UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Ambiente para automação via Web Semântica utilizando Linux embarcado em microcontroladores ARM

Patrícia Pires Otoni

Orientador: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos

2013

Patrícia Pires Otoni

Ambiente para automação via Web Semântica utilizando Linux embarcado em microcontroladores ARM

Trabalho de Conclusão de Curso
Apresentado à Escola de Engenharia de São Carlos
Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em Eletrônica

ORIENTADOR: Evandro Luis Linhari Rodrigues

São Carlos
2013

Otoni, Patricia Pires
O087ta Ambiente para automação via Web Semântica utilizando Linux embarcado em microcontroladores ARM / Patricia Pires Otoni; orientador Evandro Luis Linhari Rodrigues. São Carlos, 2013.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2013.

1. ARM. 2. Linux. 3. Web Semântica. I. Título.

FOLHA DE APROVAÇÃO

Nome: Patricia Pires Otoni

Título: "Telemetria via Web Semântica utilizando Linux embarcado em microcontroladores ARM"

Trabalho de Conclusão de Curso defendido e aprovado em 26 / 11 / 2013 ,

com NOTA 9,8 (Nove, oito), **pela Comissão Julgadora:**

Prof. Associado Evandro Luís Linhari Rodrigues - (Orientador - SEL/EESC/USP)

Prof. Associado Ivan Nunes da Silva - (SEL/EESC/USP)

Mestre Alex Antonio Affonso - (Doutorando - SEL/EESC/USP)

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Homero Schiabel

Dedicatória

Dedico este trabalho aos meus familiares, especialmente meus pais, Teresinha e José Wilson, que sempre me apoiaram e fizeram dos meus sonhos os seus e que nunca mediram esforços para que eu continuasse a busca pelos meus ideais.

Agradecimentos

Agradeço primeiramente a Deus, que se fez presente desde o início, dando-me força quando eu já não a tinha mais, dando-me coragem para continuar na batalha quando eu acreditava que ela já estava perdida.

Agradeço aos meus pais, as minhas irmãs, avós, tios, primos e namorado pela confiança depositada em mim, pelo apoio e compreensão nesta jornada.

Agradeço ao Prof. Evandro que desde sempre me apoiou, motivando-me a buscar cada vez mais conhecimento.

Por fim, obrigada a todos que me ajudaram a concretizar este trabalho.

Sumário

Lista de ilustrações

Lista de abreviaturas e siglas

ARM	*Advanced Risc Machine*
HTML	*HyperText Markup Language*
RDF	*Resource Description Framework*
SPARQL	*SPARQL Protocol and RDF Query*
PHP	*Hypertext Preprocessor*
ODBC	*Open Database Connectivity*
DSN	*Data Source Name*
CSS	*Cascading Style Sheets*
WS	*Web Semântica*

Resumo

A Web Semântica tem como objetivo ser uma nova forma de atribuir um significado ao conteúdo da Web, tornando este compreensível não só por humanos como atualmente, mas também por agentes computacionais. Assim, esse projeto visa o estudo e aplicações das ferramentas criadas para a Web Semântica, tais como os *frameworks* RDF (*Resource Description Framework*), RDF-Schema (*Resource Description Framework Schema*) e por fim a linguagem de consulta para documentos RDF "SPARQL" (*Protocol and RDF Query Language*). Com base nos conceitos assimilados construiu-se uma aplicação para microcontrolador ARM com sistema embarcado Linux, explorando também o acesso remoto a banco de dados. Os resultados apresentados demostram, ainda que inicialmente, a viabilidade, a praticidade e as possibilidades para um sistema de automação via WEB acessível a usuários com diferentes níveis de conhecimento técnico, estimulando assim o emprego crescente dos conceitos da Web Semântica para as mais diversas aplicações.

Palavras-chave: ARM, Linux, Web Semântica, RDF

Abstract

The Semantic Web aims to be a new way to assign a meaning to Web content, making this understandable not only by humans as today, but also by computational agents. Thus, this project aims to study and application of the tools created for the Semantic Web, such as RDF frameworks (Resource Description Framework), RDF - Schema (Resource Description Framework Schema) and finally the query language for RDF documents " SPARQL " (Protocol and RDF Query Language). Based on the concepts assimilated built up an application for microcontrollers ARM embedded Linux system, also exploring remote access to the database. The results presented demonstrate, although initially, the feasibility, practicality and possibilities for an automation system via WEB accessible to users with different levels of technical knowledge, thus stimulating employment increasing the concepts of the Semantic Web for the most diverse applications.

Keywords: ARM, Linux, Semantic Web, RDF

1. Introdução

Os microcontroladores com arquitetura ARM têm sido amplamente utilizados nos últimos anos, devido à simplicidade e eficiência das instruções em tal arquitetura, caracterizando-a com boa velocidade de execução e baixa potência consumida. Assim, eles são amplamente utilizados em aparelhos eletrônicos modernos, como celulares, MP3 players, câmeras e dispositivos de automação.

Além disso, sistemas embarcados, cada vez mais, têm se tornado atraentes para o desenvolvimento de novas tecnologias, pois são dedicados a tarefas específicas, podendo-se aperfeiçoar o projeto reduzindo tamanho, recursos computacionais e custo do produto.

De forma genérica, sistemas embarcados podem ser definidos como sistemas que contém pelo menos um processador, o qual é responsável por uma atividade específica. Apresentam basicamente os mesmos componentes que um computador convencional (processador, memória RAM e memória flash, interfaces, conexões USB. Serial, ethernet, dentre outras), mas diferenciam-se no quesito portabilidade e no fato de que podem ser limitados a executar uma determinada tarefa, repetidamente.

Em adição, a Web Semântica, idealizada por BERNERS, HENDLER e LASSILA (2001), é a nova geração da internet, a qual, em termos gerais, possibilita com que as máquinas ligadas na rede façam atividades mais úteis e com que o desenvolvimento de sistemas possa oferecer suporte a interações na rede. De certa forma, ela proporciona aos usuários a capacidade de criar repositórios na Web, construir vocabulários e escrever regras que interoperem com esses dados.

Na forma prática, a Web Semântica tem como objetivo ser uma nova forma de organizar o conteúdo da Web, tornando este compreensível não só por humanos, como atualmente, mas também por agentes computacionais.

Apesar da proposta inicial já ter completado 10 anos, atualmente são poucas as aplicações Web que utilizam este padrão. Ao longo desses anos, já foram desenvolvidos e definidos vários dos padrões necessários, e existem diversas ferramentas para auxiliar o desenvolvimento dessa nova Web, o que falta é o envolvimento de profissionais da área para que possam aplicar tais conceitos e, gradualmente colocá-los em prática e falta também o apoio financeir para financiar pesquisas e desenvolvimentos na área.

Segundo o artigo VERTICAL Applications (2013), a organização W3C (Consórcio World Wide Web) está trabalhando com diferentes setores – por exemplo nas áreas da Saúde, de Governos e de Energia, para promover e melhorar a colaboração, pesquisa,

inovação e adoção da tecnologia de Web Semântica. Na área da Saúde, por exemplo, a Web Semântica auxilia a tomada de decisões no domínio da investigação clinica, e para interoperar informações biológicas e médicas entre as instituições.

Um bom exemplo de aplicação da Web Semântica dentro das aplicações verticais da W3C citadas em VERTICAL Applications (2013) pode-se citar o Grupo Semantic Web Health Care and Life Sciences Interest Group (HCLS IG), o qual tem a missão de desenvolver, defender e apoiar o uso de tecnologias da Web Semântica através de cuidados da saúde, pesquisas clínicas e medicina translacional. Tais domínios tendem a ganhar benefícios da aplicação intra e inter domínio de tecnologias da Web Semântica, uma vez que dependem da interoperabilidade de informações de vários assuntos.

Assim, esse projeto visa o estudo e aplicações das ferramentas criadas para a Web Semântica, tais como os frameworks RDF (Resource Description Framework), RDF-Schema (Resource Description Framework Schema) e a utilização da linguagem de consulta para documentos RDF “SPARQL” (Protocol and RDF Query Language). Serão utilizados os conceitos para o desenvolvimento de uma aplicação em microprocessador ARM com sistema embarcado Linux, disponibilizando a automação via WEB acessível a usuários com diferentes níveis de conhecimento técnico.

1.1 Objetivos

Como objetivos desse trabalho pode-se salientar:

- A implementação de uma plataforma com sistema operacional Linux;
- A implementação dos conceitos e ferramentas da Web Semântica;
- A implementação e adaptação de ferramentas gerenciadoras de conteúdo;
- Desenvolvimento de comunicação entre dispositivos/usuário e dispositivos/dispositivos por meio da internet;

1.2 Motivação

Diante da constante inovação da tecnologia, despertou-se a vontade de criar algo conciliado com tais com avanços e que permitisse uma maior interação entre o usuário e a máquina. Dessa forma, optou-se pela arquitetura ARM, presente na maioria dos *smartphones* e *tablets* atuais. Com relação à Web Semântica, optou-se pela mesma uma vez que ela representa o futuro da Web e propõe estruturas de armazenamento de dados muito mais eficientes que a Web de Documentos atual.

1.3 Estrutura da Monografia

A monografia está dividida em 6 capítulos:

- O capítulo 1 é a introdução da monografia, em que contém a delimitação do assunto tratado e os objetivos da pesquisa.
- O capítulo 2 é constituído de uma descrição dos principais conceitos técnicos e ferramentas que serão utilizados para o desenvolvimento do projeto em questão.
- O capítulo 3 consiste da descrição do projeto realizado e dos materiais utilizados no mesmo.
- O capítulo 4 abrange toda a descrição do desenvolvimento do projeto, incluindo passos de instalação e configuração das ferramentas, criação da interface Web e criação da Base de Dados.
- O capítulo 5 exibe uma discussão e análise dos resultados obtidos, incluindo possíveis trabalhos futuros.
- O capítulo 6 é constituído de considerações finais sobre o projeto.

2. Conceitos técnicos

Neste capítulo serão apresentados alguns conceitos que serão de extrema importância no desenvolvimento do projeto.

2.1 Sistemas Embarcados

Sistemas embarcados podem ser definidos como sistemas desenvolvidos com foco em uma determinada atividade, assim o hardware e o software são fortemente acoplados. Dessa forma, as aplicações de software são elaboradas para a execução em hardwares específicos.

Diferentemente de sistemas de propósito geral, como são os computadores pessoais, um sistema embarcado realiza um conjunto de tarefas determinadas, e, normalmente, com requisitos de funcionamento bem específicos, às vezes até com restrições de tempo real, como são os sistemas que cuidam de usinas nucleares e freios ABS de carros modernos.

Nesses sistemas, o programa a ser executado é armazenado em memória FLASH ou ROM, mostrando assim uma limitação de memória, diferentemente de um PC, no qual as informações são alocadas em disco rígido e a memória pode ser estendida. No entanto, essa memória de armazenamento não elimina a necessidade de existência de memória RAM em um embarcado.

Os sistemas embarcados são utilizados em diversas arquiteturas, cada uma focada em uma determinada aplicação, o que proporciona uma grande vantagem de se trabalhar com estes dispositivos.

Por exemplo, tem-se os sistemas mais básicos e bem difundidos com arquiteturas de 8 bits (pode-se citar PIC 12F,16F,18F (MICROCHIP *8-bit PIC Microcontrollers*, 2013) e o clássico 8051, (INTEL, 2013)), os quais tem aplicações mais simples, como sensoriamento remoto, controle de máquinas de lavar, máquinas eletrônicas de refrigerante ou café, dentre outros.

Dentre os de 16 bits, pode-se citar MSP430 (INSTRUMENTS,2013*)* e os dsPIC (MICROCHIP *16-bit PIC24 MCUs and dsPIC DSC*.,2013). Tais dispositivos são utilizados em aplicações que exigem um processamento matemático maior, como controle de sistemas mecânicos e industriais.

Por fim, têm-se os sistemas de arquitetura 32 bits, os quais são mais complexos e se aproximam, em termos funcionais, dos computadores pessoais. O desenvolvimento para esta família de sistemas acaba acontecendo mais em nível de aplicação, e não em baixo nível como os sistemas que envolvem programação de

componentes de *hardware*. Os dispositivos mais famosos são os ARM (ARM,2013) e MIPS (MIPS,2013).

2.2 Arquitetura ARM

ARM, acrônimo para *Advanced RISC Machine*, é uma arquitetura para processadores amplamente utilizada para sistemas embarcados. Por conter um conjunto de instruções reduzido (RISC), processadores com tal arquitetura apresentam menor consumo frente a processadores com conjunto de instruções CISC, vantagem explorada nos diversos equipamentos portáteis como MP3 *players*, celulares, *tablets* e dispositivos de automação.

A arquitetura ARM, baseado em GOMES, LEITE e CAETANO (2005), começou como um projeto em 1984 na *Acorn Computers de Cambridge*, Inglaterra, para desenvolver um processador que fosse similar ao já usado MOS Technology 6502. Esse projeto foi finalizado em 1985 e nomeado como ARM1.

O desenvolvimento continuou, e o nome original (*Acorn RISC Machine*) foi substituído quando a ARM Ltda foi criada e assumiu as patentes e o desenvolvimento da arquitetura ARM. Atualmente a arquitetura já está em sua 15ª versão, mas suas versões antigas ainda são bastante usadas e desenvolvidas, já que o uso de cada família é voltado para uma área específica do mercado.

Pode-se citar também a família de processadores ARM Cortex-M (CORTEX, 2013), os quais são RISC de 32 bits e oferecem uma gama de variedades com relação aos quesitos eficiência energética, custo menor, melhor performance e facilidade do uso. Atualmente, existem quatro tipos: Cortex-M0, Cortex-M1, Cortex-M3 e Cortex-M4. E, de certa forma, devido aos quesitos citados anteriormente, eles tem sido mais frequentemente utilizados pelos fabricantes de dispositivos.

Existem também várias extensões especializadas para alguma função ou processamento, como o *Jazelle* e o *Thumb*.

