UNIVERSIDADE FEDERAL DO ABC - UFABC

Curso de Especialização em Tecnologias e Sistemas de Informação

Aplicações da Tecnologia Java para a Internet das Coisas

Marcelo dos Santos

São José dos Campos - SP

2015

Marcelo dos Santos

Aplicações da Tecnologia Java para a Internet das Coisas

Monografia apresentada ao Curso de Especialização da Universidade Federal do ABC, como requisito parcial à obtenção do título de Especialista em Tecnologias e Sistemas de Informação .

Área de concentração: Computação em Todo Lugar

Orientador: Prof. Dr. Jorge Tomioka

São José dos Campos - SP
Universidade Federal do ABC
2015

Universidade Aberta do Brasil - Universidade Federal do ABC
Programa de Pós-graduação em Tecnologias e Sistemas de Informação

Monografia intitulada "*Aplicações da Tecnologia Java para a Internet das Coisas*", de autoria de Marcelo dos Santos, aprovada pela banca examinadora constituída pelos professores:

Professor 1

Professor 2

Orientador

Prof. Dr. Jorge Tomioka

São José dos Campos - SP, 20 de Agosto de 2015

Dedico este trabalho

A Deus pela orientação, inspiração e conforto espiritual em cada momento da vida.

Aos meus pais e familiares por minha formação pessoal e apoio em minhas decisões.

Aos meus amigos e colegas pelo suporte nas dificuldades do dia a dia.

A todos que acreditam em meu potencial e ajudaram de alguma forma ou outra a tornar este caminho mais fácil de ser percorrido.

AGRADECIMENTOS

Agradeço ao Prof. Dr. Jorge Tomioka pela orientação e direcionamento nos estudos.

Agradeço a UFABC e a UAB pela oportunidade de participar desse processo de crescimento pessoal e profissional. Obrigado a todos do corpo docente e discente.

Quem de boa vontade carrega o difícil

Supera também o menos difícil

(Poema 26 - Lao-Tsé)

RESUMO

O conceito *Internet* das Coisas é o próximo ramo de estudo da computação, exigindo sistemas computacionais em objetos e sua conexão com a *Internet*. A abordagem da pesquisa traz a caracterização da *Internet* das Coisas e plataforma *Java Embedded* para diferentes escopos de aplicações, como interfaceamento de periféricos de baixo nível, arquitetura cliente/servidor e processamento de eventos complexos. A metodologia aplicada com a plataforma *Java Embedded* em aplicações de diferentes níveis de arquitetura demonstra o potencial da tecnologia. Pesquisar uma plataforma aplicável para diversos contextos, enriquece o desenvolvimento de novas tecnologias e traz novas possibilidades para o mercado, direcionando para futuros estudos em especificação e desenvolvimento de protocolos e sistemas operacionais especializados.

PALAVRAS-CHAVE: *Internet* das Coisas; *Java*; *Software* Embarcado.

ABSTRACT

The Internet of Things concept is the next computing branch of study, requiring computer systems on objects and their connection to the Internet. The research approach brings the characterization of the Internet of Things and Embedded Java platform for different scopes of applications such as low-level peripheral interfacing, client / server architecture and complex event processing. The methodology applied to Embedded Java platform applications of different levels of architecture demonstrates the potential of technology. Search an applicable platform for different contexts enriches the development of new technologies and brings new possibilities to the market, directing to future studies on specification and development of protocols and specialized operating systems.

KEYWORDS: Internet of Things; Java; Embedded Software.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A *Internet* das Coisas - *Internet of Things* (*IoT*) tem como objetivo prove a conexão de rede aos objetos e tornando-os inteligentes, em um sistema composto de um conjunto de três pilares, chamado de *Three Is*: Instrumentação, Interconexão e Inteligência (LAMPKIN WENG TAT LEONG, 2012).

A Instrumentação captura as informações através de sensores, enquanto a Interconexão provê o meio para a transferência da informação do ponto de coleta para o ponto de consumo, finalmente a Inteligência com a responsabilidade sobre o processamento e análise da informação para derivar o máximo de valor e conhecimento (LAMPKIN WENG TAT LEONG, 2012).

Os anos demonstram o crescimento do número de equipamentos conectados à *Internet*, em 2000 a quantidade de humanos na Terra foi de 6 bilhões para 500 milhões de equipamentos, enquanto em 2011 com 7 bilhões de humanos para 13 bilhões de equipamentos, para 2015 está previsto um número três vezes maior de equipamentos do que humanos na Terra (PRESSER, 2012).

Os objetos passam a sentir o contexto do ambiente e a comunicar com os humanos, tornando ferramentas poderosas para a obtenção de conhecimento.

Os exemplos de aplicações são inúmeros, tais como:

- Medição do consumo de energia e água, analisando os resultados em tempo real, assim avaliando os custos (gastos, créditos e demanda), a consciência do desperdício pelo comportamento do consumo, a eficiência das instalações (quando realizar melhorias e reparos) e a comparação estatística com outras fontes pela mídia social (PRESSER, 2012).
- Cuidado contínuo dos pacientes idosos e com doenças crônicas, permitindo o repouso em casa ao invés de esperarem em filas e quartos de hospitais e clínicas, em que os médicos monitoraram remotamente o estado de saúde dos pacientes por equipamentos nas roupas ou distribuídos pela casa e/ou ruas (PRESSER, 2012).
- Iluminação Urbana por regular o controle de luz com a presença de carros e pedestres, como também destacar situações adversas como vazamento de óleo, acidentes de trânsito e trabalhadores de rua (PRESSER, 2012).

As partes de interconexão e instrumentação são disponibilizadas em uma variedade de tecnologias como padrão de mercado, diferente da inteligência na qual requisita o desenvolvimento completo do *software*, desde a fase de requisitos, análise, desenvolvimento e testes. Dentre

as plataformas disponíveis para o desenvolvimento estão o *Java* e o *C/C++*, juntamente com suas linguagens de *front-end* como o *Python*. Dos sistemas operacionais estão as distribuições *Linux* como *Debian* e o *Ångström*, sendo o último especializado para dispositivos embarcados.

A plataforma selecionada para o estudo é o *Java Embedded*, em que apresenta compatibilidade com diversas arquiteturas de *hardware* através de sua Máquina Virtual - *Virtual Machine* (*VM*) e disponibiliza uma variedade de Interface de Programação de Aplicação - *Application Programming Interface* (*API*) para periféricos de baixo nível, arquitetura Cliente/Servidor - *Client/Server* e Processamento de Eventos Complexos - *Complex Event Processing* (*CEP*).

A metodologia de trabalho apresentará as tecnologias disponíveis na plataforma *Java Embedded* e demonstrará exemplos de aplicações com a utilização da plataforma de prototipagem *Raspberry PI B+*.

1.1 Justificativa

O conceito *IoT* é a próxima evolução da computação, principalmente pela preocupação com os recursos ambientais, médicos e planejamento urbano através do processo de monitoração, aquisição e controle dos dados. O mercado disponibiliza uma ampla variedade de *hardware* para o sensoriamento e processamento, facilitando a atividade de construir um sistema computacional. O *software* é o diferencial para a pesquisa e o desenvolvimento de novas tecnologias, conhecer a plataforma embarcada do *Java* amplia essas qualidades e direciona para as futuras oportunidades na *IoT*.

