Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

RAFAEL LONGO MACHADO

SISTEMA DE TEMPO REAL PARA MONITORAMENTO DE TEMPERATURA EM UM *DATA CENTER*

Assis/SP

2016

RAFAEL LONGO MACHADO

SISTEMA DE TEMPO REAL PARA MONITORAMENTO DE TEMPERATURA EM UM *DATA CENTER*

Projeto de pesquisa apresentado ao Curso de Ciências da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e à Fundação Educacional do Município de Assis – FEMA, como requisito para a obtenção do Certificado de Conclusão.

Orientando: Rafael Longo Machado

Orientador: Prof. Dr. Luiz Ricardo Begosso

Avaliador: Prof. Me. Fábio Eder Cardoso

Assis/SP

2016

FICHA CATALOGRÁFICA

Machado, Rafael Longo

Sistema de Tempo Real para Monitoramento de Temperatura em um *data center*/ Rafael Longo Machado. Fundação Educacional do Município de Assis – FEMA – Assis, 2016.

39p.

Orientador (a): Luiz Ricardo Begosso.

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA.

1. Sistema de Tempo Real. 2. RRDTool. 3. Raspberry Pi

CDD: 001.6
Biblioteca da FEMA

Sistema de Tempo Real para Monitoramento de Temperatura em um *data center*

RAFAEL LONGO MACHADO

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Bacharelado em Ciência da Computação, analisado pela seguinte comissão examinadora:

Orientador: Prof. Dr. Luiz Ricardo Begosso

Analisador: Prof. Me. Fábio Eder Cardoso

Assis

2016

RESUMO

Este trabalho tem como principal objetivo apresentar um estudo teórico sobre sistemas de tempo real e demonstrar a partir da ferramenta *RRDTool* a temperatura e umidade de um *Data Center*, utilizando o *Raspberry Pi* e a linguagem *Python*. Por meio dessas ferramentas será possível realizar o monitoramento de temperatura e umidade de um servidor.

Palavras-chave: Sistemas de Tempo Real; RRDTool; RaspBerry Pi.

ABSTRACT

This work has as main objective to present hum Theoretical Study on Real-Time Systems and demonstrate Breaking Tool RRDTool the temperature and humidity hum Data Center using the Raspberry Pi and Python. Through these tools you can perform temperature monitoring and humidity hum Server.

Keywords: Real Time Systems; RRDTool; Raspberry Pi.

Lista de Ilustrações

Sumário

1. Introdução

Este trabalho tem como objetivo principal realizar um estudo teórico sobre Sistemas de Tempo Real e, ao final, para simular um ambiente, será demonstrado uma ferramenta para monitoramento de temperatura e umidade de um *Data Center*. Um sistema de tempo real monitora um ambiente captando as amostras de dados durante sua operação e, após determinado processamento, toma decisão em tempo hábil para retornar uma resposta para o sistema.

Alguns sistemas de tempo real apresentam restrições de tempo mais rigorosas do que outros, entre esses se encontram os sistemas responsáveis por fazer monitoramento e sistemas de supervisão. De acordo com Oliveira *et al*.(2000), todas as aplicações que estão sujeitas a restrições temporais são agrupadas como Sistemas de Tempo Real.

Segundo Oliveira *et al.(*2000), os sistemas de tempo real podem ser classificados a partir do ponto de vista da segurança em: Sistemas Não Críticos de Tempo Real, quando as consequências de uma falha estão relacionadas ao tempo e a mesma ordem e grandeza que os benefícios dos sistemas em operação, e Sistemas Críticos de Tempo Real, quando as consequências de pelo menos uma falha excede os benefícios normais do sistema. Nesse caso a falha é dita catastrófica.

Para a demonstração da ferramenta proposta, será utilizado o *Raspberry Pi,* o qual é caracterizado como um pequeno computador de placa única que contêm os elementos principais de um computador usual. O ambiente de programação será elaborado no *Raspberry Pi*, no qual será realizado o monitoramento de temperatura.