As principais características da arquitetura, segundo GOMES, LEITE e CAETANO (2005), são:

- Processador de 32 bits;
- 16 registradores de uso geral;
- Conjunto de instruções extensível com o uso de co-processadores;
- Instruções de três endereços;
- Capacidade de executar instruções de 16 bits usando a arquitetura Thumb;
- Baixo consumo de energia;

- Tamanho do núcleo reduzido;
- Até 16 co-processadores lógicos;
- Arquitetura LOAD/STORE: Operações de processamento de dados não operam diretamente com o conteúdo da memória, somente com o conteúdo dos registradores.

Portanto, a arquitetura ARM é implementada em grande parte dos processadores de 32 bits atuais, principalmente devido a sua versatilidade de uso em diversas aplicações práticas, pelo bom desempenho que apresenta e pela constante atualização de novas versões que são lançadas pela ARM Ltda.

2.3 Web Semântica

Os computadores são úteis para organização e processamento lógicos, mas não são capazes de estabelecer associações de significado. Um computador tipicamente mantém as informações em hierarquias rígidas, enquanto a mente humana tem a habilidade especial de integrar pequenas unidades de informações de forma randômica. Com base nessa constatação, a terceira geração da *World Wide Web* (WWW, Web), cunhada como Web Semântica, envolve o arranjo das ideias e de suas associações de forma não restrita. Assim, um computador poderia representar associações entre coisas que poderiam parecer não relacionadas, mas que de fato, compartilham algum relacionamento.

A ideia da Web Semântica surgiu em 2001, após a publicação de um artigo na revista *Scientific American* (SCIENTIFIC, 2013) por Tim Berners Lee, James Hendler e Ora Lassila, denominado "*The Web Semantic: A new form of Web contente thais is meaningful to computers will unleash a revolution of new possibilities*" (BERNERS, HENDLER e LASSILA, 2001).

Desta lista de autores o primeiro, Tim Berners-Lee, é atual presidente da W3C (*World Wide Web Consortium*), professor do MIT (*Massachusetts Institute of Technology*) no Laboratório de Inteligência Artificial e Ciência da Computação e é tido como um dos maiores gênios ainda vivos. James Hendler trabalha com Inteligência Artificial no Instituto Politécnico Rensselaer (Rensselaer Polytechnic Institute), nos Estados Unidos. Ora lassila é arquiteto e estrategista de tecnologia, trabalha atualmente na Nokia Services e é membro do Nokia's CEO Technology Council.

Dessa forma, Tim Berners Lee via na Web a necessidade de uma evolução, até que ela tenha o poder de fazer com que as informações possuam formatura tal que as máquinas venham a fazer associações entre informações que se relacionam. Quando isso ocorrer de fato, terá sido implementada a Web Semântica.

Para um melhor entendimento da Web Semântica, pode-se utilizar o exemplo abaixo:

- Marley é um cachorro.
- Existe um cachorro chamado Marley.

As duas sentenças acima representam um conjunto de dados, os quais unidos formam uma informação, e apesar das duas sentenças serem diferentes ambas expressam a mesma ideia. Dessa forma, pode-se afirmar que as duas são semanticamente equivalentes, apesar de serem sintaticamente diferentes.

Pessoas com a capacidade de ler e interpretar o português conseguem concluir que “Marley” é um nome próprio e que “cachorro” é um tipo de animal e, portanto entendem que existe um cachorro chamado Marley. E ainda, por meio de conhecimentos passados adquiridos sobre cachorros, o usuário consegue concluir que Marley tem quatro patas, late, é um mamífero, com certeza nasceu da barriga de sua mãe, dentre outras informações.

Entretanto, o computador não consegue obter as mesmas conclusões que um humano teria, surgindo assim o principal objetivo da Web Semântica: criar esta habilidade que as pessoas possuem de inferir conhecimentos, ou seja, criar padrões para que todo o conteúdo Web possa ser entendido não somente por humanos, mas também por agentes computacionais.

2.4 Principais tecnologias e Padrões da Web Semântica

Nesta seção serão abordadas as principais tecnologias e padrões da Web Semântica, dando uma breve descrição de cada uma delas. A Figura 1 representa a estrutura da Web Semântica dividida em camadas, onde cada cápsula faz uso de uma tecnologia e tem uma função específica.

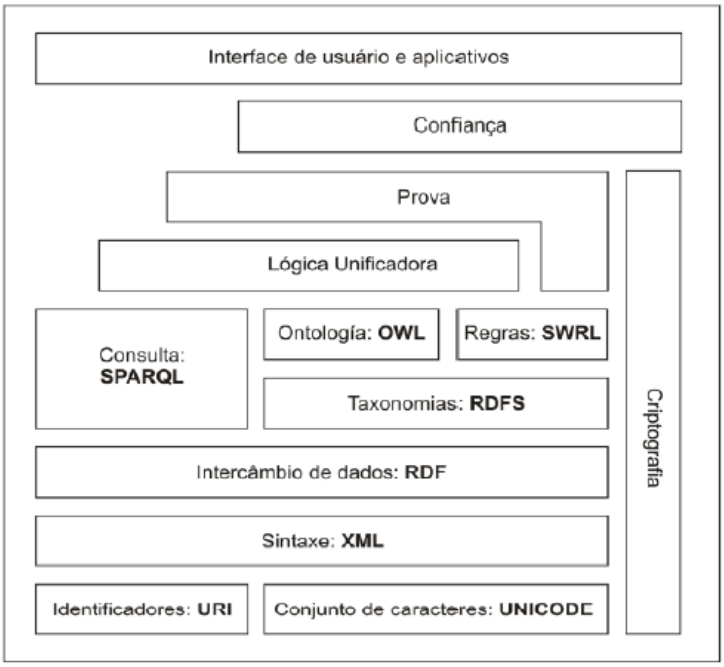

Figura 1: Estrutura de camadas da Web Semântica, retirado de Hebeler et al.(2009)

Em seguida, tem-se a explicação resumida de cada camada:

- **Identificador – URI**: (*Uniform Resource Identifier*), Segundo Hebeler et al. (2009) pode ser definido como uma maneira padronizada de nomear recursos. Geralmente, um URI está ligado a algum tipo de arquivo publicado na Web (página, foto, som...), mas seu uso na Web Semântica é apenas identificar recursos.
- **Conjunto de caracteres – Unicode**: Unicode é um padrão internacional para representação de caracteres, fornecendo um único número para cada caractere (UNICODE, 2013). A Web Semântica adotou esse padrão a fim de garantir a internacionalização das informações.
- **Sintaxe – XML**: (*Extensible Markup Language*) pode ser definida como uma linguagem de marcadores como a HTML e foi desenhada pra escrever dados. Além disso, ela foi projetada para definir documentos estruturados. Dessa forma, a XML serviu como base para outras linguagens da Web Semântica.
- **Intercâmbio de Dados – RDF**: (*Resource Description Framework*) é uma linguagem simples que está apta a prover um modelo de afirmações e citações em que se pode mapear os dados em qualquer novo formato. Ela foi definida como padrão para representar a informação (metadados) sobre recursos por meio de afirmações triplas (recurso, propriedade e valor), fornecendo assim a capacidade para descrever

recursos, usufruindo de uma sintaxe baseada em XML. Mais informações sobre tal linguagem estão no tópico 2.4.1 a seguir.

- **Taxonomias – RDFs**: (*Resource Description Framework Schema*) framework que fornece meios para a criação de vocabulárioas RDF, onde é possível descrever novas classes e propriedades e relacionamentos entre elas.
- **Consulta – SPARQL**: (*SPARQL Protocol and RDF Query Language*) pode ser definido como o protocolo e linguagem para consultas a modelos RDF. Sua sintaxe é similar a linguagem SQL. Mais informações sobre SPARQL estão no tópico 2.4.2 a seguir.
- **Ontologias – OWL**: (*Web Ontology Language*) é uma extensão do RDF, onde é possível a inclusão de mais termos de vocabulário para descrever classes, fatos e relacionamentos sobre estas classes e características destas relações atrás de axiomas lógicos. A OWL baseia-se em linguagens de representação do conhecimento, também conhecida como DL (Lógica de Descrição).
- **Regras – SWRL**: (*Semantic Web Rules Language*) são regras que visam inferior novos conhecimentos em modelos RDF/OWL.

Para facilitar o entendimento da Web Semântica, também chamada de Web de Dados, pode-se estabelecer um paralelo com a Web de Documentos, a qual é utilizada atualmente. Tal comparação é mostrada na Tabela 1.

Tabela 1: Comparação Web de Documentos x Web Semântica

Web de Documentos	*Web Semântica*
Navegadores HTML	Navegadores RDF
Links HTML conectando documentos	Links RDF interligando dados
Mecanismo de identificação – URIs	Mecanismo de identificação – URIs
Mecanismo de acesso – HTTP	Mecanismo de acesso – HTTP
Formato de conteúdo – HTML	Modelo de dados – RDF
	Linguagem de consulta – SPARQL

Os tópicos a seguir detalham duas ferramentas básicas da Web Semântica: RDF e SPARQL.

2.4.1 RDF (Resource Description Framework)

RDF (*Resource Description Framework*) é a especificação padrão para modelagem e intercâmbio de dados na Web Semântica. Ela foi recomendada e aprovada pelo W3C, sendo sua última revisão publicada em fevereiro de 2004.

O Modelo RDF é um modelo de dados descentralizado, baseado em grafo e extensível, possuindo um alto nível de expressividade e permitindo a interligação entre dados de diferentes conjuntos de dados.

Na Web Semântica, a linguagem RDF exerce o papel de descrever um recurso, atribuindo a este, propriedades com seus respectivos valores. Tal atribuição é realizada por meio de triplas constituídas de sujeito, predicado e objeto, onde o sujeito é uma URI, o objeto pode ser uma URI ou um dado literal e o predicado é uma URI que define como o sujeito e predicado estão relacionados.

O RDF é indicado para situações em que a informação precisa ser processada por máquinas e não somente apresentada, pois fornece uma interface que permite que informações sejam trocadas entre aplicações sem perda de consistência. Pelo fato de utilizar URI para identificar recursos e vocabulários, oferece um meio para a eliminação de ambiguidades, mesmo em informações provenientes de várias fontes diferentes.

Como exemplo de uma tripla RDF, a afirmação: “O comando desligar tem como código shutdown” pode ser definida através de uma tripla conforme ilustrado na Figura 2.

Sujeito: <http://seudominio.com/comandos/Desligar>

Predicado: <http://seudominio.com/codigos>

Objeto: “shutdown”

Figura 2: Exemplo de Tripla RDF

Dessa forma, cada tripla faz parte da Web Semântica e pode ser usada como ponto de partida para explorar esse espaço de dados. Triplas de diferentes conjuntos podem ser facilmente combinadas para formar um único grafo. Além disso, é possível usar termos de diferentes vocabulários para representar os dados, como no exemplo acima, em que se utilizou um domínio próprio para as URIs, entretanto pode-se usar domínios que já existem na Web (por exemplo, a propriedade códigos pode já existir em alguma ontologia na WEB). O modelo RDF ainda permite a

representação de dados em diferentes níveis de estruturação, sendo possível representar desde dados semiestruturados a dados altamente estruturados.

2.4.2 SPARQL (*SPARQL Protocol and RDF Query Language)*

Consultas à Web podem ser realizadas por meio da Linguagem SPARQL, a qual é a linguagem de consulta padrão da Web Semântica para recuperação de informações contidas em grafos RDF. Desta forma, SPARQL permite consultar documentos RDF remotamente através da sua linguagem de consulta que é sintaticamente semelhante ao SQL (*Structured Query Language*), sendo essa a linguagem de consulta mais popular entre os desenvolvedores.

Consultas SPQRL são realizadas em um banco de dados RDF por meio de um SPARQL *endpoint.* Um SPARQL *endpoint* é uma interface que usuários (humanos ou aplicações) podem acessar para realizar consultas SPARQL. Para humanos este *endpoint* pode ser uma aplicação *stand*-alone ou Web. Para aplicações, o *endpoint* é um conjunto de APIs usadas para acessar os dados. Um SPARQL *endpoint* pode ser configurado para retornar os resultados em vários formandos (RDF/XML, Turtle, HTML, etc).

Nos tópicos de desenvolvimento do projeto a seguir, será explicado o funcionamento de um *endpoint* e qual foi escolhido para a realização do projeto.

Como exemplo para uma busca SPARQL em uma base de dados RDF, pode-se utilizar a DBpedia: uma comunidade que vem extraindo e estruturando dados da Wikipedia.org em RDF. Atualmente, a DBpedia possui aproximadamente 2.5 bilhões de triplas em RDF. Mais informações podem ser encontradas no site da DBpedia (2013).

A Figura 3 a seguir ilustra uma consulta SPARQL realizada na base RDF da DBpedia, a qual pesquisa todas as corridas as quais Ayrton Senna terminou em 1º lugar.

```
select ?corrida
where {
?corrida <http://dbpedia.org/ontology/firstDriver> <http://dbpedia.org/resource/Ayrton_Senna> .
}
ORDER BY ?corrida
```

Figura 3: Consulta SPARQL na base de dados RDF da DBpedia

Parte do resultado da pesquisa está na Figura 4.

corrida
http://dbpedia.org/resource/1985_Belgian_Grand_Prix
http://dbpedia.org/resource/1985_Portuguese_Grand_Prix
http://dbpedia.org/resource/1986_Detroit_Grand_Prix
http://dbpedia.org/resource/1986_Spanish_Grand_Prix
http://dbpedia.org/resource/1987_Detroit_Grand_Prix
http://dbpedia.org/resource/1987_Monaco_Grand_Prix
http://dbpedia.org/resource/1988_Belgian_Grand_Prix
http://dbpedia.org/resource/1988_British_Grand_Prix
http://dbpedia.org/resource/1988_Canadian_Grand_Prix
http://dbpedia.org/resource/1988_Detroit_Grand_Prix
http://dbpedia.org/resource/1988_German_Grand_Prix
http://dbpedia.org/resource/1988_Hungarian_Grand_Prix

Figura 4: Resultado da pesquisa na DBpedia

Uma observação importante sobre o exemplo anterior, é que o resultado desta consulta foi uma lista contendo todas as corridas que satisfazem a consulta. Entretanto, a lista não se trata de uma lista de *strings*, como geralmente ocorre em consultas SQL a uma base de dados convencional. A lista em questão é uma lista de recursos, ou seja, cada grande prêmio da lista é um recurso, e cada um tem várias propriedades que os descrevem, sendo assim possível obter mais informações sobre cada uma das corridas, e até mesmo, adicionar novos critérios de busca. Esta é uma das vantagens de se utilizar, sempre que possível, um recurso e não apenas uma literal como valor de propriedade.