1.2 Objetivos

1.2.1 Geral

Elaborar um estudo sobre as características e aplicações da plataforma *Java Embedded* para o domínio de *software* embarcado, focado no conceito *IoT*.

1.2.2 Específicos

- Especificar as características do conceito *IoT*.
- Especificar as tecnologias *Java* disponíveis para o desenvolvimento no domínio de *software* embarcado.
- Apresentar as interfaces para a comunicação de periféricos de baixo nível.
- Apresentar a aplicação da arquitetura Cliente/Servidor em embarcados.
- Apresentar a aplicação de Processamento de Eventos Complexos em embarcados.

1.3 Metodologia

Revisão bibliográfica: será feito uma pesquisa sobre o tema do projeto, procurando informações sobre as tecnologias da plataforma *Java Embedded* e a conceitualização da computação ubíqua para o desenvolvimento de aplicações para o conceito *IoT*.

Caracterização da *Internet* das Coisas: descrever as características da computação ubíqua (combinação da computação móvel com a computação embarcada) presentes no conceito *IoT*.

Caracterização do *Java Embedded*: descrever as características, vantagens e desvantagens das tecnologias da plataforma *Java Embedded* para o desenvolvimento no domínio de *software* embarcado, utilizando a plataforma de prototipagem *Raspberry PI B+*.

Primeiro exemplo de aplicação: desenvolver um exemplo de aplicação *Java ME* (Edição Micro - *Micro Edition*) com a *Device I/O API* (Entrada/Saída - *Input/Output*) (Interface de Programação de Aplicação - *Application Programming Interface*), explanando o interfaceamento de periféricos de baixo nível presentes em sistemas embarcados.

Segundo exemplo de aplicação: desenvolver um exemplo de aplicação *Java Embedded Suite* (*JES*), explanando o uso da arquitetura Cliente/Servidor em sistemas embarcados.

Terceiro exemplo de aplicação: desenvolver um exemplo de aplicação *Java Oracle Event Processing* (*OEP*), explanando a utilização de Processamento de Eventos Complexos em sistemas embarcados.

Discussão dos resultados: apresentar os resultados obtidos pelos exemplos aplicados, evidenciando as principais vantagens e desvantagem observadas.

2 EVOLUÇÃO DA COMPUTAÇÃO

A computação passou por diversas evoluções no último século, devido as pesquisas e desenvolvimento em diferentes áreas de estudo, seus resultados veem através de novas tecnologias e aplicações, como o objeto de estudo: a aplicação *Internet* das Coisas - *Internet of Things* (*IoT*) e a tecnologia *Java Embedded*.

2.1 Paradigmas Computacionais

Os paradigmas computacionais delimitam as principais evoluções da computação, sendo os principais marcos:

- sistemas *mainframe*: na década de 60, consistindo de grandes computadores do tamanho de salas inteiras em departamentos de processamento de dados. Suas principais características estão: a grande capacidade de Entrada/Saída - *Input/Output* (*I/O*), o relacionamento de uma máquina para um usuário e o processamento em lote (*batch*);
- sistemas por terminais (*terminals*): na década de 70, consistindo de dispositivos remotos conectados ao computador central (*mainframe*) via rede telefônica para a entrada e saída de dados aos usuários. Suas principais características estão: o processamento interativo e multitarefa, o relacionamento de uma máquina para muitos usuários e o compartilhamento de tempo de processamento (*time sharing*);
- computadores pessoais (*desktop*): na década de 80, consistindo de pequenos computadores com preço acessível para uso em pequenas aplicações como escritório até grandes aplicações como jogos, sendo estendido para aplicações em ambientes de sistemas embarcados. Suas principais características estão: processamento local e interativo e o relacionamento de uma máquina para um usuário;
- redes de computadores (*network*): na década de 90, consistindo de interconexão de computadores com as vantagens de comunicação, compartilhamento de recursos e acesso não local. Sua principais características estão: a implementação do modelo de arquitetura Cliente/Servidor - *Client/Server* e a comunicação centralizada; e
- sistemas distribuídos (*distributed systems*): a partir da década de 90, consistindo de um conjunto de computadores compartilhando o processamento para a aplicação, aparentando com um único computador ao usuário. Suas principais características estão: a implementação do modelo de arquitetura Distribuído, a comunicação descentralizada

e as aplicações por *Grid* e *Cloud*.

O próximo candidato de paradigma computacional é a computação ubíqua, uma união das pesquisas e desenvolvimento entre computação móvel e computação pervasiva.

A computação móvel traz aos usuários a informação contínua, independente de sua localização física, enquanto a computação pervasiva traz o processamento autônomo para o usuário em diversos dispositivos, adicionando inteligência ao mesmo.

2.2 Computação Ubíqua

A computação ubíqua tem origem nos trabalhos de *Mark Weiser* da *Xerox Palo Alto Research Center* (*PARC*) durante o início da década de 90, com a visão da criação de ambientes inteligentes através de sistemas embarcados (computação pervasiva) aplicados aos objetos do dia a dia, trocando informações entre si e pela rede de computadores. Segundo *Weiser* (WEISER, 1991, p. 94): "Especializados elementos de *hardware* e *software*, conectados por fios, ondas de rádio e infravermelho, serão tão ubíquos que ninguém irá notar sua presença."

Nesse ambiente o usuário estaria cercado por diversos dispositivos realizando computação imperceptivelmente, sem uma interface de usuário como em computadores pessoais (menos intrusão). Esses computadores operam autonomamente e realizam tomadas de decisões e interações inteligentes, conforme o contexto do ambiente.

A computação ubíqua traz aos desenvolvedores um alto grau de mobilidade e imersão computacional. O grau de mobilidade refere-se a capacidade de locomoção física dos serviços computacionais, enquanto o grau de imersão computacional refere-se a capacidade de inteligencia do dispositivo em detectar, explorar e construir de maneira dinâmica o contexto do ambiente.

2.3 Internet das Coisas

O termo *Internet* das Coisas ou *Internet of Things* (*IoT*) surgiu em 1999 pelo pesquisador britânico *Kevin Ashton* do Instituto de Tecnologia de *Massachusetts - Massachusetts Institute of Technology* (*MIT*) durante uma apresentação executiva.

A ideia era utilizar as tecnologias de Identificação por Rádio Frequência - *Radio-Frequency IDentification* (*RFID*) e sensores para habilitarem os computadores para observar, identificar e compreender o mundo, sem a limitação dos dados inseridos por usuários (ASHTON, 2009).

A definição do *Cluster* Europeu de Pesquisa da *Internet* das Coisas - *European Research Cluster on the Internet of Things* - (*IERC*) (SMITH, 2012, p. 26) para *IoT* é:

> Infraestrutura de rede dinâmica global com capacidades de autoconfiguração com base em protocolos de comunicação padronizados e interoperáveis onde as coisas físicas e virtuais têm identidades, atributos físicos, personalidades virtuais, utilizam interfaces inteligentes e estão perfeitamente integrados na rede de informação.

Hoje o termo *IoT* é utilizado para relacionar comercialmente a Computação Ubíqua.

2.4 A Tecnologia Java

A tecnologia *Java* foi desenvolvida por um grupo de engenheiros da *Sun* (hoje adquerida pela *Oracle*) conhecidos por "*Green Team*" em 1991, com o propósito de contribuir para os dispositivos digitais para serem inteligentes, funcional e de um meio mais divertido.