No ambiente de programação, será utilizada a linguagem *Python,* a qual é recorrentemente utilizada para o desenvolvimento de ferramentas de monitoramento de temperatura. Além disso, *Python* é uma linguagem expressiva, na qual é fácil transmitir o raciocínio em um algoritmo.

Além disso, outra ferramenta a ser utilizada será o *Round Robin Database (RRDTool*) que tem a capacidade de armazenar séries de dados, a exemplo da temperatura e uso de CPU.

1.1. Objetivos

Este trabalho tem como principal objetivo realizar um estudo teórico sobre sistemas de tempo real e, ao final, demonstrar uma ferramenta para monitoramento de temperatura de um *Data Center*, utilizando o *Raspberry Pi*, a linguagem *Python* e o sistema de base de dados *RRDTool*. Por meio dessas ferramentas será possível realizar o monitoramento de temperatura e umidade de um servidor.

1.2. Relevância

O sistema para monitoramento do ambiente em *Data Center* é de extrema relevância para as empresas. Determinada falha nos sistemas de uma companhia impede o acesso ao banco de dados, às informações ou até mesmo o acesso à Internet, ou seja, paralisa a organização, podendo causar diversos prejuízos financeiros. As maiores falhas em *datas centers* são causadas pela interrupção de energia, pela falha de ar condicionado, bem como por uma refrigeração inapropriada.

Tendo isso em vista, é fundamental que todas as organizações adquiram um sistema de monitoramento de temperatura e umidade para *data centers*. Assim, cabe aos analistas de sistemas ou aos programadores desenvolverem estudos que diminuam os riscos de falha nos servidores das empresas. Nesse sentido, realizar-se-á o levantamento bibliográfico sobre estudos teóricos a respeito de Sistemas de Tempo Real a fim de se obter uma visão geral sobre este tema e realizar a demonstração de uma ferramenta para o monitoramento de temperatura e umidade de um *Data Center*.

1.3. Motivação

Falhas de refrigeração são comuns em empresas que não possuem sistemas de monitoramento. Desse modo, pretende-se desenvolver essa pesquisa a fim de auxiliá-las a evitarem falhas de refrigeração em *datas centers*, o que minimizaria diversos prejuízos na organização, a exemplo da paralisação dos serviços.

Além disso, as ferramentas a serem utilizadas no decorrer da pesquisa são financeiramente acessíveis. Também será possível montar o ambiente para o desenvolvimento do sistema com os recursos próprios.

1.4. Perspectivas de Contribuição

Tendo em vista os demais trabalhos, artigos e pesquisas que tem como principal conteúdo os Sistemas de Tempo Real, nota-se que já foram elaboradas diversas análises sobre este tema e suas implicações. Sendo assim, este trabalho contribuirá com estudos voltados à área de monitoramento de temperatura e umidade, uma vez que será feita a demonstração da ferramenta.

1.5. Metodologia

A realização do estudo teórico dar-se-á por meio de um levantamento bibliográfico que tem como principal objeto de estudo os Sistemas de Tempo Real.

A demonstração desse projeto ocorrerá na seguinte sequência: a) o *RaspBerry Pi* coletará as informações do sensor através de um script em *Python* e enviará para o RRDTool; b) o *RaspBerry Pi* mandará essas informações por meio da rede para o servidor; c) o *RRDTool* receberá as informações e armazenará em uma base de dados; d) utilizando tal base de dados, o *RRDTool* construirá um gráfico e disponibilizará via web.

1.6. Recursos Necessários

Os recursos necessários para o desenvolvimento deste trabalho são:

- Os trabalhos teóricos que tem como principal objeto de estudo os Sistemas de Tempo Real;
- O *RaspBerry PI;*
- A linguagem *Python;*
- O sensor de temperatura e umidade;
- A ferramenta *RRDTool*.

1.7. Estrutura do Trabalho

Capitulo 1: Introdução

Na Introdução é feita uma breve descrição sobre o que este trabalho pretende abordar, como também o seu tema, objetivos, sequência em que o projeto será desenvolvido e quais serão os recursos necessários para o desenvolvimento do trabalho.