Dessa forma, é possível notar a contribuição que a Web Semântica e suas ferramentas básicas trazem para a busca na Web, e ainda vale ressaltar que as buscas em SPARQL somente são possíveis em RDF, ou seja, existe a necessidade prévia da reestruturação de conteúdos da Web atual para o padrão RDF, para que se possa então usufruir desse tipo de busca.

3. Descrição do projeto e materiais utilizados

Neste capítulo serão apresentadas a descrição do projeto realizado e dos materiais utilizados no mesmo.

3.1 Materiais utilizados

Para o desenvolvimento do projeto, utilizou-se o kit SAM-L9260 da OLIMEX, mostrado na Figura 5, o qual é constituído principalmente de um microcontrolador ARM9 de 32 bits, 64MB de memória RAM e 512MB de memória flash, para a instalação do sistema operacional Linux (utilizou-se a distribuição Debian).

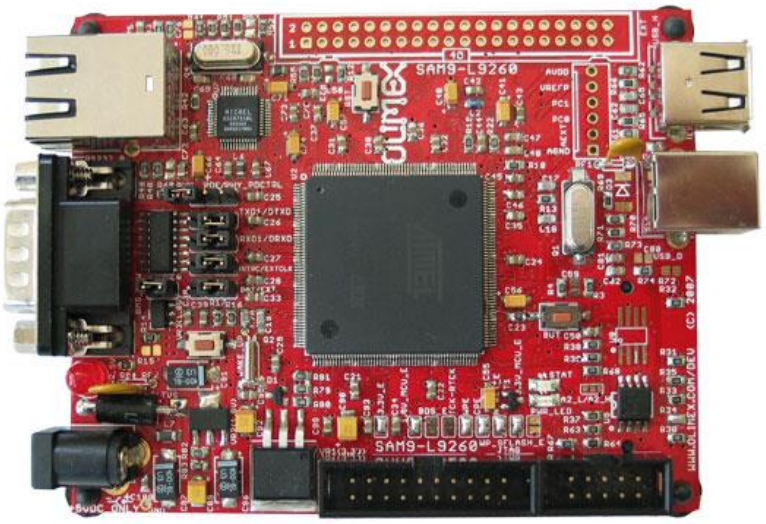

Figura 5: O kit SAM-L9260, retirado do MANUAL Olimex (2013)

O sistema operacional utilizado no desenvolvimento do trabalho foi compilado em trabalho anterior (PASSOS, RODRIGUES, 2011), onde foi atualizado para a nova versão do kernel Linux com suporte a arquitetura ARM e devidamente adaptado de acordo com as limitações de memória do kit.

Mais informações sobre a placa, como desenhos e "*data sheets*" dos componentes existentes pode ser encontradas no MANUAL Olimex (2013).

3.2 Descrição do projeto

Em princípio, os trabalhos desenvolvidos vislumbraram a criação de processos mais simples que possibilitassem a comunicação usuário-máquina por meio da internet. O foco maior foi fazer com que a interação entre ambos se tornasse cada

vez mais simples e acessível, do ponto de vista do usuário, obedecendo aos conceitos e ferramentas da Web Semântica.

Dessa forma, foram utilizadas ferramentas criadas exclusivamente para o uso da Web Semântica, dentre elas a linguagem RDF, RDF-Schema e SPARQL.

Em seguida, foram desenvolvidas funções para que as máquinas pudessem comunicar-se entre si enviando comandos que fossem compreensíveis por elas e pelos usuários. Nesse caso, além do uso da internet e dos conceitos citados nos tópicos anteriores, foi explorada a comunicação via RF entre as placas, como uma proposta de alternativa para casos de falhas na internet.

Foram implementadas ferramentas gerenciadoras de conteúdo para fazer intermediação dessa comunicação, como por exemplo, a utilização de um banco de dados para o sistema de *login* e a plataforma WEB, desenvolvida em linguagem HTML, PHP e CSS, a qual permite o usuário acessar as funções disponíveis apenas após o *login*.

Assim, para um melhor entendimento do projeto, pode-se citar o exemplo:

1. O usuário, após realizar *login* no site, enviará o comando: “placa, por favor, desligue.”

2. A interface Web deverá interpretar a semântica da frase e descobrir qual comando e condições ela deverá seguir. Nesse caso, o comando é “desligar”.

3. Busca-se na base de dados SPARQL o comando “desligar”.

4. Verificar se o comando existe, caso positivo exibe na tela as referências encontradas para aquele comando. Caso negativo exibirá a mensagem de que nenhum comando foi encontrado na base de Dados.

5. O usuário escolherá o comando a ser executado e a ordem desejada, e o kit o executará.

Portanto, em resumo, o projeto consiste em adaptar os conceitos de Web Semântica e das ferramentas gerenciadoras de conteúdo na telemetria via WEB utilizando Linux Embarcado em arquitetura ARM, possibilitando assim uma interação mais simples entre usuário e máquina.

4. Implementação e Desenvolvimento

Inicialmente, fez-se um estudo da plataforma de desenvolvimento e sistema operacional a ser utilizado. Optou-se pelo sistema compilado em trabalho anterior (PASSOS, RODRIGUES, 2011), o qual foi atualizado para a nova versão do *kernel* Linux com suporte a arquitetura ARM e devidamente adaptado de acordo com as limitações de memória do kit.

Nos tópicos a seguir serão discutidos todos os passos e atividades desenvolvidas para que o projeto em questão fosse implementado.

4.1 Escolha das ferramentas da Web Semântica (WS)

Paralelamente ao desenvolvimento da interface Web, a qual será detalhada posteriormente, foi realizado o estudo aprofundado das ferramentas disponibilizadas até então para o desenvolvimento de aplicações com a Web Semântica.

Conforme explicado anteriormente, escolheu-se a linguagem RDF (*Resource Description Framework*) e a RDF *Schema* para a codificação dos dados armazenados. E, para a consulta escolheu-se a linguagem SPARQL. Tal escolha baseou-se no fato de ambas possuírem maior suporte e por serem recomendadas pela W3C, empresa cujo presidente Tim Berners Lee foi o idealizador da Web Semântica.

Após a escolha das linguagens de dados e de consulta, foi necessário um estudo da linguagem de consulta SPARQL e de como é feita a manipulação dos dados RDF. Entretanto, devido ao fato de tal assunto ser pouco utilizado atualmente, o completo entendimento e aprofundamento de tais ferramentas foi um processo demorado e trabalhoso.

Para ajudar no entendimento do assunto, buscou-se plataformas e softwares que já utilizassem tal tecnologia. Assim, após várias pesquisas no assunto, descobriu-se as ferramentas “OpenLink Virtuoso” e a ferramenta “Protegé”. Ambas terão suas aplicações, funcionamentos e justificativas de escolha detalhados nos tópicos a seguir.

4.1.1 OpenLink Virtuoso

OpenLink Virtuoso é um *middleware* e sistema gerenciador de banco de dados que combina a funcionalidade de um banco de dados convencional com banco de dados RDF. Pode ser utilizado como um servidor de aplicação para serviços Web e oferece uma interface Web para consultas em SPARQL.

Ao contrário de outros bancos de dados que armazenam seus dados em linhas, colunas ou pares de chave-valor, um banco de dados de grafos armazena toda informação em uma rede de nós e arestas. As arestas representam o relacionamento entre os nós que representam os objetos. Devido aos nós e arestas serem representados como objetos é possível definir atributos a eles. Adicionando uma direção para uma aresta, cria-se o conhecido grafo de propriedades que representa a explícita estrutura de dados dentro de um banco de dados de grafo.

Além do OpenLink Virtuoso, outras plataformas que suportam a linguagem SPARQL e funcionam como banco de dados RDF foram estudadas, como por exemplo a ferramenta D2RQ (2013) , a qual fornece um ambiente integrado com múltiplas opções para acessar dados relacionais como grafos RDF. Tal plataforma apresenta alguns métodos de acesso: Dumps RDF, APIs RDF e HTML View.

Dessa forma, o OpenLink Virtuoso foi escolhido para ser utilizado nesse trabalho por uma série de fatores, dentre os quais podemos destacar: comandos simples, performático, *OpenSource*, trabalha muito bem com inferências e transitividade, além de armazenar a tripla básica (sujeito, predicado e objeto) ele ainda grava o grafo relacionado, conseguindo trabalhar com múltiplos grafos ao mesmo tempo.

Além das vantagens descritas acima, o OpenLink Virtuoso possui facilidade e flexibilidade para efetuar conexões remotas, utilizando-se de ferramentas como ODBC (explicado no item 4.3).

A interface HTML do OpenLink Virtuoso é bastante completa e permite a visualização gráfica de todas as funcionalidades da plataforma. A Figura 6 ilustra tal interface.

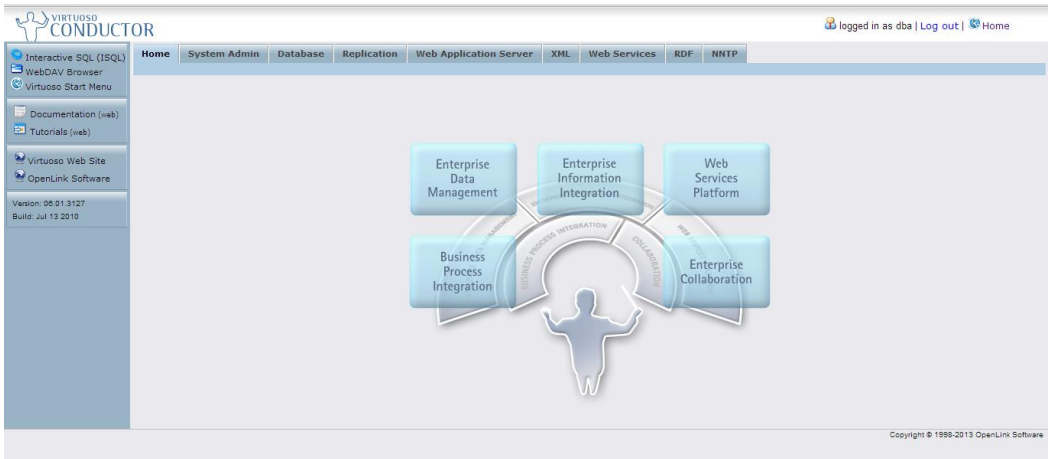

Figura 6: Interface HTML do OpenLink Virtuoso

Segundo o MANUAL do OpenLink Virtuoso (2013), pode-se dizer que o mesmo é revolucionário, é a próxima geração, é o banco de dados de alta performance para a era da Computação Distribuída, é a tecnologia de acesso a dados relacionais definida para acelerar os avanças na era da Informação conduzida pela Web Semântica.

4.1.2 Protegé

Protegé é uma plataforma desenvolvida pelo grupo de pesquisa *Stanford Medicina Informatics* da escola de medicina da Universidade de Stanford e que permite construir ontologias, personalizar formulários de entrada de dados, inserir e editar dados, possibilitando então, a criação de bases de conhecimento guiadas por uma ontologia.

Protegé é baseada na linguagem JAVA, é extensível e conta com uma forte comunidade de desenvolvedores, acadêmicos e usuários, os quais se utilizam do Protegé para soluções de conhecimento de áreas diversas como a biomedicina, coleta de informações e modelagem corporativa.

Tal ferramenta foi escolhida devido a vários fatores: é um editor de ontologias de código aberto, suporta a edição via *Web Client* ou via *Desktop Client*, ou seja, o acesso pode ser feito via Web ou via aplicativo instalado na máquina, permite o desenvolvimento de ontologias em vários formatos, como OWL, RDF e XML Schema, permite a utilização de vocabulários já existentes durante a criação da ontologia própria, permite a exportação da ontologia para o formato desejado e possui uma interface gráfica de fácil manipulação.

Sua interface gráfica provê acesso à barra de menus e à barra de ferramentas, além de apresentar cinco áreas de visualização (views) que funcionam como módulos de navegação e edição de classes, atributos, formulários, instâncias e pesquisas na base de conhecimento, propiciando a entrada de dados e a recuperação das informações. A interface da plataforma Protegé pode ser visualizada na Figura 7.

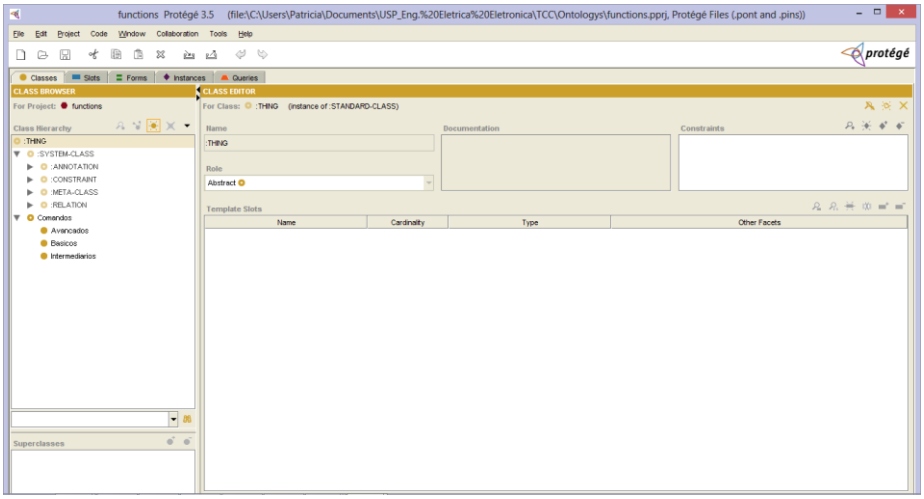

Figura 7: Interface da Plataforma Protegé

4.2 Instalação e configuração do OpenLink Virtuoso

4.2.1 Instalação no servidor

A fim de se familiarizar mais com a ferramenta OpenLink Virtuoso, e para não perder tempo com a instalação direta no kit Olimex, limitado de recursos, instalou-se primeiramente no servidor do laboratório, o qual possui processador Intel Pentium 4 3.00 Ghz, 1 MB Cache e 2GB de memória RAM, agilizando assim a instalação.