Liderado por *James Gosling*, o time desenvolveu a linguagem de programação *Java* e demonstrou através de um dispositivo portátil para a indústria de televisão a cabo, como também incorporou ao navegador de *Internet Netscape Navigator* em 1995, após a explosão da *WEB*.

Hoje a tecnologia *Java* possui um amplo de escopo de aplicações, sendo de *chips* Sistema Global para Comunicações Móveis - *Global System for Mobile Communications* (*GSM*) até grandes servidores empresariais; e com interesses na *IoT*, a *Oracle* lançou a plataforma *Java Embedded*.

3 CARACTERIZAÇÃO DA INTERNET DAS COISAS

O primeiro passo para definir as ferramentas para o desenvolvimento de *software* para a *Internet* das Coisas - *Internet of Things* (*IoT*) é entender a formação de seus componentes e plataforma, empregadas para a Inteligência. Assim visualizando a implementação em plataformas de desenvolvimento, como o *Raspberry PI B+*.

3.1 Componentes

A *IoT* engloba os seguintes componentes: Interconexão, Instrumentação e Inteligência, conforme a representação do conceito Planeta Inteligente - *Smarter Planet* da *International Business Machines* (*IBM*) na figura 1.

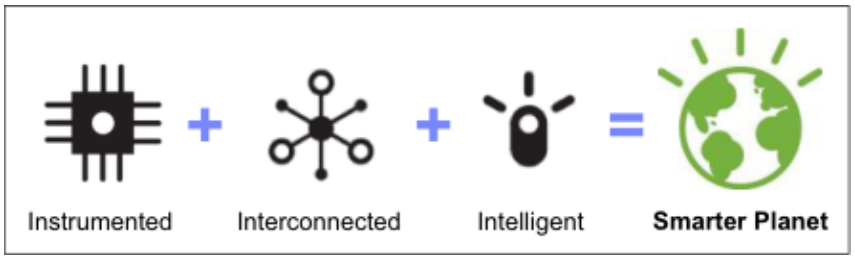

Figura 1: Os Três Pilares de um Planeta Inteligente

A interconexão é construída pela conectividade com as redes locais e a *Internet* através de várias conexões física, tais como *Bluetooth*, *Ethernet* e Fidelidade Sem Fio - *Wireless Fidelity* (*Wi-Fi*).

A instrumentação é construída pelos sensores e atuadores, no papel de interfacear com o ambiente em indicar, monitorar e controlar os fenômenos de diversas natureza. No processo de indicação estão os Diodos Emissores de Luz - *Light Emitting Diodes* (*LEDs*) e os *Displays* Alfanumérico ou Gráfico. No processo de monitoração estão os sensores eletrônicos para temperatura, umidade, presença, dentre outros... E no processo de controle estão os motores e atuadores. Em uma constituição mais completa de instrumentação estão a Identificação por Rádio Frequência - *Radio-Frequency IDentification* (*RFID*), Sistema de Posicionamento Global - *Global Positioning System* (*GPS*) e Sistema de Navegação Inercial - *Inertial Navigation System* (*INS*).

A inteligência é construída pelos algoritmos em *software* embarcado, em que realiza a junção do sistema computacional pelo processamento, análise e ação da informação em conhecimento.

3.2 Raspberry PI

O *Raspberry PI* é uma plataforma de desenvolvimento destinada para a educação, desenvolvido por *Even Upton* em 2006 da *Raspberry PI Foundation*. A figura 2 apresenta a versão *Raspberry PI B+* utilizada no objeto de estudo, um sistema baseado em *Linux* utilizando a arquitetura de Máquina *RISC* Avançada - *Advanced RISC Machine ARM*) de baixo custo (por U$ 40) do tamanho de uma cartão de crédito.

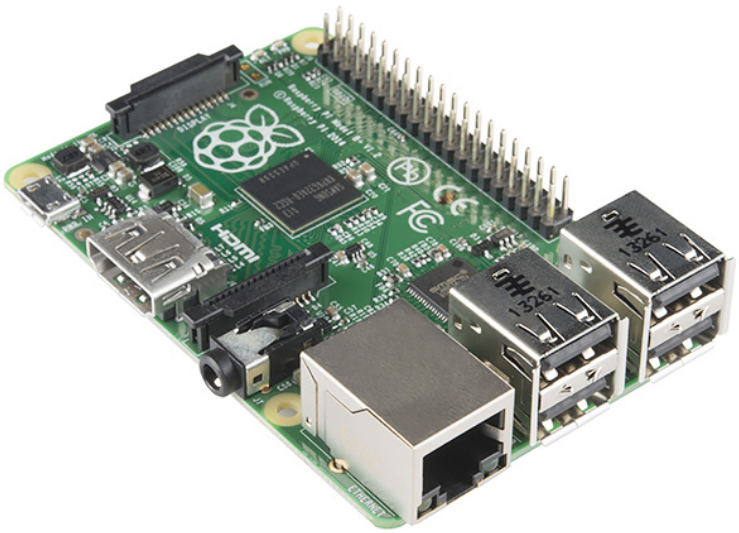

Figura 2: Raspberry PI B+

O ecossistema do *Raspberry PI* possui uma grande comunidade na *WEB*, diversas opções de *hardware* periféricos, variações de sistema operacionais *Linux* como especializações para Tempo Real - *Real Time* (*RT*) e Computador com um Conjunto Reduzido de Instruções - *Reduced Instruction Set Computer* (*RISC*).

Existem outros sistemas para uso na *Internet* das Coisas, tais como *BeagleBone*, *Cubieboard*, *Odroid* dentre outros... A maioria baseada no mesmo sistema computacional do *Raspberry PI*, com sistema *Linux* em arquitetura *ARM*.

4 CARACTERIZAÇÃO DO JAVA EMBEDDED

O *Java Embedded* oferece aos desenvolvedores o estado da arte de uma plataforma otimizada para um espectro de dispositivos embarcados, envolvendo bibliotecas e ferramentas para o melhor aproveitamento dos Sistema-em-um-chip - *System-on-a-chip* (*SoC*) através de um conjunto de plataformas, conforme o *footprint* da aplicação a ser desenvolvida.

A plataforma possui componentes 100% em código *Java*, desenvolvendo aplicações embarcadas mais simples do que em plataforma *C*, integrando ferramentas de desenvolvimento, tais como *NetBeans* e *Eclipse*. Os *bytecodes* compilados para *desktop* são identicos aos executados no embarcado, com exceção da velocidade de execução, permitindo as etapas de desevolvimento como codificação, execução, *debug* e *profile* sem a necessidade do dispositivo embarcado.

4.1 Plataforma Java Embedded

As plataformas abordadas no respectivo trabalho serão: *Java ME Embedded*, *Java Embedded Suite* (*JES*) e *Java Oracle Event Processing* (*OEP*).

4.1.1 Java ME Embedded

O *Java ME Embedded* é uma versão reestruturada da plataforma *Java ME* com a inclusão da *Device I/O API* (Entrada/Saída - *Input/Output*) (Interface de Programação de Aplicação - *Application Programming Interface*), destinadas aos periféricos de baixo nível encontrados nos dispositivos embarcados, tais como Entrada/Saída de Propósito Geral - *General Purpose Input/Output* (*GPIO*), Conversor Analógico para Digital - *Analog to Digital Converter* (*ADC*), Circuito Inter-Integrado - *Inter-Integrated Circuit* (*I2C*), Interface Periférica Serial - *Serial Peripheral Interface* (*SPI*), Transmissor/Receptor Assíncrono Universal - *Universal Asynchronous Receiver/Transmitter* (*UART*) e etc.