Capitulo 2: Sistemas de Tempo Real

Neste capítulo encontram-se os métodos e conceitos utilizados em Sistemas de Tempo Real.

Capitulo 3: *RaspBerry Pi*

O foco deste capítulo é descrever sobre o *Raspberry Pi* e mostrar algumas de suas funções.

Capitulo 4: Python

Este capítulo tem como foco descrever, brevemente, o surgimento da linguagem *Python*.

Capitulo 5: *RRDTool*

Neste capítulo encontram-se os métodos e os conceitos para se utilizar a ferramenta *RRDTool*.

1.8. Cronograma

Tarefa	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago
Definição do Tema											
Coleta do Material											
Desenvolvimento do Pré-Projeto											
Entrega do Pré-Projeto											
Desenvolvimento do Texto de Qualificação											
Entrega da Qualificação											
Preparação para o Exame de Qualificação											
Exame de Qualificação											

Desenvolvimento do Projeto											
Apresentação do TCC ao Orientador											
Solicitação de Defesa											
Entrega do TCC											
Apresentação Final											

2. Sistemas de Tempo Real

2.1. O que é um Sistema de Tempo Real

Sistemas de tempo real são sistemas que trabalham sob restrições de tempo. De acordo com Oliveira *et al*.(2000), esse tipo de sistema consiste em subsistemas de controle e subsistemas controlados (ambientes físicos) que se interagem por meio de três operações: tiragem de amostras, processamento e respostas.

Primeiramente, o subsistema de controle tira amostras de dados do ambiente físico durante a sua operação normal. Em seguida, os dados são imediatamente processados, retornando uma resposta para o ambiente. Um exemplo seria um sistema que controla as pernas de um robô, que responde continuamente às mudanças do ambiente físico num determinado intervalo de tempo. Caso o controle das pernas do robô responda no tempo errado, ele cairá.

Segundo a *Oxford Dictionary of Computing*, “qualquer sistema para o qual é importante o instante em que a saída é produzida”. Isto porque, normalmente, a entrada corresponde a algum movimento no mundo físico, e a saída tem que se relacionar com esse mesmo movimento. O atraso entre os instantes de entrada e de saída deverá ser suficientemente pequeno para obter um desempenho razoavelmente atempado.

Segundo o dicionário citado, Sistemas de Tempo Real podem ser classificados em:

- *Hard Real Time*: devem seguir as restrições de tempo para evitar consequências catastróficas e estão normalmente relacionados à vida das pessoas. Como exemplo, podem-se citar sistemas de controle de avião e sistemas de controle de processos químicos.

- *Soft Real Time*: sistemas que podem continuar funcionalmente corretos mesmo que restrições temporais não sejam respeitadas. Alguns exemplos são os sistemas de aquisição de dados e os sistemas de reserva de passagens aéreas.

2.2. Conceitos e Caracterização de um Sistema de Tempo Real

Um *Sistema de Tempo Real (STR)* é um sistema computacional que deve reagir a estímulos do seu ambiente em prazos específicos.

O atendimento desses prazos resulta em requisitos de natureza temporal sobre o comportamento desses sistemas. Como consequência, em cada reação, o sistema de tempo real deve entregar um resultado correto dentro de um prazo específico, sob pena de ocorrer uma falha temporal. O comportamento correto de um sistema de tempo real, portanto, não depende só da integridade dos resultados obtidos (correção lógica ou *"correctness"*), mas também dos valores de tempo em que são produzidos (correção temporal ou *"timeliness"*). Uma reação que ocorra além do prazo especificado pode não ter utilidade ou até mesmo representar uma ameaça. (OLIVEIRA et. al. 2000).