Primeiramente, verificou se o pacote OpenLink Virtuoso estava disponível para o servidor executando o seguinte comando:

```
# sudo apt-cache search virtuoso
```

O comando acima retornou os vários pacotes do aplicativo virtuoso. Optou-se então por instalar a versão completa e mais atualizada *"virtuoso-opensource-6.1".* com seguinte comando:

```
# sudo apt-get install virtuoso-opensource-6.1
```

O comando *apt-get* instala automaticamente o pacote desejado e todas as dependências e requisições do mesmo.

Após instalado, encontrou-se uma gama de dificuldades em acessar o programa, uma vez que não existem muitas informações e detalhes sobre sua instalação no manual. Foi por meio de pesquisas em fórums e conversas com usuários do

aplicativo que se conseguiu o comando para inicializar o Virtuoso e verificar o status do mesmo.

Para inicializar, deve-se primeiramente entrar na pasta onde foi instalado o OpenLink Virtuoso (*/etc/virtuoso-opensource-6.1/*)e executar o comando como super usuário:

```
# sudo virtuoso-t -f
```

Tal comando mostra um relatório de atividades durante sua execução, e percebeu-se que determinadas linhas continham falhas que impediam de continuar o processo de inicialização do Virtuoso:

```
11:37:04 Failed to start listening at SQL port '1111'
11:37:04 Failed to start listening at SQL port '8890'
 11:37:04 Server exiting
```

Pela análise do erro, percebe-se que as falhas estão na “escuta” de duas portas do sistema: 1111 e 8890. Para testar as portas em questão, utilizou-se o comando:

```
# netstat –l
```

Tal comando tem por função listar todas as portas sendo utilizadas pelo sistema, e ainda analisa se estão “abertas” ou não, ou seja, se elas podem ser acessadas livremente. Pelo resultado do comando, percebeu-se que as portas “1111” e “8890” não estavam liberadas.

Após a liberação, executou-se novamente o comando de inicialização do Virtuoso, e o relatório não apresentou nenhum erro.

Consultando o MANUAL do OpenLink Virtuoso (2013) e fórums sobre o assunto, verificou-se como realizar o acesso à ferramenta. Primeiramente, tentou-se o acesso via linha de comandos, o qual pode ser executado pelo comando abaixo.

```
#isql-vt
```

Com isso, foi possível executar alguns comandos na linguagem SPARQL via linha de comando. Entretanto, percebeu-se que o acesso via linha de comando torna-se complicado na medida em que não se possui uma visualização clara de todas as funcionalidades do OpenLink Virtuoso, e além disso, tal acesso possui

certas limitações como, por exemplo, não permitir a importação de uma base de dados em RDF..

Dessa forma, estudou-se sobre a interface HTML da ferramenta e seguindo o MANUAL do OpenLink Virtuoso, conseguiu-se o link de acesso, o qual é constituído do endereço do servidor e da porta responsável pela conexão, 8890. O respectivo link está logo a seguir.

```
http://143.107.235.37:8890
```

A visualização imediata da interface HTML pode ser visualizada na Figura 6, citada no tópico anterior.

Para o acesso às funcionalidades da Interface HTML, é preciso informar o *Login* e Senha de Acesso. Tais informações são inseridas durante o processo de instalação do OpenLink Virtuoso, as quais são:

- *Login: dba*
- *Senha: lavisim*

Após o acesso, percebeu-se que a interface é dotada de muitas subdivisões e possui muitos tópicos que até então não haviam sido estudados. Iniciou-se então a análise das opções oferecidas pela interface e os testes de funcionalidades.

Com isso, concluiu-se que as funcionalidades requeridas para o projeto, Base de Dados RDF e a consulta SPARQL a essa base, estavam concentradas em apenas um tópico, constituído de sete sub-tópicos. Tal tópico pode ser visualizado na Figura 8.

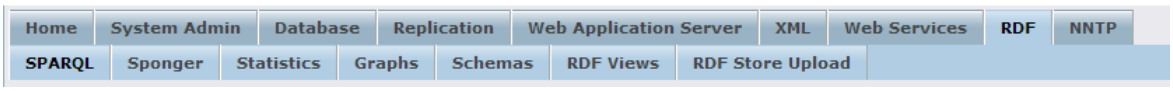

Figura 8: Tópico e sub-tópicos da interface HTML requeridos para o projeto

Os sub-tópicos “Sponger”, “Statistics” e “RDF Views” não foram necessários. Quanto aos restantes, abaixo está uma sucinta explicação:

- *SPARQL:* realiza a consulta SPARQL a toda base de dados contida no OpenLink Virtuoso e exibe os resultados na forma de tabelas. Permite a opção de salvar determinadas consultas para serem realizadas posteriormente.
- *Graphs:* permite inserir grafos já existentes que poderão ser usados ou referenciados na base de dados.
- *Schemas:* permite inserir *Schem*as já existentes que poderão ser usados ou referenciados na base de dados.

- *RDF Store Upload:* permite importar arquivos RDFs já existentes para a base de dados do OpenLink Virtuoso.

A fim de se testar a consulta SPARQL via OpenLink Virtuoso, como primeiro teste digitou-se o seguinte código em linguagem SPARQL na interface HTML:

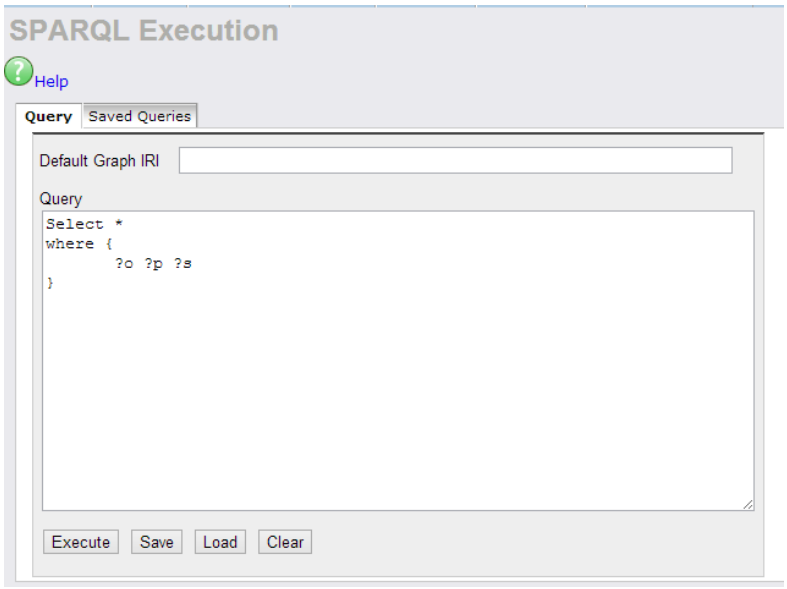

Figura 9: Teste realizado na consulta SPARQL via interface HTML

O comando mostrado na Figura 9 tem como resposta todas as triplas armazenadas na base de dados do OpenLink Virtuoso instalado no servidor.

Depois de instalado, configurado e testado o software OpenLink Virtuoso, o próximo passo foi instalar o software no kit Olimex, criar o código do projeto em linguagem RDF e importar para essa base de dados.

4.2.2 Instalação no Kit Olimex

Da mesma forma que foi realizada a instalação do OpenLink Virtuoso no servidor, tentou-se instalar o mesmo no kit Olimex utilizado no projeto. Pesquisou-se então se existia alguma versão compatível com a arquitetura ARM utilizando o comando:

```
# apt-cache search virtuoso
```

O pacote “virtuoso-opensource-6.1” instalado no servidor não apareceu na lista dos resultados do comando acima. Apareceram apenas pacotes suporte ao

OpenLink Virtuoso. Dessa forma, como tentativa, tentou-se inserir o comando abaixo para forçar o sistema a procurar o pacote desejado.

```
# apt-get install virtuoso-opensource-6.1
```

Entretanto, tal comando teve como resultado o seguinte erro:

```
Package virtuoso-opensource-6.1 is not available, but is referred to
by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'virtuoso-opensource-6.1' has no installation candidate
```

Verificou-se então que não existia a versão compilada do OpenLink Virtuoso para a arquitetura ARM, utilizada no kit Olimex. Dessa forma, foram estudadas possíveis formas de compilação para o mesmo e após várias análises concluiu-se que tal compilação demandaria tempo e trabalho excessivos, correndo-se o risco de sobrecarregar o kit, impossibilitando a execução do pacote, uma vez que o kit, como citado nos tópicos anteriores, é dotado de limitações de hardware.

Portanto, como uma solução para o problema encontrado acima, optou-se por manter o software instalado no servidor, e realizar uma conexão remota do kit com o servidor para acessar o bando de dados. Tal conexão será explicada no tópico a seguir.

4.3 Conexão remota entre Kit e Servidor

A conexão remota à base de dados do Virtuoso tornou-se algo complexo na medida em que não se encontrou suporte necessário em fórums e links da internet. Frente a isso, iniciou-se uma pesquisa a fim de unir o servidor PHP com o Virtuoso, de forma que o PHP suportasse funções que realizassem a conexão. Segundo *DEPLOYING PHP applications using Virtuoso as Application Server* (2013) era preciso instalar duas outras ferramentas: ODS e PHPBB3, as quais são responsáveis pela conexão remota e pela implementação do código em PHP que realizasse a conexão, respectivamente.

Após várias análises na solução apresentada (utilizando ODS e PHPBB3) conclui-se que ela é muito complexa para o projeto em questão e demandaria tempo

para que fosse implementada. Além disso, percebeu-se que tal solução seria para execução na mesma máquina, ou seja, a conexão se daria apenas entre o PHP e a base de dados caso ambos estivessem no mesmo local. E isso não se aplica a este projeto, o qual tem dois locais diferentes para conexão.

Portanto, buscou-se por aplicativos independentes que realizassem a conexão remota, e por meio do MANUAL do OpenLink Virtuoso, foi possível encontrar quais pacotes o Virtuoso oferece suporte. Descobriu-se então o ODBC (*Open Database Connectivity*), o qual é um padrão para acesso a sistemas gerenciadores de banco de dados, e é muito utilizado para conexões com banco de dados conhecidos, como por exemplo MySQL, Access e SQL Server. Entretanto, como restrição, o pacote do ODBC exige a existência de drivers ODBC específicos para as bases de dados as quais se deseja conectar.

Para auxiliar na instalação e configuração do ODBC, utilizou-se como suporte o artigo publicado por Daniel Lewis (Lewis, 2011).

4.3.1 Instalação ODBC no servidor

Da mesma forma que foi feita a instalação do OpenLink Virtuoso, fez-se com o ODBC, ou seja, instalou-se primeiramente no servidor para testes e análises, para posteriormente instalá-lo definitivamente no Kit Olimex.

Primeiramente, se executou o comando para instalar o pacote ODBC:

```
# sudo apt-get install iodbc
```

Após completar a instalação do pacote, instalou-se a biblioteca específica do pacote ODBC, contendo as funções básicas do mesmo:

```
# sudo apt-get install libmyodbc
```

Por fim, para completar a instalação do pacote ODBC e de seus requisitos, instalou-se a biblioteca PHP para conexões utilizando ODBC:

```
# sudo apt-get install php5-odbc
```

Após finalizar as instalações, começou-se a configuração do mesmo para ser utilizado via conexão remota ao Virtuoso. Segundo *PHP SAPI module for Virtuoso*,

alterou-se o arquivo *php.ini* (responsável por armazenar configurações básicas do pacote PHP, localizado em */etc/php5/apache2/)*, adicionando o trecho abaixo:

```
[Virtuoso]
virtuoso.logging = On
virtuoso.local_dsn = Local Virtuoso
virtuoso.allow_dba = 0
```

Dessa forma, a segunda linha do código acima indica que todas as mensagens em PHP são retornadas para o virtuoso por meio de um arquivo log. A terceira linha nomeia como “Local Virtuoso” o DSN indicado no arquivo odbc.ini, o qual será explicado posteriormente. Por fim, a última linha rejeita o uso do *login* ‘dba’ quando se está em um DSN não especificado no arquivo odbc.ini.

Alterou-se também o arquivo odbc.ini (*/etc/odbc.ini*), deixando-o conforme o trecho abaixo:

```
[VOS]
Description = Virtuoso OpenSource
Driver = /usr/lib/odbc/virodbc.so
Address = localhost:1111
UserName = dba
User = dba

[ODBC Data Sources]
VOS = localhost virtuoso
```

A terceira linha do código acima indica o local onde está armazenado o driver do Virtuoso específico para o ODBC. Tal driver é instalado automaticamente durante a instalação do pacote “*virtuoso-opensource-6.1*”.