A plataforma *Java ME Embedded* consiste de duas versões:

- *Java ME Embedded*: dedicado para as aplicações sempre ativa (*always-on*), sem interface gráfica com o usuário (*headless*) e com conexões a dispositivos. Possui *footprint* menor que 1 *megabyte* de memória;
- *Java ME Embedded Client*: dedicado para as aplicações com maior necessidade de características da plataforma *Java*. Possui *footprint* menor que 10 *megabytes* de memória.

Exemplos de aplicações:

- prototipagem para aplicações de computação embarcada;
- possibilidades para utilização de sensores como fonte de dados em aplicações para o conceito *Internet* das Coisas - *Internet of Things* (*IoT*).

4.1.2 Java Embedded Suite

O *JES* é uma versão da plataforma *Java EE* (Edição Empresarial - *Enterprise Edition*) para os sistemas embarcados, trazendo a capacidade da arquitetura Cliente/Servidor - *Client/Server* através de banco de dados - *databases*, serviços *Web* - *Web services* e aplicações *Servlet*.

Os componentes disponíveis para a plataforma são:

- *GlassFish*: versão reduzida do servidor de aplicações *GlassFish*, com suporte apenas para os componentes *Servlet 3.0* e *Bean Validation 1.0*;
- *Java DB*: versão otimizada do banco de dados *Derby* para uso embarcado (não possui características para arquitetura Cliente/Servidor), acessado via Linguagem de Consulta Estruturada - *Structured Query Language* (*SQL*) sobre o Conectividade de Banco de Dados *Java Java Database Connectivity* (*JDBC*);
- *Jersey*: *framework* para serviços *RESTful Web Services*, conforme a especificação *JAX-RS* (*JSR 311*).

4.1.3 Java Oracle Event Processing

O *OEP* é uma versão do sistema de Processamento de Eventos Complexos - *Complex Event Processing* (*CEP*) para os sistemas embarcados, trazendo a capacidade de transformar dados em informação para os dispositivos.

4.2 Vantagens e Desvantagens

A plataforma *Java Embedded* possui vantagens e desvantagens em relação a plataforma nativa (*C*).

4.2.1 Vantagens

- Suporte a aplicações *headless*: serviços executados em segundo plano;
- Segurança *sandbox*: a máquina virtual constrói um ambiente de execução para a aplicação, diferentemente da plataforma nativa em que não possui a característica;
- Múltiplos processos: capacidade para execução multitarefa de aplicações;
- Comunidade: a tecnologia *Java* possui uma grande comunidade de desenvolvedores em todo mundo, ideal para o aprendizado e suporte, na plataforma nativa a comunidade é fragmentada;
- Escalabilidade: excelente capacidade de gerenciamento de recursos e adição de novas funcionalidades a aplicação;
- Capacidade de atualização: a execução em máquina virtal traz para o sistema possui um excelente mecanismo para atualização da aplicação, na plataforma nativa o sistema é inferior em dificuldade.

4.2.2 Desvantagens

A principal desvantagem em relação a plataforma nativa está no desempenho, pois a máquina virtual utiliza de recursos como memória e processamento, além dos requeridos para a aplicação.

Uma abordagem para minimizar os problemas está em definir *profiles* para o tipo de aplicação, assim reduzindo a carga da máquina virtual.

4.3 Otimização da JVM

A Máquina Virtual Java - *Java Virtual Machine* (*JVM*) é otimizada para as Unidades Central de Processamento - *Central Processing Units* (*CPU*) da arquitetura (*ARM*), com suporte a *hard-float* no *Kit* de Desenvolvimento *Java* - *Java Development Kit* (*JDK*) 1.8. Várias plataformas de *hardware* são compatíveis como: *ARM*, *Intel*, *Atmel*, dentro outros.

5 METODOLOGIA

A metodologia adotada para o objeto de estudo foca na observação da tecnologia *Java* para o desenvolvimento no conceito *Internet* das Coisas - *Internet of Things* (*IoT*) através de três diferentes escopos de aplicações. A experimentação das aplicações proporcionará um amplo alcance de possibilidades, por meio das bibliotecas e ferramentas de *software*.

Os escopos selecionados para a experimentação são:

- periféricos de baixo nível: apresentar as Interfaces de Programação de Aplicação - *Application Programming Interfaces* (*APIs*) e periféricos destinados a comunicação e interfaceamento de baixo nível com o *Raspberry PI*. A aplicação utilizará o *Java ME* (Edição Micro - *Micro Edition*) com a *Device I/O API* (Entrada/Saída - *Input/Output*);
- arquitetura cliente/servidor: apresentar a utilização de servidores no *Raspberry PI*. A aplicação utilizará o *Java Embedded Suite* (*JES*);
- processamento de eventos complexos: apresentar o ambiente para desenvolvimento de Processamento de Eventos Complexos - *Complex Event Processing* (*CEP*). A aplicação utilizará o *Java Oracle Event Processing* (*OEP*).

As aplicações serão desenvolvidas pelo Ambiente de Desenvolvimento Integrado - *Integrated Development Environment* (*IDE*) *Netbeans* 8.0.2 com o *Kit* de Desenvolvimento de *Software* - *Software Development Kit* (*SDK*) *Java* 8. O sistema base é composto de um ambiente virtualizado no *Oracle VM VirtualBox* com o sistema operacional *Linux Debian*. No caso do *Java ME SDK 8 Embedded* será utilizado através do sistema operacional *Windows* 7, pois não há uma versão para o sistema operacional *Linux*.

As aplicações serão testadas com a plataforma de prototipagem *Raspberry PI B+* com o sistema operacional *Raspbian - Debian Wheezy* versão de 16/02/2015.

A comunicação do computador de desenvolvimento e a plataforma de prototipagem será pelo acesso remoto com os utilitários *Secure Shell* (*SSH*) e *Virtual Network Computing* (*VNC*) por rede *Ethernet*.

Os resultados serão apresentados através da indicação das principais vantagens e desvantagem observadas, dentro do conceito *IoT*.

6 JAVA ME EMBEDDED

Este capítulo tem como objetivo demonstrar a plataforma *Java Embedded - Java ME Embedded*.

6.1 Instalação

A plataforma de prototipagem deve estar operacional com o sistema operacional *Linux* e a plataforma *Java 8*, no caso de estudo a plataforma *Raspberry PI B+*. A figura 3 apresenta as configurações do sistema em estudo.

Figura 3: Configuração do Sistema

Para a plataforma de prototipagem é necessário a instalação do pacote *Oracle Java ME Embedded 8.1 for Raspberry Pi Model B (ARM11/Linux)*, necessário ter uma conta no site da *Oracle* para realizar o *download*.