De acordo com Farines, "a maior parte dos sistemas de tempo real se comportam como sistemas reativos com restrições temporais". A reação dos sistemas de tempo real aos eventos vindos do ambiente externo ocorre em tempos compatíveis com as exigências do ambiente e mensuráveis na mesma escala de tempo. A concepção do sistema de tempo real está diretamente ligada ao ambiente, o qual, por sua vez, está relacionado com o comportamento temporal do mesmo. Na classe de Sistema de Tempo Real encontram-se os sistemas embutidos (*"Embedded Systems"*) e os sistemas de supervisão e controle, os quais distinguem-se entre o Sistema a Controlar, o Sistema Computacional de Controle e o Operador. O Sistema a Controlar e o Operador são considerados como o Ambiente do Sistema Computacional. A interação entre os mesmos ocorre por meio de interfaces de

instrumentação (compostas de sensores e atuadores) e da interface do operador. (FRAGA et. al. 2000).

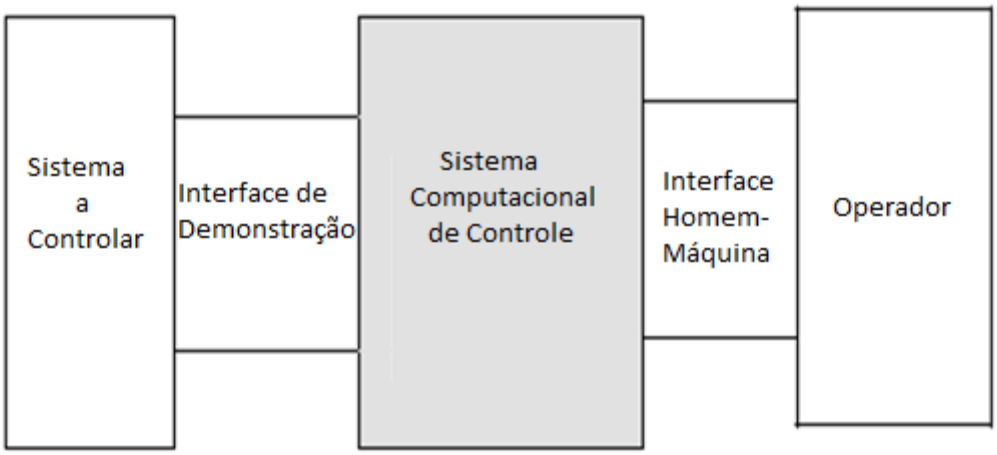


Figura 1 - Tipo de Sistema de Tempo Real

Fonte: (OLIVEIRA, Rômulo Silva de.; FARINES, Jean-Marie.; FRAGA, Joni da Silva. Sistemas de Tempo Real. Florianópolis, 2000.)

2.3. Previsibilidade

Quando uma ação computacional se desvia do que foi especificada ou esperada, o sistema se mostra defeituoso. Defeitos, definidos como estados incorretos do sistema, por sua vez, são causados por falhas. Um defeito ocorre na dimensão que o sistema produz resultados (valores) fora de sua especificação. Na dimensão temporal um sistema pode produzir resultados *antecipadamente* ou em *atraso*. Se algum resultado nunca é produzido, diz-se que o sistema apresenta defeitos por *omissão*. Esse tipo de defeito pode ser considerado pertencente tanto à dimensão funcional (resultados nulos são produzidos) quanto à temporal (o sistema está infinitamente atrasado). Omissões persistentes são chamadas de defeitos tipo *crash*. O tipo mais genérico de defeito que um sistema pode apresentar é chamado de *bizantino* ou arbitrário. Nesse caso, o sistema pode apresentar comportamento completamente imprevisível em ambas as dimensões.

Construir um sistema de tempo real 100% previsível é, portanto, evitar seus possíveis defeitos. Como é impossível construir-se tal sistema perfeito, na prática, busca-se minimizar a probabilidade da ocorrência de defeitos, tanto funcionais quanto temporais.

Isso pode ser feito por meio de diversas técnicas complementares, seja durante a especificação do sistema, da sua programação ou da implementação de mecanismos que, em tempo de execução, aumentam a confiança no funcionamento do sistema (KOPETZ, et.al.1998).