Para testar a configuração do ODBC com o Virtuoso, executou-se o comando abaixo, o qual exibe todas as bases de dados configuradas no ODBC.

```
# sudo iodbtest
```

O resultado do comando anterior pode ser visualizado na Figura 10:

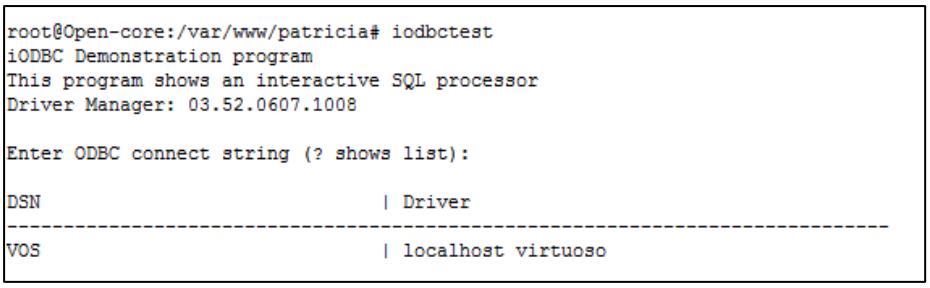

Figura 10: Resultado do comando "iodbctest"

Após verificada a configuração correta da base de dados do virtuoso dentro do ODBC, testou-se a conexão entre ambos via linha de comandos:

```
# sudo iodbctest "DSN=VOS;UID=dba;PWD=lavisim"
```

Tal comando, caso bem sucedido, acessa a base de dados requerida.

Dessa forma, conclui-se a instalação no servidor do pacote de conexão ODBC. Os próximos passos foram: instalar o pacote ODBC no kit Olimex e configurar o pacote PHP para permitir a conexão remota via PHP.

Tais passos serão explicados nos tópicos seguintes.

4.3.2 Instalação ODBC no kit Olimex

Seguindo os mesmos passos utilizados para a instalação no Servidor, instalou-se o ODBC no kit Olimex.

Ao configurar o ODBC no kit para se conectar remotamente ao Banco de Dados Virtuoso localizado no servidor http://www.opencore.sel.eesc.usp.br (http://143.107.235.37:8890), percebeu-se que havia a necessidade de um driver específico do Virtuoso que permitisse o acesso remoto pelo ODBC (virtodbc.so). Após pesquisar sobre o assunto, descobriu-se que tal drive está disponível na biblioteca libvirtodbc0. Entretanto, ao tentar instalá-la na arquitetura ARM ocorreu o erro de incompatibilidade, ou seja, tal biblioteca não existe versão compilada para ARM.

Sendo assim, começou-se um estudo para realizar a compilação de tal biblioteca para a arquitetura utilizada.

Como no momento da execução do projeto a biblioteca não possuía um código fonte disponível para a compilação em arquitetura ARM, e a tal biblioteca ser

dependente de outros pacotes do banco de dados Virtuoso que também não existiam versão compilada para a arquitetura, e a fim de minimizar tempo e trabalho manual (o qual seria extremamente trabalhoso e complexo) optou-se por realizar uma solução que não demandaria o pacote ODBC instalado no kit, muito menos os drivers do Virtuoso próprios para ODBC.

A solução proposta consistiu da utilização dos pacotes instalados da ferramenta ODBC no servidor http://www.opencore.sel.eesc.usp.br, dispensando dessa forma a necessidade de instalação dos pacotes e bibliotecas exigidos pelo ODBC e pelo Virtuoso. Assim, a interface WEB do kit enviaria os dados desejados para o servidor via URL utilizando o método GET e o servidor ficaria responsável por executar a consulta ao banco de dados e retornar a resposta por meio de uma página PHP.

O usuário visualizará a página PHP com os resultados finais, de forma que não perceberá o fato de que a mesma não está sendo executada no próprio kit e então ele escolherá se deseja prosseguir com o comando encontrado pelo banco de dados ou se deseja cancelar a operação.

Após todo esse processo, concluiu-se que apesar da impossibilidade da instalação dos pacotes no kit, a solução final apresentada mostrou-se muito mais inteligente e apropriada, no sentido de que poderá utilizar-se o ODBC e o Virtuoso sem a necessidade de instalá-lo em cada kit. Dessa forma, pensando numa aplicação geral, em que serão utilizados vários kits, a praticidade é muito maior ao precisar instalar os pacotes apenas em um servidor e não em todos os kits.

4.3.3 Conexão remota via PHP

Para permitir o acesso remoto ao Virtuoso via PHP, deve-se ativar o *plugin* do PHP no arquivo virtuoso.ini (*/etc/virtuoso-opensource-6.1/*) retirando o comentário das duas linhas mostradas abaixo:

```
Load7                   = attach, libphp5.so
Load8                   = Hosting, hosting_php.so
```

A fim de testar a conexão via página em PHP, utilizou-se o exemplo do PHP SAPI module for Virtuoso, o qual está localizado no Apêndice A. Assim, caso a conexão seja efetuada, aparecerá “Conexão efetuada” na página, caso contrário, aparecerá “Conexão não efetuada”.

Dessa forma, após certificar o funcionamento correto da conexão remota via PHP, alterou-se o código exemplo para que se efetuasse uma consulta em SPARQL à base de dados do Virtuoso.

O seguinte trecho de código foi adicionado ao código exemplo:

```
$sql ='sparql select * where { ?o ?p ?s}';

$result = odbc_exec($db, $sql );
$imprimir = odbc_result_all($result);
echo $imprimir;
```

Pelo trecho acima se observa que são utilizadas algumas funções da linguagem PHP, como *odbc_exec()* e *odbc_result_all().* Tais funções são encontradas no MANUAL da Linguagem PHP, (2013) o qual possui todas as funções da linguagem PHP e suas especificações.

É importante ressaltar que no código em linguagem SPARQL deve-se adicionar o termo “sparql” antes da *query* a ser executada.

Com o Protegé e o OpenLink Virtuoso instalados e o acesso remoto funcionando corretamente, partiu-se para a criação da base de Dados do projeto utilizando o software Protegé.

4.4 Criação da Base de Dados em RDF

Conforme explicado nos tópicos anteriores, escolheu-se a ferramenta Protegé para a criação da base de dados em RDF.

Seguindo um guia prático de construção de ontologias OWL (Horridge, 2008), criou-se primeiramente um projeto com o formato “*RDF files*”, uma vez que o objetivo final é criar arquivos RDF para serem importados no software OpenLink Virtuoso. A Figura 11 mostra a janela de criação de um novo projeto no Protegé com todas as opções de formato.

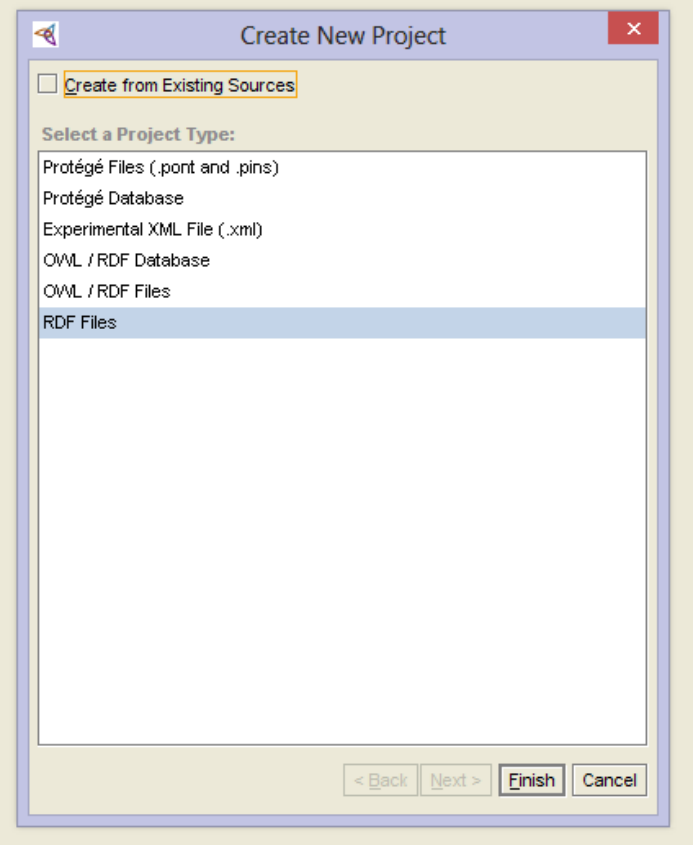

Figura 11: Janela de criação de projeto no Protegé

Para a construção da estrutura da Base de Dados em RDF foram criadas primeiramente as Classes para organizar os recursos e suas propriedades. As classes e subclasses criadas estão listadas a seguir com suas descrições:

- Classe: Comandos";
- SubClasses: "Básicos", "Intermediários" e "Avançados".

Em seguida, foram criadas as propriedades, denominadas como *slots* pelo Protegé:

- *code* : responsável por conectar o código com seu respectivo comando;
- *description*: contem a descrição do comando;
- *keywords*: responsável por conectar sinônimos e palavras chave pré-definidas com o respectivo comando. Poderá conter outro recurso.
- *name:* contém o nome do comando.

Após ter definido as classes, subclasses e propriedades, foram criadas as instâncias, as quais representam os próprios comandos com suas atribuições. Tais instâncias estão representadas na Tabela 02.

Tabela 2: Instâncias criadas com a ferramenta Protegé

Nome	Descrição
Reiniciar	O comando reiniciar é responsável por reiniciar o kit desejado.
Desligar	O comando desligar é responsável por encerrar o funcionamento do kit
Hora/Data	Mostra a data e o horário do kit.
Memoria	Exibe a memoria RAM do kit.
Processos	Exibe os processos em execução no kit.
Hardware	Exibe o hardware que está sendo utilizado para o projeto.
Processador	Exibe o Processador do kit.
Sistema Operacional	Exibe o Sistema Operacional instalado no kit.
Sistema de Arquivos	Exibe o Sistema de Arquivos em funcionamento no kit.
Piscar Led	Pisca o LED do Kit Olimex.

Portanto, com a base de dados criada, exportou-se o projeto para o formato RDF. A Figura 12 mostra os passos da exportação:

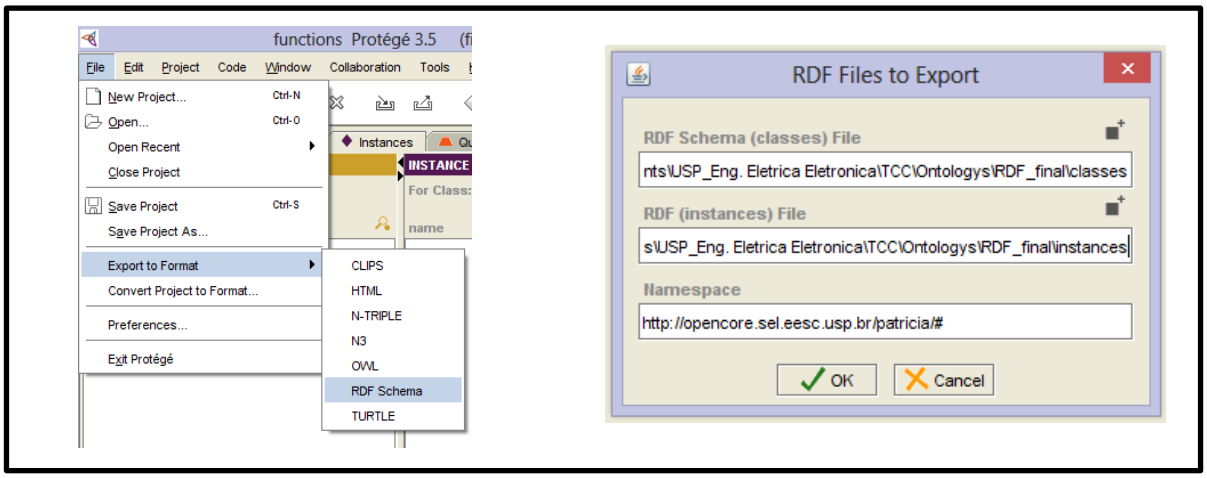

Figura 12: Exportação do projeto para formato RDF

Na Figura 12 pode-se observar que ao exportar o projeto para o formato RDF, cria-se dois arquivos: um com as definições das classes, e outro com as instâncias. Além disso, define-se também um "*namespace*", que representa o link do grafo a ser salvo, o qual será referenciado nas consultas em SPARQL.

Os dois arquivos no formato "*.rdf* " gerados encontram-se no APÊNDICE B.

Por fim, importou-se ambos os arquivos para o OpenLink Virtuoso utilizando a função "*RDF Store Upload*" da interface HTML do mesmo, mostrada na Figura 8. O próximo passo foi desenvolver a interface WEB para que ele interprete os comandos e os execute.

4.5 Interface WEB

A Interface WEB foi desenvolvida em paralelo com as atividades detalhadas nos tópicos anteriores. E, como já descrito anteriomente, tem-se que o principal meio de comunicação do usuário com o kit em questão é por meio da interface WEB. Dessa forma, criou-se a estrutura básica do site representada pela Figura 13 a seguir, utilizando-se a linguagem HTML, PHP e CSS.

Figura 13: Interface Web Página Principal

Os tópicos a seguir detalharão cada rotina contida nos links do menu a esquerda da Figura 13.

4.5.1 Sistema de Login

Primeiramente, para o funcionamento do Sistema de Login, estudou-se a possibilidade de instalação de um banco de dados MySQL no kit, e constatou-se que não seria possível, uma vez que o kit possui limitações importantes de quantidade de memória para esse tipo de aplicação. Dessa forma, a solução encontrada foi instalar apenas a versão cliente do MySQL no kit, e manter a versão servidor em um outro micro disponível no laboratório, o qual inclusive já possui a função de servidor. Assim, o kit apenas se conectará com o banco de Dados localizado no computador servidor do laboratório. Mas ressalta-se aqui que o banco de dados poderia ser instalado em qualquer outra máquina com acesso à internet.

Por meio da ferramenta "*phpmyadmin"* obtém-se acesso ao banco de dados do servidor já instalado. A janela de acesso, juntamente com o usuário e senha é mostrada na Figura 14:

Figura 14: Janela de acesso ao banco de Dados MySql

Como o MySql do servidor é utilizado por vários projetos diferentes, para fins de organização criou-se um novo banco de dados com o nome de “Web Semantica”, o qual será utilizado para a criação das tabelas do Sistema de Login.

Por experiência na área de banco de dados, sabe-se que o *phpMyAdmin* bloqueia IPs que não estejam autorizados a realizar o acesso àquele banco de dados. Dessa forma, para corrigir isto, deve-se alterar as configurações do *mysql* do servidor e liberar o acesso para o IP do kit utilizado no projeto (143.107.235.53 com senha *lavisim*).