Transfira o arquivo `oracle-jmee-8-1-rr-raspberrypi-linux-bin.zip` do sistema operacional principal para o dispositivo remoto, basta utilizar o comando *SCP*:

```
scp </path/from/file> <user>@<address>:/path/to/destination
```

Descompacte o arquivo, basta utilizar o comando *UNZIP*:

```
unzip <file>
```

Aplique a permissão 777 para os diretórios *appdb* e *bin*, basta utilizar o comando *CHMOD*:

```
chmod -R 777 appdb bin
```

6.1.1 Pacote Oracle Java ME Embedded ZIP

O pacote *Oracle Java ME Embedded ZIP* consiste de cinco diretórios:

- /appdb: este diretório é utilizado pela plataforma de prototipagem e contém as bibliotecas interna do *Java*;
- /bin: este diretório é utilizado pela plataforma de prototipagem para instalar, executar e remover as aplicações *IMlets*, possui os executáveis, *scripts* e o arquivo de configuração do *Java* `jwc_properties.ini`;

- /legal: este diretório contém a documentação de licença;
- /lib: este diretório é utilizado no processo de compilação dos *IMlets*, contém os arquivos necessários para a execução do *Developer Agent*;
- /util: este diretório possui o arquivo `proxy.jar`, requerido para a comunicação entre plataforma de prototipagem e *Desktop* através do *Developer Agent*.

No *NetBeans IDE 8* ative o *Java ME*, instale os arquivos *Oracle Java ME SDK 8.1* e *plugin Oracle Java ME SDK 8.1 Plugins for NetBeans 8*, necessário ter uma conta no site da *Oracle* para realizar o *download*. O *Oracle Java ME SDK 8.1* somente possui versão para o sistema operacional *Windows x86/64*.

6.1.2 Desenvolvimento

O processo de desenvolvimento inicia pela codificação da aplicação, em seguida seu aprimoramento pela documentação da Interfaces de Programação de Aplicação - *Application Programming Interfaces* (*APIs*), quando concluído é exercitado pelo emulador e finalmente a implantação no dispositivo.

O exemplo aplicado trata-se de um gerador de Modulação por Largura de Pulso - *Pulse-Width Modulation* (*PWM*) para a produção de sinais para fins como transmitir informações em baixo nível para outros dispositivos.

6.1.2.1 Codificação

A aplicação consiste de um *software IMlet*, uma classe na qual estende `javax.microedition.midlet.MIDlet` e implementa dois métodos:

- `startApp()`: sinaliza a aplicação a transição para o estado Ativo.
- `destroyApp()`: sinaliza a aplicação a transição para o estado Destruído.

O código final é apresentado abaixo:

```
// JavaMEExample.java
package ufabc;

import java.io.IOException;
import javax.microedition.midlet.MIDlet;
import jdk.dio.DeviceManager;
import jdk.dio.pwm.PWMChannel;

public class JavaMEExample extends MIDlet {

    PWMChannel channel;

    @Override
    public void startApp() {
        System.out.println("Início da Aplicação");
        try {
            channel = (PWMChannel) DeviceManager.open(18);
            channel.setPulsePeriod(1000000);
            channel.generate(500000, 10);
        } catch (IOException ex) {
        }
    }

    @Override
    public void destroyApp(boolean unconditional) {
        try {
            channel.close();
        } catch (IOException ex) {
        }
        System.out.println("Encerramento da Aplicação");
    }
}
```

O acesso aos periféricos são realizados através da permissão da *API* pela requisição do `jdk.dio.DeviceMgmtPermission` em Descritor de Aplicações - *Application Descriptor*, a figura 4 apresenta as permissões para o exemplo acima.

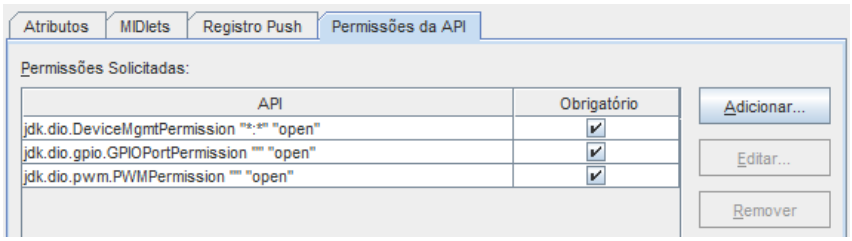

Figura 4: Descritor de Aplicações

6.1.2.2 Device I/O API

A *Device I/O API* (Entrada/Saída - *Input/Output*) (Interface de Programação de Aplicação - *Application Programming Interface*) provê uma biblioteca para o acesso aos periféricos de baixo nível em sistema embarcados através da plataforma *Java*. Sua utilização possui a mesma simplicidade das demais bibliotecas do pacote *Java*, pois basta a consulta ao *javadoc* e aplicar as implementações. A figura 5 apresenta a estrutura *javadoc* dos pacotes do *Device I/O API*.

Packages

Package	Description
jdk.dio	Provides interfaces and classes for device I/O access and control.
jdk.dio.adc	Interfaces and classes for reading analog inputs using an Analog to Digital Converter (ADC).
jdk.dio.atcmd	Interfaces and classes for controlling a Data Communication Equipment such as a modem or a cellular module using AT commands.
jdk.dio.counter	Interfaces and classes for counting pulses (or events) on a digital input line.
jdk.dio.dac	Interfaces and classes for writing analog outputs using a Digital to Analog Converter (DAC).
jdk.dio.generic	Interfaces and classes for controlling devices using generic I/O operations.
jdk.dio.gpio	Interfaces and classes for reading and writing from/to GPIO (General Purpose Input Output) pins and ports of the device.
jdk.dio.i2cbus	Interfaces and classes for I2C (Inter-Integrated Circuit Bus) device access.
jdk.dio.mmio	Interfaces and classes for performing memory-mapped I/O.
jdk.dio.modem	Interfaces and classes for controlling MODEM signals.
jdk.dio.power	Interfaces and classes for power management of devices.
jdk.dio.pwm	Interfaces and classes for generating PWM pulses on a digital output line.
jdk.dio.spi	Service-provider classes for the `jdk.dio` package.
jdk.dio.spibus	Interfaces and classes for SPI (Serial Peripheral Interface Bus) device access.
jdk.dio.uart	Interfaces and classes for controlling and reading and writing from/to Universal Asynchronous Receiver/Transmitter (UART), with optional Modem signals control.
jdk.dio.watchdog	Interfaces and classes for using system watchdog timers (WDT).

Figura 5: Pacotes do Device I/O API

6.1.2.3 Emulador

O *Java ME SDK 8 Embedded* incorpora uma emulador de dispositivos para a visualização e controle dos periféricos e recursos encontrados nos dispositivos, na figura 6 apresenta a tela inicial do emulador e na figura 7 a visualização da execução do aplicativo de *PWM*.

