utilizada e processada num ambiente de mais alto nível, independentemente do fabricante.

Neste trabalho é pedido que se desenvolva uma DLL que emule a operação dos dispositivos. De forma a possibilitar uma abordagem ponto a ponto, em que o

utilizador consegue ler diretamente o estado do *hardware*. Uma arquitetura é proposta, baseada numa aplicação para o utilizador e em que do lado do *hardware*, é utilizada uma DLL para consultar os valores disponibilizados pelos dispositivos.

A comunicação entre a aplicação disponibilizada para o cliente e os dispositivos é possível utilizando *sockets* para o efeito.

Infraestrutura

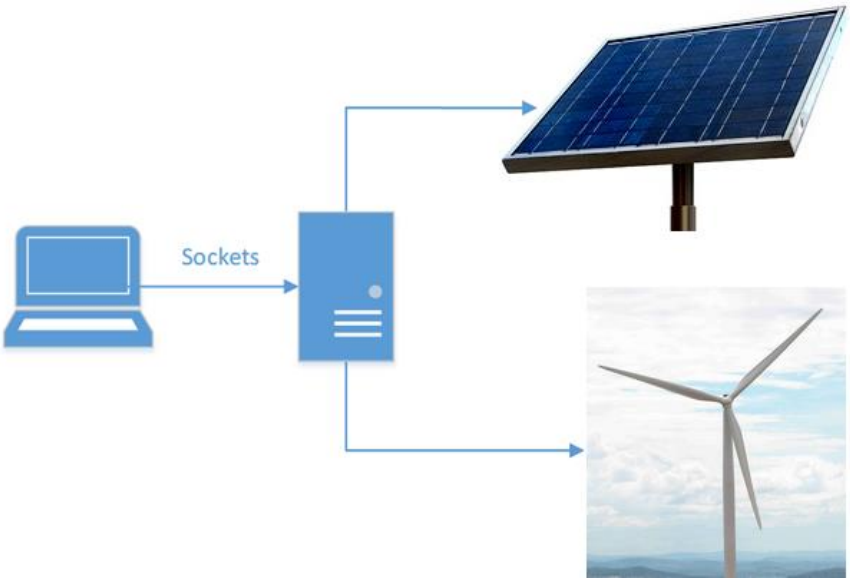

Figure 1- Infraestrutura

O corpo docente disponibiliza dois projetos, um para o cliente e um para correr na máquina que opera os dispositivos. No projeto do cliente é disponibilizada uma interface gráfica, bastante simples, que já consegue receber pedidos do utilizador e enviá-los através da comunicação por *socket* já implementada.

Do lado do computador responsável por ler os valores dos dispositivos e que se comporta como servidor nestas comunicações por *socket*, o código que permite a este se comportar como tal já está implementado.

O núcleo do trabalho a realizar encontra-se na implementação da DLL que permite operar os dispositivos e da biblioteca JNI que permite integrar o código JAVA do servidor com a mesma DLL.

A DLL, no mínimo, deverá ter a possibilidade de ler a energia produzida naquele momento, ver e mudar o estado do dispositivo (ligado / desligado) e consultar se existe algum problema com a execução.

A DLL deverá retornar um *float* no que diz respeito à energia produzida no momento, *int* no que diz respeito ao estado e uma *String* no restante método.

Métodos a Implementar

- float energyProduction();
- int turnOn(int state);
- int isOn();
- char* error();

3. Implementação

Na implementação pedida será utilizado o seguinte material:

- Linguagem JAVA no IDE Netbeans;
- Linguagem C no IDE Microsoft Visual Studio 2013 para Windows Vista ou superior;
- Código fornecido pelos docentes da cadeira;

4. Planeamento das Aulas

O código fornecido pelo corpo docente deverá ser consultado e estudado antes da primeira aula.

Aula 1 – Criação da DLL

1. Abra o Visual Studio e crie um novo projeto em C/C++ do tipo Win32 Console Application. Nas Application Settings marque Empty Project. Adicione um ficheiro main.c e defina as funções do ponto 2.1. Teste-as e garanta que estão todas a funcionar corretamente antes de avançar.
2. Crie um novo projeto em C/C++ do tipo Win32 Console Application. Em Application Settings, marque Empty Project nas Additional Options e seleccione DLL em Application Type.
3. Adicione um ficheiro ".c" e copie as funções criadas no ponto 1. Acrescente "__declspec(dllexport)" ao protótipo das funções.
4. Defina agora o ficheiro ".h". Não se esqueça de proteger o ficheiro com uma macro que não permite redefinições e mantenha o protótipo das funções igual ao descrito no ponto 3. Compile.
5. Crie outro projeto como fez no ponto 1 e introduza a função main mas para já deixe-a em branco.
6. Nas propriedades do projeto que acabou de criar faça:
    a. C/C++ -> General -> Adicional Include Directories – adicione o caminho para o ficheiro ".h" que implementou no projeto da DLL.

b. Linker ->General->Additional Library Directories – adicione o caminho para o ficheiro ".lib" criado no ponto 4 (costuma estar na pasta Debug da solução).
c. Linker ->Input->Additional Dependencies – adicione o nome do ficheiro ".lib" que está na pasta do ponto 6b.
d. Copie o ficheiro ".dll" gerado no primeiro projeto para a pasta Debug do novo projeto. Compile.