Via linha de comandos, acessa-se o *mysql* do servidor:

```
# mysql -u root -p
```

Após preencher o campo de senha do usuário root (“*lavisim*”), deve-se digitar uma sequência de comandos que alteram os privilégios do IP desejado:

```
mysql> use mysql
mysql> GRANT ALL *.* to root@'143.107.235.53' DENTIFIED BY 'lavisim'
mysql> FLUSH PRIVILEGES
mysql> EXIT
```

Após configurado corretamente o MySql para o kit Olimex, criou-se o Sistema de *Login* na interface WEB.

O Sistema de *Login* pode ser acessado pelo link “Login” no menu a esquerda, como pode ser visualizado na Figura 15. Dentro de *login* tem-se as seguintes opções: logout, cadastro, alterar senha e administrador, conforme mostra a Figura 15 a seguir.

Figura 15: Sistema de *Login*

Onde as abas têm as seguintes funções:

- ***Logout:*** funcionará apenas quando o usuário já está logado no sistema e deseja sair do mesmo.
- ***Cadastro:*** registra os dados dos usuários interessados em realizar cadastro no site no banco de dados armazenado no Servidor. Nesse link, o usuário deverá fornecer seu nome completo, nome de usuário, email e a senha de acesso, como pode ser visto na Figura 16 a seguir:

Figura 16: Cadastro do usuário na Interface Web

Tais informações são enviadas para serem armazenadas no banco de dados do servidor. Caso os campos “usuário” e “email” já existam, o cadastro não será concluído e uma mensagem será exibida ao usuário informando que o usuário e/ou email já estão cadastrados no sistema.

- **Alterar senha*:*** apenas disponível para os usuários que estão “ logados” no sistema. Permite ao mesmo a alteração da sua senha de acesso.
- **Administrador:** possui acesso restrito apenas ao administrador/desenvolvedor do site, e possui funções de listar todos os usuários cadastrados no sistema, excluir usuários e alterar os dados do próprio administrador. Para alterar o *login* do administrador, é necessário

usar a ferramenta phpMAdmin, e alterar os dados diretamente em sua interface Web.

Dessa forma, o Sistema de *Login* possibilita segurança ao kit, uma vez que apenas os usuários cadastrados poderão ter acesso às funções disponíveis.

4.5.2 Funções

Essa é a parte do site mais importante para o projeto, uma vez que todas as funcionalidades e aplicações do projeto estão alocadas nessa parte da interface Web. A Figura 17 ilustra a página de Funções:

Figura 17: Página Funções da Interface Web

Como pode ser visto na Figura 17, a página é composta por 3 sub divisões:

- Enviar comandos
- Listar Comandos
- Comunicação RF

Cada uma será detalhada e explicada nos tópicos a seguir.

4.5.2.1 Enviar Comandos

Abre uma página onde o usuário digita o comando desejado para ser executado pelo kit em questão. Dessa forma, ele poderá digitar uma frase que será interpretada pelo kit. A frase poderá ser de duas formas:

- **Objetiva:** O comando na forma pura, por exemplo: 'desligar';
- **Semântica:** O comando dentro de uma frase, por exemplo: 'Desligue o kit por favor';

Considerando o fato de que o kit não possui o ODBC configurado para a conexão remota com a base de dados RDF armazenada no OpenLink Virtuoso, esta página é responsável apenas por captar a frase digitada pelo usuário e armazená-la em uma

variável **$cmd** por meio de um formulário com o parâmetro GET. Tal variável é passada então para uma página do servidor OpenCore utilizando o comando em PHP abaixo:

```
header("Location:
http://opencore.sel.eesc.usp.br/patricia/TCC/consulta.php?cmd='$cmd'
");
```

Dessa forma, a página *consulta.php* localizada no Servidor é chamada e executada. Ela é a responsável pela busca na base de dados do Virtuoso e retorna o resultado encontrado.

É interessante notar que o usuário do site não percebe a mudança de localização das páginas, ou seja, ele não percebe que ao enviar o comando, a próxima página está sendo executada no servidor e não kit Olimex em questão.

Dessa forma, ao receber a frase digitada pelo usuário na variável **$cmd**, a página *consulta.php* conta a quantidade de palavras na string, e salva cada palavra separadamente dentro de um array. Tais comandos estão listados abaixo.

```
$cont = str_word_count ($cmd, 0);
$words = explode(" ", $cmd);
```

Em seguida, conecta-se ao banco de dados em RDF do OpenLink Virtuoso, conforme explicado anteriormente, e então é realizada a consulta em SPARQL com cada palavra armazenada no vetor **$words.** O resultado da pesquisa é armazenado na variável **$result** por meio da função em PHP ***odbc_exec():***

```
while ($words[$i] <> NULL) {
 $sql = " sparql PREFIX ex:
<http://opencore.sel.eesc.usp.br/patricia/>
 select ?Basicos ?name ?description ?code ?keywords where {

?Basicos ex:name ?name .
?Basicos ex:description ?description .
?Basicos ex:code ?code .
?Basicos ex:keywords ?keywords .
FILTER regex(str(?keywords), '$words[$i],', 'i') .
} ";
$result = odbc_exec($db, $sql);
```

Dessa forma, a linha que começa com FILTER do código anterior é responsável por filtrar apenas as instâncias que possuam alguma relação com a respectiva palavra do vetor **$words**. É importante destacar que a busca acima também verifica caso a palavra faça referência a mais de uma instância.

Para uma melhor organização do resultado da pesquisa, optou-se por utilizar como apoio uma tabela no banco de dados MySql do servidor OpenCore, no endereço http://opencore.sel.eesc.usp.br. Assim, a cada iteração do *loop while*, ou seja, a cada palavra da frase digitada pelo usuário, o sistema salva o resultado da pesquisa (nome, descrição, código e sinônimos) de cada instância na tabela "Comandos", como pode ser visualizado nos dois trechos de códigos a seguir.

```
while (odbc_fetch_row($result)) {

        $name = odbc_result($result,2);
        $description = odbc_result($result,3);
        $code = odbc_result($result,4);
        $keywords = odbc_result($result,5);
```

Primeiramente, a função odbc_fetch_row() dentro do *loop while* é responsável por percorrer todas as linhas existentes na variável **$result**, a qual contém o resultado da pesquisa SPARQL. Assim, as quatro linhas abaixo do *loop while* são responsáveis por separar cada propriedade da respectiva instância em uma variável.

```
$num = $j;
if ($name <> NULL && $description <> NULL && $code <> NULL &&
$keywords <> NULL){
        $mysql = "INSERT INTO Comandos (num, name, code,
description, keywords) VALUES ('$num','$name', '$code',
'$description', '$keywords')";
        mysql_query($mysql) or die ("NÃ£o foi possÃ-vel inserir os
dados.");
        $j++;
 }
```

Na função IF do código acima, é verificada a existência de todas as propriedades da respectiva instância, e com sintaxe "INSERT INTO" essas propriedades são inseridas na tabela Comandos em suas respectivas colunas especificadas dentro dos parênteses.

Nota-se também a existência de uma variável denominada **$num**, a qual não tem referência com os recursos da base de dados de RDF. Tal variável é utilizada apenas para nomear numericamente todas as instâncias encontradas com a busca SPARQL.

Com isso, o próximo passo é realizar uma consulta à tabela Comandos e exibir na página todas as linhas lá inseridas, as quais correspondem ao resultado da pesquisa SPARQL.

Para melhor entendimento, digitou-se a frase conforme mostra a Figura 18.

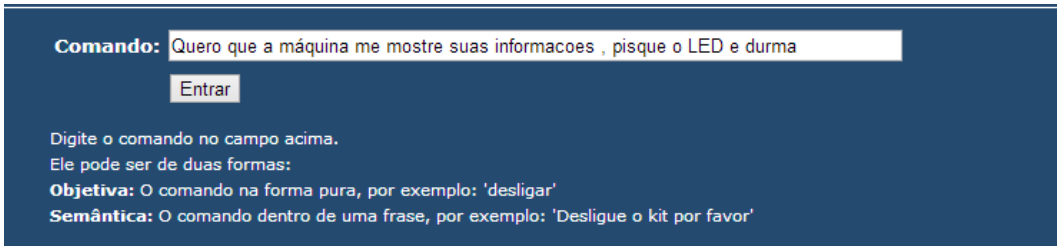

Figura 18: Exemplo de funcionamento do Projeto com a inserção de uma frase exemplo

Ao clicar em entrar, mostra-se o seguinte resultado exibido na Figura 19:

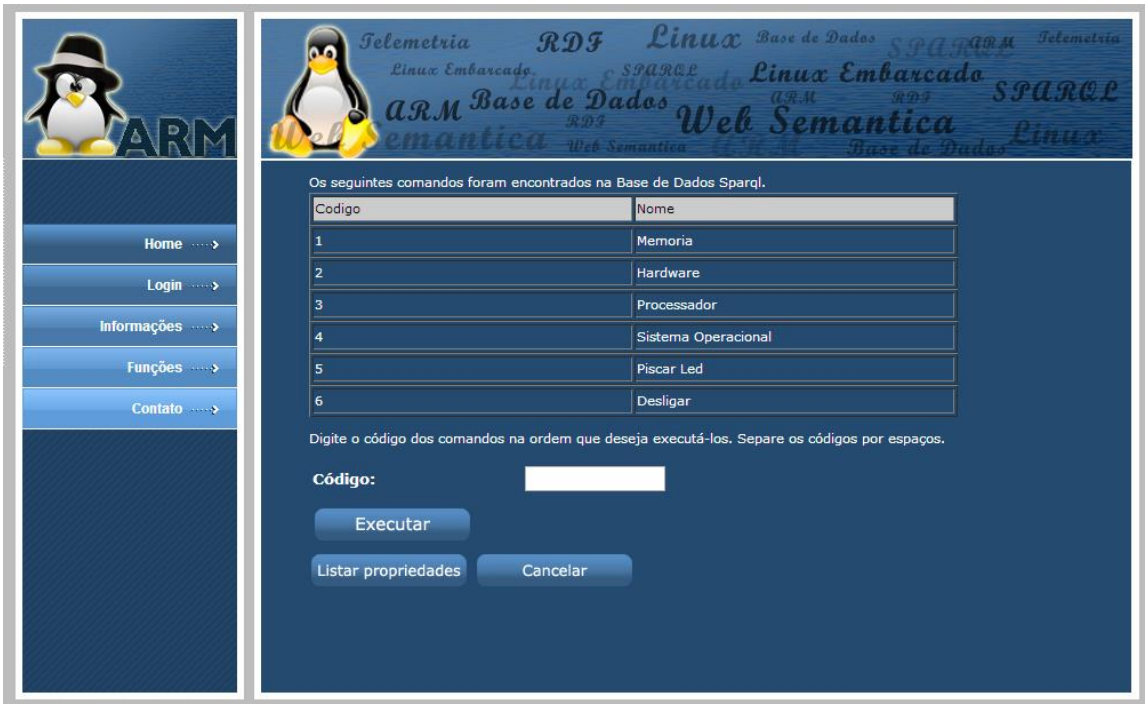

Figura 19: Resultado da pesquisa Exemplo

A partir dos resultados exibidos na tela, o usuário deverá escolher quais comandos deverão ser executados e a ordem dos mesmos preenchendo o campo

“Código” com os respectivos números dos comandos desejados e clicar em Executar, conforme mostra a Figura 19. Além disso, o usuário pode também optar por listar as propriedades de todos os comandos da tela, ou pode cancelar o processo e não executar nenhum comando.

Após escolher os comandos, a ordem de execução e clicar em Executar, o sistema volta a ser executado diretamente no kit, uma vez que a partir desse instante os comandos deverão, de fato, ser executados no próprio kit.

Dessa forma, a execução se dá da seguinte forma:

1. Ao escolher a ordem dos comandos, separa-se a *string* obtida em variáveis independes, de modo a separar os números de cada comando. Tal separação é realizada utilizando a função *explode()* a qual divide a *string* em cada espaço encontrado, como pode ser visto no código abaixo:

```
$nums = explode(" ", $num);
```

2. Em seguida, utiliza-se uma tabela de apoio no banco de dados MySql denominada “comandos_executar”, na qual são armazenados os números dos comandos e, em princípio marca-se como “ NÃO” todas as linhas da segunda coluna, indicando que tais comandos ainda não foram executados. Tal tabela é zerada e inicializada a cada busca do usuário, uma vez que ela tem utilidade apenas para a execução dos comandos desejados pelo usuário naquela busca específica.

Dessa forma, para salvar os dados na tabela, conta-se primeiramente a quantidade de linhas a serem inseridas a partir da sequência digitada pelo usuário, e atribui-se o resultado à variável **$cont.** Como a função *str_word_count()* é voltada para a contagem de palavras, é necessário incluir os caracteres numéricos “0123456789”, como pode ser visualizado abaixo:

```
$cont = str_word_count ($num, 0, '0123456789');
```

Por meio de uma variável auxiliar **$i** e da sintaxe INSERT INTO em linguagem SQL, insere-se os dados no banco de dados até que **$i** alcance **$cont**, como pode ser visualizado abaixo:

```
   for ($i=0;$i<$cont;$i++) {
          $temp = $nums[$i];
          $mysql = "INSERT INTO comandos_executar (code, exec) VALUES
('$temp','nao')";
          mysql_query($mysql) or die ("Não foi possível inserir os
dados.");
   }
```

3. Por meio da função *mysql_query()* realiza-se uma busca a todos os dados das duas tabelas utilizadas: “Comandos” e “comandos_executar” e atribui-se a duas variáveis distintas: **$result** e **$result2** respectivamente:

```
   $result = mysql_query('select * from Comandos');

   $result2 = mysql_query('select * from comandos_executar');
```

Em seguida, utilizando-se dois *loops while, p*ercorre-se cada linha da tabela “comandos_executar” por meio da função *mysql_fetch_array()*, atribuindo a uma variável a respectiva linha da tabela. E em seguida, da mesma forma, percorre-se cada linha da tabela “comandos”, procurando-se o respectivo número e resgatando assim o código do comando. O código utilizado está exemplificado a seguir:

```
   while($results2 = mysql_fetch_array($result2)){
          $num2 = $results2['code'];
          while($results = mysql_fetch_array($result)){
                  if ($num2 == $results['num']) {
                          $exec = $results['code'];
                          $mysql = "UPDATE comandos_executar SET exec =
'ok' WHERE code = '$num2'";
                          mysql_query($mysql) or die ("NÃ£o foi
possÃ-vel inserir os dados.");
                  }
          }
   }
```

Observando o código anterior, é importante destacar que ao encontrar o número desejado na tabela “Comandos”, antes da execução do código, altera-se a segunda coluna na tabela “comandos_executar” para “ok”, indicando que tal código já foi

executado. Além disso, o código do comando a ser executado é armazenado na variável **$exec**.