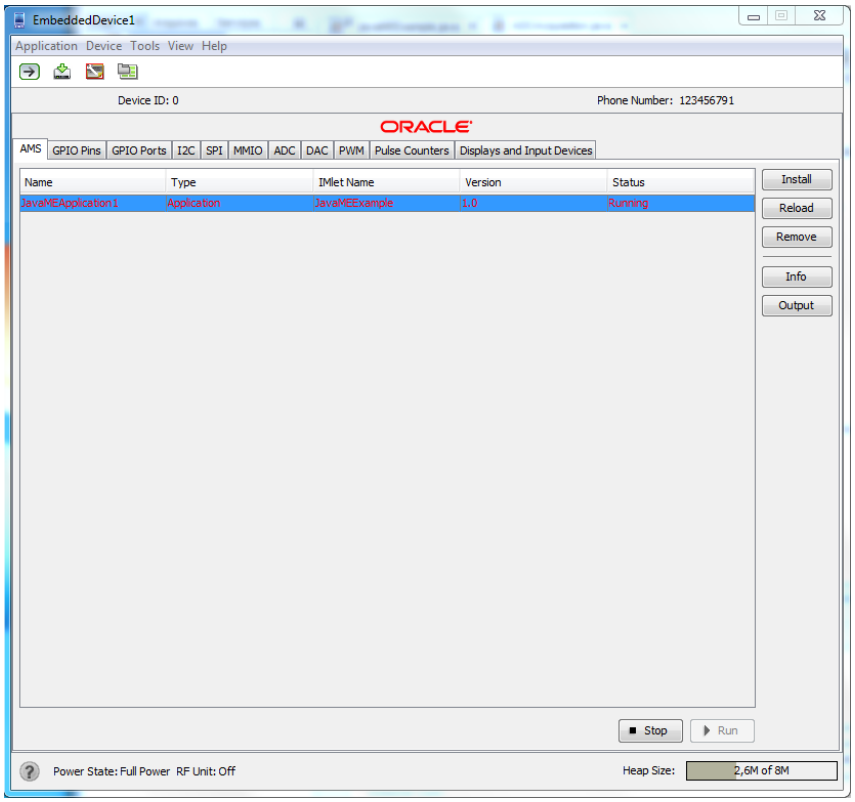

Figura 6: Emulador

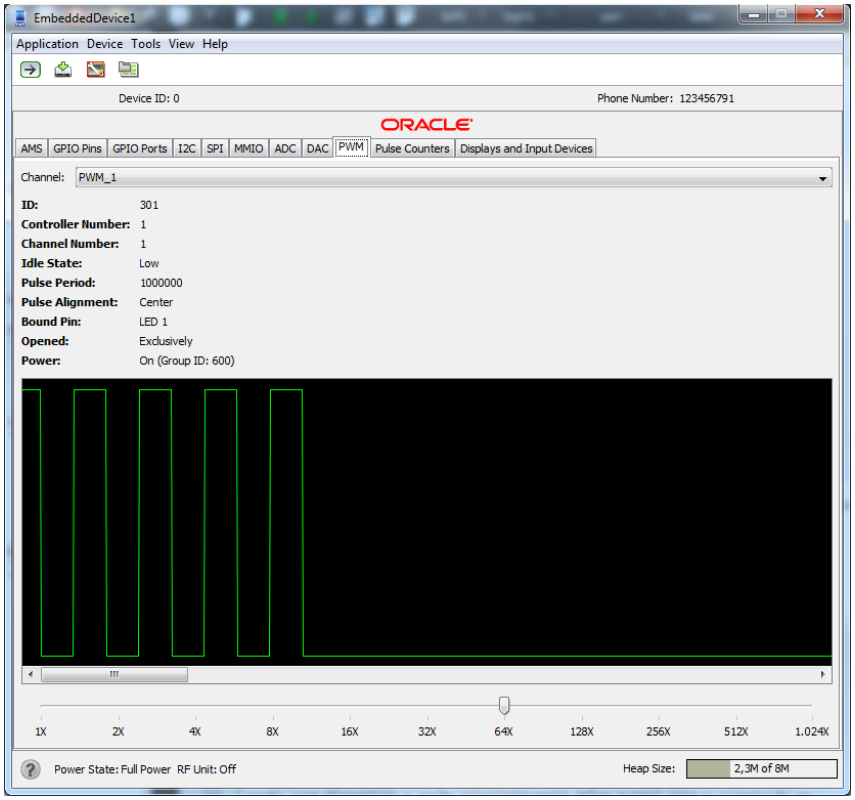

Figura 7: Sinal PWM

Diversas ferramentas são encontradas no emulador, tais como controle de conectividade, visualização da saída padrão e gerador de eventos externos, na figura 8 apresenta a tela para gerar eventos para o periférico Entrada/Saída de Propósito Geral - *General Purpose Input/Output* (*GPIO*).

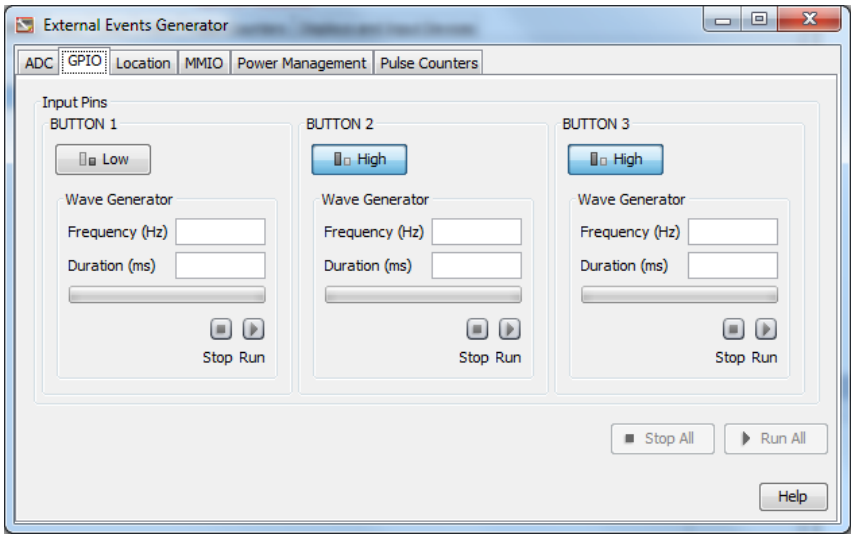

Figura 8: Emulador

6.1.2.4 Implantação

O processo de implantação requer a execução do *script* `usertest.sh` para a inicialização do *Java* no dispositivo remoto. No projeto escolha o dispositivo externo para a plataforma, como na figura 9.

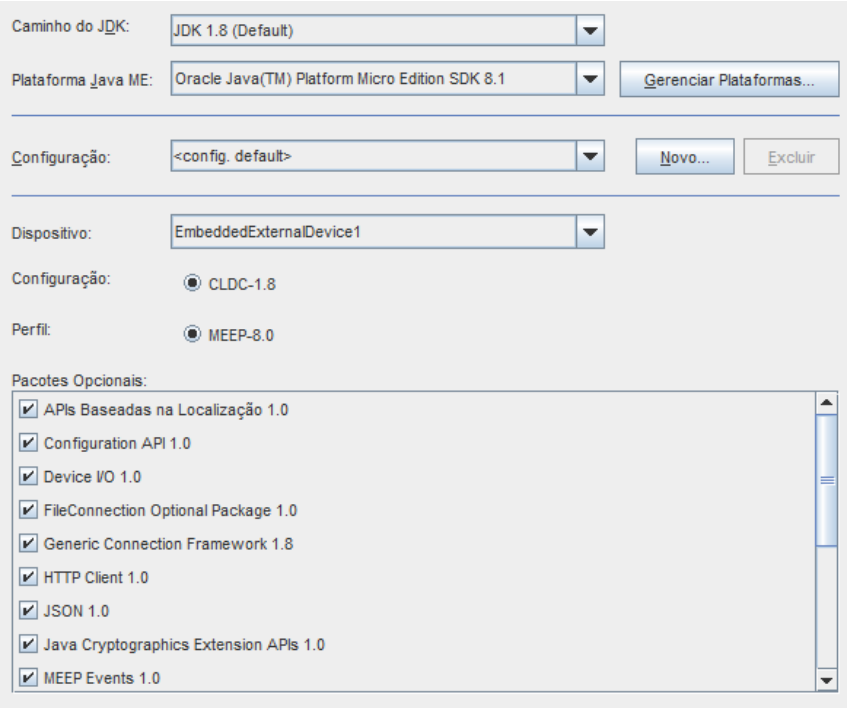

Figura 9: Dispositivo Embarcado Externo

O emulador será conectado ao dispositivo remoto para a execução da aplicação, porém será disponível a opção *Install IMlet Suite* para a instalação no dispositivo.