7. Inclua o ficheiro ".h" definido no ponto 4 no ficheiro main criado no ponto 5 e invoque as funções da DLL na função main. Teste e avalie os resultados.

Aula 2 – Invocação da DLL a partir de JAVA

1. Abra o Netbeans e crie um projeto do tipo JAVA class library. Adicione uma classe JAVA em branco.
2. Na classe declare os seguintes protótipos de função:
   a. native static float energyProduction();
   b. native static int turnOn(int state);
   c. native static int isOn();
   d. native static String error();
3. Compile o código e verifique que não existem erros.
4. Se ainda não o fez (noutras cadeiras) acrescente o caminho para a pasta Bin da distribuição de JAVA à variável de ambiente Path.
5. Abra a consola e digite javah se não obtiver nenhum erro então está tudo bem até aqui.
6. Na consola navegue até à pasta ...\build\classes do projeto criado no ponto 1.
7. Nessa pasta execute o comando javah –jni lib. O comando deverá ter gerado um ficheiro "lib.h". Inspecione o ficheiro.
8. Crie um novo projeto no Visual Studio como fez no ponto 2 da primeira aula.
9. Acrescente o ficheiro "lib.h" gerado em 7 e acrescente o nome das variáveis nos protótipos das funções. Implemente o ficheiro ".c" respetivo (deixe a segunda função por implementar para já) e compile.
10. Se correu tudo bem então o compilador deve dar o seguinte erro fatal error C1083: Cannot open include file: 'jni.h': No such file or directory.
11. Este erro quer dizer que o Visual Studio não sabe onde procurar o ficheiro "jni.h". Adicione o caminho para o ficheiro tal como fez no ponto 6a da primeira aula. Procure o ficheiro da pasta da distribuição de JAVA. Proceda de forma semelhante para outros ficheiros em falta (investigue a pasta C:\Program Files (x86)\Java\jdk1.7.0_01\include\win32). Neste momento já não deve ter erros de compilação.

12. O tipo jstring que está no protótipo da função não pode ser manipulado à custa das funções convencionais do C. É necessário recorrer as função nativas do JNI que efetuam essa conversão. Verifique o que faz a função GetStringUTFChars.
13. Existem outras funções do JNI fundamentais (identificadas abaixo) para a implementação da segunda função verifique para que servem e aplique-as criteriosamente na implementação da segunda função.
    a. ReleaseStringUTFChars
    b. NewStringUTF
14. Se a implementação de ambas as funções (ponto 2.1) estiverem corretas então a DLL está pronta a integrar com a aplicação em JAVA.

Aula 3 – Integração Final e Testes

- Volte ao programa em JAVA e acrescente a seguinte linha de código static{System.loadLibrary("lib");} e compile.
- Copie a DLL criada na aula anterior no ponto 14 para a pasta \dist do projecto em JAVA.
- Acrescente a função main à classe e teste a chamada aos métodos nativos.
- Aperfeiçoamento do Código:
    - No ponto 3.2 em vez de utilizar a DLL já existente implementou diretamente as funções nos ficheiros gerados pelo comando javah. Num cenário mais realista poderia não ter acesso ao código fonte da DLL mas apenas aos ficheiros “.lib” e “.h”. Neste caso a sua aplicação teria de ser construída a custa de uma DLL de integração JAVA/C que consumia os recursos da DLL já existente. Reveja a sua implementação do ponto 3.2 para refletir este cenário.

NOTA: Quando se faz clean and build o conteúdo da pasta \dist do projecto é apagado pelo que se torna necessário voltar a copiar para lá a dll. A função loadLibrary por vezes não funciona em alguns sistemas e pode ser substituída pela função load que recebe como argumento o caminho completo para a localização da DLL.

5. Avaliação

A avaliação do trabalho tem a seguinte ponderação:

- Correta implementação da DLL em C e testes à mesma:
  - 16 valores
- Correta integração da DLL em C com o programa em JAVA e testes:
  - 2 valores
- Correta implementação e demonstração de funcionalidades extra não definidas:
  - 2 valores.

Docentes

Pedro Monteiro pedro.monteiro@uninova.pt

Ricardo Peres ricardo.peres@uninova.pt

José Barata jab@uninova.pt