4. Sabendo-se que o código do comando armazenado em cada linha representa o nome da página em PHP que executará o respectivo comando, a página é chamada por meio do código abaixo:

```
header ("Location: http://143.107.235.53/comandos/$exec.php");
```

Escolheu-se essa forma de execução devido à complexidade que alguns comandos podem possuir, e às particularidades de cada um, impossibilitando assim que ele seja armazenado na base de Dados RDF.

Além disso, alguns comandos exigem permissão de super usuário para serem executados, o que será explicado posteriormente com mais detalhes diante do exemplo dado.

Portanto, para cada comando existente na base de dados RDF, existe uma página em PHP referente a ele, a qual tem com o nome o campo “código” da base de dados, e é responsável por executar os comandos.

Para que os comandos sejam executados de forma sequencial, dentro de cada página em PHP há uma chamada à página “executar_prox.php” (aproximadamente 10 segundos após a execução de todos os comandos da página) exemplificada a seguir:

```
header ("Refresh:10, http://143.107.235.53/executar_prox.php");
```

A página “executar_prox.php” consulta todas as linhas da tabela comandos_executar, e caso alguma linha esteja com “não” na segunda coluna, o sistema repete todo o processo de busca do código na tabela “comandos” e execução da respectiva página em PHP. O processo sequencial acaba quando todas as linhas possuírem o termo “ok” na segunda coluna.

- ***Exemplo de funcionamento***

No exemplo dado na Figura 18, ao realizar-se a busca na base de dados SPARQL, a tabela “Comandos” no banco de Dados MySql do Servidor OpenCore fica como mostrado na Figura 20:

+ Opções

num	name	code	description	keywords
1	Memoria	memoria	Exibe a memoria RAM do kit.	,memoria,ram,memoria ram,memom,memory,amr,basicas,...
2	Hardware	hardware	Exibe o hardware que estÃ¡ sendo utlizado para o p...	,hardware,placa,kit,hardw,plac,itk,basicas,informa...
3	Processador	processador	Exibe o Processador do kit.	,processador,microcontolador,processad,basicas,inf...
4	Sistema Operacional	sistema_operacional	Exibe o Sistema Operacional instalado no kit.	,sistema,operacional,so,os,sistem,operation,basica...
5	Piscar Led	pisca_led	Pisca o LED do Kit Olimex.	,pisca,LED,piscaLED,piscar,
6	Desligar	shutdown	O comando desligar Ã© responsÃ¡vel por encerrar o ...	,desligas,desligares,deslig,dormir,dorme,durma,dur...

Marcar todos / Desmarcar todos *Com marcados:*

Figura 20: Visualização da tabela Comandos no banco de dados MySql

Após a realização os passos 1 e 2 mostrados na Figura 18 e 19, escolhendo-se a sequência crescente (1 2 3 4 5 6) a tabela “comandos_executar” ficará como mostrado na Figura 21 a seguir.

+ Opções

code	exec
1	ok
2	nao
3	nao
4	nao
5	nao
6	nao

Figura 21: Visualização da tabela comandos_executar no banco de dados MySql

Observa-se que a primeira linha já está marcada como “ok”, uma vez que a imagem acima foi criada logo após a chamada da página em PHP referente ao comando 1: memória.

A página “*memoria.php*” é executada, e o resultado é dado na Figura 22:

Figura 22: Resultado da execução do comando "memória"

Da mesma forma, os outros comandos serão executados e seus resultados exibidos na tela, até que a página final com a mensagem “Todos os comandos foram executados.” seja exibida na tela.

Todas as páginas em PHP respectivas a todos os comandos estão disponíveis no Servidor OpenCore, no endereço http://opencore.sel.eesc.usp.br/patricia/TCC/.

A seguir, será detalhado o funcionamento de três páginas consideradas como mais relevantes em função do seu grau de complexidade: Piscar o LED, Desligar e Reiniciar.

- ***Comando PISCAR LED***

O comando Piscar LED se resume basicamente em acionar e manipular um LED já existente na placa. No entanto, acender e apagar um LED usando Linux Embarcado em ARM não é tão simples quanto parece, visto que envolve a manipulação de hardware do kit, por meio do Sistema Operacional. Assim, utilizou-se um programa em C implementado em trabalho anterior pelos alunos Guilherme Ferraz de Arruda e Stenio Magalhães Ranzini na Disciplina Projeto em Sistemas Digitais, o qual usa a função MMAP, responsável por criar um mapeamento das GPIOs (Generic Pourpose Input/Output) do microcontrolador em um espaço de endereçamento virtual em RAM (compartilhado com o *kernel* Linux) e assim permitir que o usuário possa acionar a GPIO adequada. O mapeamento de memória é o único meio para transferir dados entre o espaço de usuário e o espaço de *kernel* do Linux.

Ao compilar tal programa e executá-lo na plataforma WEB encontrou-se um problema: o programa exige permissão de *root* para que possa ser executado, uma vez que envolve manipulação de hardware no mesmo.

Assim, para permitir acesso root apenas do programa compilado e não do usuário *www-data* (usuário do servidor Apache), ao sistema Linux, alterou-se o código em Linguagem C inserindo a seguinte linha no inicio :

```
setuid(0)
```

Tal código informa o sistema operacional que os comandos após essa linha deverão ser executados como *root*.

Dessa forma, após compilar o código deve-se dar permissão *root* ao arquivo executável, o que pode ser feito via linha de comando, com o comando abaixo:

```
# chmod +s nome_arquivo
```

Com as permissões já concedidas e os programas compilados para serem executados como root, realizou-se uma nova busca na interface Web e verificou-se que ao requisitar o comando Piscar LED, a página em PHP é executada e o respectivo LED do kit Olimex reagiu aos comandos. Enquanto o LED pisca, é exibida na página uma mensagem informando que o comando foi executado corretamente.

Entretanto, tal ação pode ser interrompida a qualquer momento, visto que o LED no kit tem suas funcionalidades próprias e é utilizado por outras aplicações.

- ***Comando DESLIGAR / REINICIAR***

Sabe-se que o comando desligar exige permissão para ser executado, devido a sua importância e ação no Kit. Dessa forma, seria necessário liberar permissão root para o usuário *www-data* executar tal comando, e isso, de certa forma prejudica a segurança do sistema, uma vez que libera o acesso como root ao usuário *www-data*, permitindo-o realizar qualquer operação no sistema. E, consequente, isso daria permissão para que um invasor do site realizasse também operações em nível de *root*.

Frente a isso, encontrou-se a seguinte solução para o problema acima: Ao solicitar o comando “desligar” a página “shutdown.php” criará um arquivo dentro da pasta /var/www/ com o nome *shutdown.server*, como mostra o código seguinte:

```
$fh = fopen("/var/www/shutdown.server",'w');
fwrite($fh,"Shutdown now\n");
fclose($fh);
```

Criou-se o *shell script* mostrado a seguir, o qual verifica a existência do arquivo *shutdow.server* e caso ele exista remove-se o mesmo e executa o comando de desligar. A seguir pode ser visualizado o *shell script* criado e salvo em */usr/local/sbin/checkshutdown.sh.*

```
#! /bin/bash
if [ -f /var/www/shutdown.server ]; then
```

```
    rm -f /var/www/shutdown.server
    /sbin/shutdown
  fi
```

Para que o *shell script* anterior seja executado, adicionou-se uma nova tarefa no *crontab*, agendador de tarefas do Linux, a qual executa o *shell* acima a cada minuto. O arquivo *crontab* está armazenado em /etc/crontab/ e a linha adicionada está representada da seguinte forma:

```
# m h dom mon dow user  command
* *     * * *   root    /usr/local/sbin/checkshutdown.sh
```

Para o comando Reiniciar, realizou-se o mesmo procedimento, trocando-se apenas o código referente a ação reiniciar.

4.5.2.2 Listar Comandos

Esta página tem a função de exibir na tela todos os comandos salvos na base de dados RDF armazenada no OpenLink Virtuoso.

Assim como o arquivo "*consulta.php*", o arquivo "*listacomandos.php*" também está armazenada no servidor, uma vez que necessita conectar-se com o Virtuoso, via ODBC.

A lógica utilizada para a consulta é a mesma explicada no tópico anterior: conecta-se ao banco de dados virtuoso e ao MySql, pesquisa-se por todas as instâncias salvas na base de Dados RDF, salva o resultado em uma tabela de apoio no banco de dados MySql e por fim , exibe o resultado na tela a partir da tabela de apoio.

A única parte que difere do processo realizado no arquivo "consulta.php", é a consulta SPARQL, a qual se dá pelo código mostrado a seguir:

```
 $sql = " sparql PREFIX ex:
<http://opencore.sel.eesc.usp.br/patricia/>
  select ?Basicos ?name ?description where {

            ?Basicos ex:name ?name .
            ?Basicos ex:description ?description .

  } ";
```

O resultado de tal pesquisa pode ser visualizado na Figura 23.

Figura 23: Página Listar Comandos

4.5.2.3 Comunicação RF

Para realizar a Comunicação RF entre dois kits, foram utilizadas antenas de modelo nrf24l01, as quais utilizam a tecnologia SPI para se comunicar com o kit. Assim, por meio de programas em linguagem C, pode-se acessar o hardware para acessar os registradores do microcontrolador fazendo-se uso novamente da função MMAP. Dessa forma, por meio do mapeamento de memória virtual pode-se acessar os endereços dos registradores de memória como se fossem variáveis da aplicação, permitindo a leitura e escrita nos registradores.

A implementação dos programas em C foi realizada em atividades paralelas ao projeto por uma equipe do laboratório (NASCIMENTO, RODRIGUES, 2013). Esse projeto apenas utilizou e aplicou tais estudos em uma interface Web.

Como os programas em C exigem execução como root, foi realizado o mesmo procedimento descrito no item anterior para que eles pudessem ser executados pelos arquivos escritos em PHP: ou seja, dentro do código C adicionou-se a linha *setuid(0)*, e após compilado, deu-se permissão de root para tal arquivo.

Para efetuar a transmissão, deve-se então conectar as antenas em dois kits diferentes e acessá-los via internet. Na plataforma WEB do kit que irá enviar os dados, deve-se acessar o link Comunicação RF e então em Enviar.

Para essa implementação são propostas três caixas de texto para o preenchimento da mensagem, em que cada bloco deverá conter no máximo 32 bytes (tal limite deve-se ao tamanho limitado do buffer da antena) e após digitar a mensagem clica-se em enviar. Abre-se então o site do kit que irá receber, e acessando o link Comunicação RF e Receber, a mensagem recebida é exibida na tela e automaticamente é salva no arquivo “mensagens.txt” juntamente com a data e hora que foi recebida.

Foram feitas funções auxiliares como zerar o arquivo de histórico de mensagens e exibi-lo na tela.

No entanto, identificou-se um erro no programa de transmissão: ao clicar em receber em um determinado kit sem que outro tenha enviado algum dado, o programa retorna 3 blocos de 32 caracteres que realmente não pertencem a nenhuma mensagem válida. Tal erro deverá ser corrigido em trabalhos posteriores.

Ao explorar os módulos da antena reconheceu-se que eles permitem uma rede de comunicação proprietária para automação, uma vez que se pode construir ou utilizar protocolos seguros via um canal RF, em que apenas os proprietários tenham acesso.

A seguir tem-se um exemplo de comunicação RF.

- ***Exemplo de funcionamento***

A fim de demonstrar o funcionamento da comunicação RF entre dois kits, digitou-se três blocos de textos a serem transferidos, conforme mostra a Figura 24:

Figura 24: Mensagem a ser enviada via Comunicação RF

Ao clicar em Enviar, aparece a mensagem de confirmação de envio na tela exibindo os 3 blocos enviados.

Esperam-se aproximadamente cinco segundos, abre-se a interface Web do outro kit, e clica em Receber. O resultado é mostrado na Figura 25:

Figura 25: Mensagem recebida via Comunicação RF

Ao receber a mensagem, o sistema salva a mesma no arquivo “mensagem.txt”, o qual possui o histórico de mensagens recebidas daquele kit. Nesse exemplo, o histórico é constituído apenas por uma mensagem, como é representado na Figura 26.

Figura 26: Histórico de mensagens recebidas do kit

Pode-se conferir o funcionamento da comunicação RF pelos endereços http://143.107.235.53 e http://143.107.235.51.

5. Resultados e análises

A partir desse projeto pode-se perceber que o assunto Web Semântica é pouco difundido atualmente e carece de usuários, experiências e informações sobre o mesmo. Isto, por um lado, acabou prejudicando e atrasando a execução do projeto, e por outro lado, proporcionou um conhecimento e aprofundamento ainda maior sobre o assunto. Dessa forma, grande parte do tempo dedicado a este projeto foi voltado para o estudo das técnicas e ferramentas que seriam utilizadas.

Após o estudo das ferramentas da WS, partiu-se para a instalação das mesmas no kit. Foi então que se deparou com uma imensa dificuldade: as limitações do kit. Devido a baixa quantidade de memória RAM e à arquitetura diferenciada dos computadores convencionais, o Kit não permitiu a instalação das ferramentas requeridas pela Web Semântica, como por exemplo a base de dados OpenLink Virtuoso e o pacote de conexão remota ODBC. Entretanto, tal dificuldade proporcionou uma solução de certa forma mais eficiente e inteligente para a aplicação: a utilização de um Servidor com configurações suficientes para uma boa execução das ferramentas necessárias para o projeto. Com a utilização de um servidor, descarta-se a necessidade de instalação de todas as ferramentas no próprio kit, o que em uma aplicação que exija vários kits seria um tanto quanto prático.