Outra opção é transferir a aplicação Descritor da Aplicação *Java* - *Java Application Descriptor* (*JAD*) para o dispositivo remoto e executar o *script* `installMidlet.sh` e `runSuite.sh`.

6.2 Aplicações para a Internet das Coisas

O interfaceamento com periféricos de baixo nível permite a utilização de uma gama de fontes de dados, tais como sensores e atuadores. Esses dados podem serem combinados com outros dispositivos embarcados.

6.3 Resultados

Os resultados demonstram a visão na qual a plataforma *Java ME Embedded* contribui para o desenvolvimento de aplicações destinadas a *Internet* das Coisas - *Internet of Things* (*IoT*).

6.3.1 Positivos

- Permite uma abstração dos periféricos de baixo nível, sem a necessidade de endereços e estruturas de *bits* através da *Device I/O API*;
- O Emulador auxilia no desenvolvimento de aplicações, por disponibilizar uma variedade de interfaces de *hardware*.

6.3.2 Negativos

- O *Java ME 8 SDK* não possui uma versão para o sistema operacional *Linux*, o que dificulta durante o ciclo de desenvolvimento;
- A aplicação necessidade ser inicializada pelo Sistema Operacional, para aplicações autônomas.

7 JAVA EMBEDDED SUITE

Este capítulo tem como objetivo demonstrar a plataforma *Java Embedded - Java Embedded Suite* (*JES*).

7.1 Instalação

A plataforma de prototipagem deve estar operacional com o sistema operacional *Linux* e a plataforma *Java 8*, no caso de estudo a plataforma *Raspberry PI B+*. A figura 10 apresenta as configurações do sistema em estudo.

```
pi@raspberrypi ~/Downloads $ uname -a
Linux raspberrypi 3.18.14+ #793 PREEMPT Sat May 30 13:15:19 BST 2015 armv6l GNU/Linux
pi@raspberrypi ~/Downloads $ java -version
java version "1.8.0"
Java(TM) SE Runtime Environment (build 1.8.0-b132)
Java HotSpot(TM) Client VM (build 25.0-b70, mixed mode)
```

Figura 10: Configuração do Sistema

Para a plataforma de prototipagem é necessário a instalação do pacote *Oracle Java Embedded Suite 7.0*, necessário ter uma conta no site da *Oracle* para realizar o *download*.

Transfira os arquivos `jes-7.0-ga-bin-b11-linux-arm-runtime-15_nov_2012.zip` e `jes-7.0-ga-b11-linux-samples-15_nov_2012.zip` do sistema operacional principal para o dispositivo remoto, basta utilizar o comando *SCP*:

```
scp </path/from/file> <user>@<address>:/path/to/destination
```

Descompacte os arquivos, basta utilizar o comando *UNZIP*:

```
unzip <file>
```

O caminho para o *JES* define a variável ambiente `$JES_HOME` no sistema. Os *scripts* já definem as variáveis de ambiente necessárias para o sistema executar cada tipo de aplicação do pacote *JES*.

7.1.1 Pacote Oracle Java Embedded Suite ZIP

O pacote *Oracle Java Embedded Suite ZIP* consiste de cinco diretórios:

- *glassfish*: este diretório contém servidor de aplicações *GlashFish* através da biblioteca `glassfish-jes.jar`;
- *javadb*: este diretório contém o banco de dados *Derby* através das bibliotecas `derby.jar` e `derbytools.jar`;

- *jersey*: este diretório contém o *framework* de aplicações *Jersey*;
- *jre*: este diretório contém o *runtime* para a execução de aplicações *Java*. Não é necessário, quando o *Java 8* encontrasse corretamente instalado;
- *samples*: este diretório contém exemplos de aplicações e os *scripts* para a execução das aplicações.

O arquivo `jes-verify.sh` realiza o diagnóstico da instalação do ambiente de execução.

7.1.2 Desenvolvimento

O processo de desenvolvimento não difere da forma utilizada em servidores tradicionais através de aplicações *Web application*, em codificar e realizar a implantação da aplicação.

O exemplo implementa uma aplicação *Web* para a monitoração de temperatura da Unidade de Processamento Gráfico - *Graphics Processing Unit* (*GPU*) do *Raspberry PI* através do comando `/opt/vc/bin/vcgencmd measure_temp` do sistema operacional.

7.1.2.1 Codificação

A aplicação consiste de um *software Servlet* com a definição do método `doGet()`, sendo o código final apresentado abaixo:

```
// MonitorTemp.java
package ufabc;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(urlPatterns = {"/"})
```

```
public class MonitorTemp extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        response.getWriter().write("<b>Monitoração da GPU<b><br/>");
        Runtime runtime = Runtime.getRuntime();
        BufferedReader br = new BufferedReader(
                new InputStreamReader(
                        runtime.exec("/opt/vc/bin/vcgencmd measure_temp
                        .getInputStream()));
        response.getWriter().write(br.readLine());
    }
}
```

7.1.2.2 Implantação

A implantação da aplicação é por transferir o arquivo para o dispositivo remoto e executar via o *script* para o tipo de aplicação:

- `gfhost.sh`: utilizado para implantar a aplicação *Web* no *JES*, permite múltiplas aplicações ao mesmo tempo;
- `hclient.sh`: utilizado para acessar serviços *Web - Web services*;
- `hstorage.sh`: utilizado para aplicação com o *Java DB*;
- `lwhost.sh`: utilizado para implantar a aplicação de serviço *Web Web service* no *JES*.

O *script* `config.sh` permite a configuração do *classpath* e dos argumentos passados para o *runtime* do *Java*.

Para realizar a implantação do exemplo, basta utilizar o *script* `gfhost.sh` no caminho `$JES_HOME/samples/dist/run`:

```
./gfhost.sh ../../../dist/MonitorTemp.war
```

Ao final da implantação será exibida a mensagem apresentada na figura 11.

Figura 11: Implantação da Aplicação

No navegador basta acessar o endereço do *Raspberry PI* pela porta `8080` e adicionar o caminho da aplicação, conforme apresentado na figura 12.

Figura 12: Monitor de Temperatura da GPU

7.2 Aplicações para a Internet das Coisas

Um servidor de aplicações traz a *Internet* das Coisas - *Internet of Things* (*IoT*) uma interface para a disponibilização de serviços, seja para a comunicação com usuários quanto outros dispositivos embarcados. O *JES* permite a interoperabilidade com aplicações *Java ME Embedded*, trazendo o poder dos periféricos de baixo nível como fonte de dados. Esses serviços agrega as possibilidades de obtenção, armazenamento e processamento de dados.

7.3 Resultados

Os resultados demonstram a visão na qual a plataforma *JES* contribui para o desenvolvimento de aplicações destinadas a *IoT*.

7.3.1 Positivos

- A possibilidade de implantar um servidor de aplicações *Web*, banco de dados e *Web services* em um sistema embarcado;
- Interfaceamento com as classes de aplicações *Java ME Embedded*.

7.3.2 Negativos

- Não possui muitos componentes para os servidores, assim reduzindo as possibilidades de desenvolvimento.

8 JAVA ORACLE EVENT PROCESSING

Este capítulo tem como objetivo demonstrar a plataforma *Java Embedded - Java Oracle Event Processing* (*JOEP*).

8.1 Instalação

A plataforma de prototipagem deve estar operacional com o sistema operacional *Linux* e a plataforma *Java 8*, no caso de estudo a plataforma *Raspberry PI B+*. A figura 13 apresenta as configurações do sistema em estudo.

```
pi@raspberrypi ~/Downloads $ uname -a
Linux raspberrypi 3.18.14+ #793 PREEMPT Sat May 30 13:15:19 BST 2015 armv6l GNU/Linux
pi@raspberrypi ~/Downloads $ java -version
java version "1.8.0"
Java(TM) SE Runtime Environment (build 1.8.0-b132)
Java HotSpot(TM) Client VM (build 25.0-b70, mixed mode)
```

Figura 13: Configuração do Sistema

Para a plataforma de prototipagem é necessário a instalação do pacote *Oracle Java Embedded Suite 7.0* e *Oracle Event Processing for Oracle Java Embedded*, necessário ter uma conta no site da *Oracle* para realizar o *download*.

Transfira os arquivos `jes-7.0-ga-bin-b11-linux-arm-runtime-15_nov_2012.zip`, `jes-7.0-ga-b11-linux-samples-15_nov_2012.zip` e `ofm_oep_embedded_11_1_1_7_1_linux.zip` do sistema operacional principal para o dispositivo remoto, basta utilizar o comando *SCP*:

```
scp </path/from/file> <user>@<address>:/path/to/destination
```

Descompacte os arquivos, basta utilizar o comando *UNZIP*:

```
unzip <file>
```

8.2 Aplicações para a Internet das Coisas

O *Java Oracle Event Processing* permite o processamento em tempo real dos dados amostrados através dos periféricos e recursos encontrados nos dispositivos, assim aplicando técnicas de filtragens e estatísticas para a realização de análises.

8.3 Resultados

Os resultados demonstram a visão na qual a plataforma *Java Oracle Event Processing* contribui para o desenvolvimento de aplicações destinadas a *Internet* das Coisas.

8.3.1 Positivos

- Grande poder de processamento, por oferecer uma gama de recursos para trabalhar com dados;
- Utilização de linguagem de consulta para a manipulação e amostragem dos dados;
- Traz a capacidade de *gateway* para o dispositivo, quando conectado em outros dispositivos.

8.3.2 Negativos

- Não possui uma interface de desenvolvimento produtiva, como encontrada na versão tradicional.

9 ANÁLISE DOS RESULTADOS

O *Java Embedded* é composto de diversos módulos para o desenvolvimento de *software* embarcado, compatível com as necessidades encontradas no conceito de *Internet* das Coisas - *Internet of Things* (*IoT*).

No *Java ME Embedded* são encontradas diversas ferramentas para o desenvolvimento, juntamente com uma rica documentação do *Device I/O API* (Entrada/Saída - *Input/Output*) (Interface de Programação de Aplicação - *Application Programming Interface*). Entretanto a plataforma não é fixa em apenas um sistema operacional, pois existem recursos voltados para *Windows* e outros para *Linux*. A parte embarcada é disponibilizada por uma estrutura de projeto com diversos *scripts* para a execução e/ou interfaceamento com o sistema operacional da aplicação desenvolvida.

No *Java Embedded Suite* (*JES*) são disponibilizadas os principais módulos para aplicações com servidores através de versões reduzidas e otimizadas para o ambiente embarcado, no conceito *IoT* faz uso de tais aplicações por conectividade com outros sistemas e usuários. A plataforma possui interoperabilidade com demais plataformas do *Java Embedded*, trazendo um grande escopo de fonte de dados. O desenvolvimento é compatível com o tradicional no Java EE (Edição Empresarial - *Enterprise Edition*), utilizando um número menor de componentes e sua implantação através de *scripts*, como no *Java ME Embedded* com a vantagem do interfaceamento com o sistema operacional.

10 CONCLUSÃO

O conceito *Internet* das Coisas alcança cada vez mais as aplicações do dia a dia, trazendo novas possibilidades e desafios para o mundo da tecnologia. A tecnologia *Java* incorpora diversas ferramentas para o desenvolvimento de sistemas embarcados através da plataforma *Java Embedded*.

Diversos escopos são preenchidos, de periféricos de baixo nível até processamento de eventos complexos, porém prover a característica de produtividade não é o único ponto de escolha, o desempenho ainda é a principal qualidade visada principalmente em dispositivos com baixo poder computacional. O Java permite o meio termo entre desempenho e produtividade, principalmente pela diversidade de ferramentas e Interfaces de Programação de Aplicação - *Application Programming Interfaces* (*APIs*), como também versões otimizadas da Máquina Virtual Java - *Java Virtual Machine* (*JVM*).

Aspectos importantes a serem previstos estão na especificação e desenvolvimento de protocolos, *frameworks* e sistemas operacionais especializados.

REFERÊNCIAS

ARAUJO, R. B. de. Computação Ubíqua: Princípios, Tecnologias e Desafios. Departamento de Computação – Universidade Federal de S. Carlos (UFSCar).

ASHTON, K. *That 'Internet of Things' Thing*. 2009. Disponível em: <http://www.rfidjournal.com/articles/view?4986>.

BAHGA, V. M. A. *Internet of Things: A Hands-On Approach*. [S.l.]: Vpt, 2014. ISBN 9780996025515.

DACOSTA, F. *Rethinking the Internet of Things: A Scalable Approach to Connecting Everything*. Santa Clara, California: Apress, 2013. (SpringerLink: Bücher). ISBN 9781430257400.

LAMPKIN WENG TAT LEONG, L. O. S. R. N. S. R. X. V. *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*. North Castle Drive, Armonk, NY: IBM Redbooks, 2012. ISBN 9780738437088.

MATEUS, A. A. F. L. G. R. *Introdução à Computação Móvel*. second. [S.l.]: Departamento de Ciência da Computação da UFMG, 2004.

MCEWEN, H. C. A. *Designing the Internet of Things*. United Kingdom, UK: Wiley, 2013. ISBN 9781118430651.

PATTERSON, J. L. H. D. A. *Computer Organization and Design: The Hardware/Software Interface*. fifth. Waltham, MA: Elsevier Science, 2013. (The Morgan Kaufmann Series in Computer Architecture and Design). ISBN 9780124078864.

PRESSER, M. *Inspiring the Internet of Things - Special Edition*. IT City of Katrinebjerg, Aarhus: The Alexandra Institute, 2012.

SMITH, I. G. *The Internet of Things 2012 New Horizons*. Halifax, UK: IERC - Internet of Things European Research Cluster, 2012. ISBN 9780955370793.

TANENBAUM, T. A. A. S. *Structured Computer Organization*. sixth. Upper Saddle River, NJ: Prentice Hall, 2013. ISBN 9780132916523.

WEISER, M. The Computer for the Twenty-First Century. *Scientific American*, p. 94–104, sep 1991.