Com relação à solução proposta, executar a pesquisa na linguagem SPARQL à base de dados RDF a partir do servidor, pode-se observar que o kit, juntamente com a interface Web obtiveram um bom comportamento, com um tempo de resposta rápido (Aproximadamente 3 segundos para que o Servidor fosse chamado e a busca fosse executada).

Tendo em vista a gama de conhecimentos adquiridos com este projeto, observa-se que existe uma série de trabalhos futuros que poderão ser realizados em cima do projeto atual. Por exemplo:

- Na própria interface Web poderia existir a opção de adicionar um comando àquela base de Dados e conectá-los com as respectivas propriedades.
- Ainda na interface Web poderia também existir a opção do usuário falar a frase desejada por meio de comando de voz, ao invés de digitar a frase.
- Poderia ser implementado um algoritmo que utilize a Lógica Fuzzy para interligar as palavras digitadas na interface Web com a Base de Dados, otimizando ainda mais a busca pelos comandos.

6. Considerações Finais

Durante os estudos realizados ficou claro que a Web Semântica proporciona inúmeros benefícios na interação usuário máquina, deixando-a cada vez mais informal e acessível, sendo o único requisito nessa interação o conhecimento prévio do usuário de algumas funções que aquela máquina poderá ou não executar.

Além disso, pode-se perceber que a WS fornece um novo cenário a integração de dados, torando-os cada vez mais inter-relacionados, mas em contrapartida, também traz novos desafios, como por exemplo a aplicação das linguagens próprias, RDF, RDF Schema e SPARQL, e a conversão dos dados atuais para o padrão da WS.

Entretanto, durante a execução do projeto, deparou-se com uma gama de dificuldades, que de certa, exigiram um maior esforço e dedicação, como, por exemplo o fato de que o assunto Web Semântico é pouco difundido e carece de usuários, experiências e informações. Além disso, as limitações do kit dificultaram a execução do projeto, uma vez que o mesmo não suportava a instalação das ferramentas necessárias.

Dessa forma, com esse projeto foi possível conciliar a Web Semântica com a arquitetura ARM presente no Kit Olimex e o Sistema Operacional Linux utilizando-se de outra plataforma externa, no caso o Servidor OpenCore disponível no laboratório. Assim, todas as ferramentas necessárias para a execução do projeto puderam ser instaladas em outra arquitetura, que não a ARM, e o Kit Olimex utilizou-se de tais ferramentas via conexão remota. E a partir disso, observa-se que a Web Semântica pode ser utilizada em qualquer plataforma, permitindo assim sua expansão na Web atual.

Referências Bibliográficas

ABREU, M. M; RODRIGUES, E.L.L – Sistema para identificação de Áudio utilizando Linux embarcado em microcontrolador ARM. Trabalho de Conclusão de Curso. 2012

ARM: The Architecture for the Digital World. Disponível em: <http://www.arm.com/> Acesso em 02 jul. 2013.

BERNERS, T.L; HENDLER, J.; LASSILA, O. The Semantic Web. **Magazine Scientific American.** Maio/2001**.** Disponível em: <http://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American_%20Feature%20Article_%20The%20Semantic%20Web_%20May%202001.pdf/> Acesso em: 03 jul. 2013.

CORTEX-M Series. ARM. Disponível em < http://www.arm.com/products/processors/cortex-m/index.php>. Acesso em 03.jul.2013

D2RQ, Accessing Relational Databases as Virtual RDF Graphs. Disponível em: <http>// http://d2rq.org/>. Acesso em 10 jul. 2013.

DBPedia. Disponível em: <http://wiki.dbpedia.org/About>. Acesso em 09 out. 2013.

DEPLOYING PHP applications using Virtuoso as Application Server. Disponível em <http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtuosoPHP#Developers:%20building%20Virtuoso%20Open-Source%20with%20PHP%20support>. Acesso em 04 Set. 2013.

GOMES, P; LEITE, T; CAETANO,U. A Arquitetura ARM. Projeto de Sistemas Computacionais, Universidade de Campinas, 2005.

HEBELER, J.;FISHER M.;BLACE R.; PEREZ-LOPEZ, A. Semantic Web Programming. United States of America: Wiley Publishing, 2009.

HORRIDGE, M. et al. Um guia prático para a construção de ontologias OWL, plugin

INSTRUMENTS, Texas. *MSP430 Ultra-Low Power 16-Bit Microcontrollers*. Disponível em: <http://www.ti.com/lsds/ti/microcontroller/16-bit_msp430/overview.page> Acesso em: 02 jul. 2013.

INTEL microcontrollers. Disponível em: <http://www.intel.com/design/embcontrol/> Acesso em 02 jul. 2013.

LEWIS, Daniel. Starting to use ODBC with PHP, 2011. Disponível em < http://www.ibm.com/developerworks/library/os-php-odbc/os-php-odbc-pdf.pdf>. Acesso em 04.Set. 2013

MANUAL da Linguagem PHP. Disponível em Disponível em <http://php.net/>. Acesso em 06.Set.2013.

MANUAL do OpenLink Virtuoso. Disponível em <http://docs.openlinksw.com/pdf/virtdocs.pdf>. Acesso em 10 jul. 2013.

MANUAL Olimex. Disponível em: <http://www.olimex.com/Products/ARM/Atmel/SAM9-L9260>. Acesso em 02 jul. 2013.

MICROCHIP, *16-bit PIC24 MCUs and dsPIC DSC*. Disponível em: <http://www.microchip.com/ja_jp/family/16bit/> Acesso em 02 jul. 2013.

MICROCHIP, *8-bit PIC Microcontrollers*. Disponível em: <http://www.microchip.com/ja_jp/family/8bit/index.html> Acesso em 02 jul. 2013.
MIPS Technologies. Disponível em: <http://www.mips.com/> Acesso em 02 jul. 2013.

NASCIMENTO, V.; RODRIGUES, E.L.L. – Desenvolvimento de Driver Linux para Antena nrf2401 para Sistemas com Linux Embarcado em microcontroladores ARM. Iniciação Científica LAVISIM 2013.

PASSOS, L.B.S.; RODRIGUES, E.L.L. – Automação usando sistemas operacionais Linux embarcados em microcontroladores ARM. Iniciação Científica LAVISIM 2011.

PHP SAPI module for Virtuoso. Disponível em <http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSBldPHP>. Acesso em 04. Set. 2013.

Protégé-OWL 3.4. Trad. SOARES, D.R.; ALMEIDA, M.B. 2008. Disponível em http://eci.ufmg.br/mba/. Acesso em 10. Set. 2013

SCIENTIFIC American Magazine. Disponível em: <http://www.scientificamerican.com/> Acesso em 03 jul. 2013.

UNICODE. Disponível em: <http://www.unicode.org/standard/standard.html/> Acesso em: 02 jul. 2013

VERTICAL Applications of Semantic Web. W3C. Disponível em: <http://www.w3.org/standards/semanticweb/applications>. Acesso em 03 jul. 2013.

Apêndice A – Códigos PHP

- **Página teste para conexão ODBC via PHP**

```
<?php
        //
        // ODBC Connection Variables
        //
        $o_DSN = 'VOS';
        $o_UID = 'dba';
        $o_PWD = 'lavisim';

        if (function_exists ('__virt_internal_dsn')) {
                $db = odbc_connect (__virt_internal_dsn(), null, null);
                echo "conexao efetuada";
        } else {
                $db = odbc_connect ($o_DSN, $o_UID, $o_PWD);
                echo "conexao efetuada";
        }

        if (!$db)
                error log ('odbc connect failed');
                echo "conexao nao efetuada";

        // .....

        odbc_disconnect ($db);
  ?>
```

Apêndice B – Arquivos RDF gerados pelo Protegé

- **Arquivo “classes.rdf”**

```
       <?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
        <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
        <!ENTITY patricia 'http://opencore.sel.eesc.usp.br/patricia/#'>
        <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
]>
<rdf:RDF xmlns:rdf="&rdf;"
        xmlns:patricia="&patricia;"
        xmlns:rdfs="&rdfs;">
<rdfs:Class rdf:about="&patricia;Avancados"
        rdfs:label="Avancados">
       <rdfs:subClassOf rdf:resource="&patricia;Comandos"/>
</rdfs:Class>
<rdfs:Class rdf:about="&patricia;Basicos"
        rdfs:label="Basicos">
       <rdfs:subClassOf rdf:resource="&patricia;Comandos"/>
</rdfs:Class>
<rdfs:Class rdf:about="&patricia;Comandos"
        rdfs:label="Comandos">
       <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="&patricia;Intermediarios"
        rdfs:label="Intermediarios">
       <rdfs:subClassOf rdf:resource="&patricia;Comandos"/>
</rdfs:Class>
<rdf:Property rdf:about="&patricia;code"
        rdfs:label="code">
       <rdfs:domain rdf:resource="&patricia;Comandos"/>
       <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&patricia;description"
        rdfs:label="description">
       <rdfs:domain rdf:resource="&patricia;Comandos"/>
       <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
```

```
<rdf:Property rdf:about="&patricia;keywords"
        rdfs:label="keywords">
       <rdfs:domain rdf:resource="&patricia;Comandos"/>
       <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&patricia;name"
        rdfs:label="name">
       <rdfs:domain rdf:resource="&patricia;Comandos"/>
       <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&patricia;timer"
        rdfs:label="timer">
       <rdfs:domain rdf:resource="&patricia;Comandos"/>
       <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
</rdf:RDF>
```

- **Arquivo “instances.rdf”**

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
        <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
        <!ENTITY patricia 'http://opencore.sel.eesc.usp.br/patricia/#'>
        <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
]>
<rdf:RDF xmlns:rdf="&rdf;"
        xmlns:patricia="&patricia;"
        xmlns:rdfs="&rdfs;">
<patricia:Basicos rdf:about="&patricia;functions_Class0"
        patricia:code="hora_data"
        patricia:description="Mostra a data e o horário do kit."

patricia:keywords=",hora,data,horadata,hours,date,datas,daat,hoar,taad,ra
ho,"
        patricia:name="Hora/Data"
        rdfs:label="Hora/Data"/>
<patricia:Basicos rdf:about="&patricia;functions_Class1"
```

```
        patricia:code="memoria"
        patricia:description="Exibe a memoria RAM do kit."
        patricia:name="Memoria"
        rdfs:label="Memoria">
       <patricia:keywords>,memoria,ram,memoria
ram,memom,memory,amr,basicas,informacoes,informações,informaçoes,</patric
ia:keywords>
</patricia:Basicos>
<patricia:Intermediarios rdf:about="&patricia;functions_Class10"
        patricia:code="restart"

patricia:keywords=",restart,reiniciares,reinicia,reinici,cinirei,"
        patricia:name="Reiniciar"
        rdfs:label="Reiniciar">
       <patricia:description>O comando reiniciar é responsável por
reiniciar o kit desejado.</patricia:description>
</patricia:Intermediarios>
<patricia:Intermediarios rdf:about="&patricia;functions_Class10001"
        patricia:code="shutdown"
        patricia:name="Desligar"
        rdfs:label="Desligar">
       <patricia:description>O comando desligar é responsável por
encerrar o funcionamento do kit desejado.</patricia:description>
       <patricia:keywords>,desligas,desligares,deslig,dormir,dorme,durma,
durmas,desigar,desligrare,</patricia:keywords>
</patricia:Intermediarios>
<patricia:Basicos rdf:about="&patricia;functions_Class10002"
        patricia:code="proc_exec"
        patricia:description="Exibe os processos em execução no kit."
        patricia:name="Processos"
        rdfs:label="Processos">
       <patricia:keywords>,processos,execução,execucao,execucão,execuçao,
exec,proc,processos,executa,</patricia:keywords>
</patricia:Basicos>
<patricia:Basicos rdf:about="&patricia;functions Class10003"
        patricia:code="hardware"
        patricia:description="Exibe o hardware que está sendo utlizado
para o projeto."
```

```
        patricia:name="Hardware"
        rdfs:label="Hardware">
       <patricia:keywords>,hardware,placa,kit,hardw,plac,itk,basicas,info
rmacoes,informações,informaçoes,</patricia:keywords>
</patricia:Basicos>
<patricia:Basicos rdf:about="&patricia;functions_Class10004"
        patricia:code="processador"
        patricia:description="Exibe o Processador do kit."
        patricia:name="Processador"
        rdfs:label="Processador">
       <patricia:keywords>,processador,microcontolador,processad,basicas,
informacoes,informações,informaçoes,</patricia:keywords>
</patricia:Basicos>
<patricia:Basicos rdf:about="&patricia;functions_Class10005"
        patricia:code="sistema_operacional"
        patricia:description="Exibe o Sistema Operacional instalado no
kit."
        patricia:name="Sistema Operacional"
        rdfs:label="Sistema Operacional">
       <patricia:keywords>,sistema,operacional,so,os,sistem,operation,bas
icas,informacoes,informações,informaçoes,</patricia:keywords>
</patricia:Basicos>
<patricia:Basicos rdf:about="&patricia;functions_Class10006"
        patricia:code="sistema_arquivos"
        patricia:description="Exibe o Sistema de Arquivos em
funcionamento no kit."
        patricia:keywords=",arquivos,sistema,systemfiles,files,system,"
        patricia:name="Sistema de Arquivos"
        rdfs:label="Sistema de Arquivos"/>
<patricia:Avancados rdf:about="&patricia;functions Class10007"
        patricia:code="pisca led"
        patricia:description="Pisca o LED do Kit Olimex."
        patricia:keywords=",pisca,LED,piscaLED,piscar,"
        patricia:name="Piscar Led"
        rdfs:label="Piscar Led"/>
</rdf:RDF>
```