Erros durante a fase de concepção do sistema tendem a se propagar durante sua construção ou implementação e execução. Portanto, é importante detectar tais erros antes mesmo do sistema ser construído. Algumas técnicas, tais como prototipação e testes, auxiliam nessa tarefa. No caso de sistemas de tempo real, apenas essa abordagem não é o bastante. Outra alternativa é o uso de ferramentas baseadas em formalismos, por meio das quais os comportamentos funcional e temporal do sistema possam ser provados. Só então é possível certificar-se de que a especificação do sistema está correta de acordo com critérios de correção identificados (KOPETZ, et.al.1998).

3. RaspBerry Pi

3.1 O que é um *RaspBerry Pi*

A placa *Raspberry PI* é um computador de baixo custo e de pequenas dimensões. Criado pela *Raspberry Pi Foundation* com sede no Reino Unido, seu desenvolvimento faz parte de um projeto que tem como intuito promover o campo da computação e o funcionamento dos computadores desde cedo para os jovens, funcionando, assim, como um gerador de futuros profissionais a fim de garantir e contribuir com o desenvolvimento deste campo científico[1].

Seu pré-lançamento foi feito com apenas 50 unidades em agosto de 2011, sendo disponibilizadas para a venda em abril de 2012 e, a partir desta data, as vendas decolaram devido ao seu baixo custo e todo o poder que elas proporcionavam. Rapidamente ela se tornou referência na comunidade de desenvolvedores, muitas aplicações e projetos apareceram como servidores *web*, automatização residencial e centrais de mídia[2].

---

[1] **FÓRUM da Raspberry Pi Foundation para usuários da plataforma**. Disponível em: <http://www.raspberrypi.org/forums>. Acesso em: 28 jan.2016.

[2] **FAQS | Raspberry Pi. Raspberry Pi**, Disponível em: <http://www.raspberrypi.org/help/faqs>. Acesso em: 02 jan.2016.

Na figura 3 estão indicadas as principais características que fizeram da *RaspBerry PI* um sucesso, como a saída *HDMI* em alta definição e o processador *ARM11*.

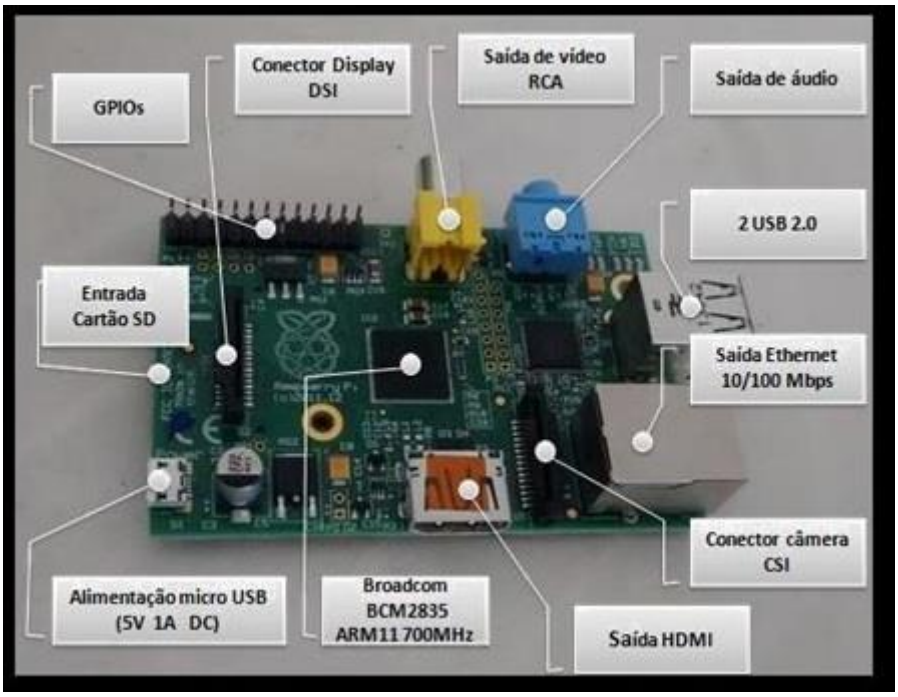


Figura 2 - Descrição dos componentes da *RaspBerry Pi*

Fonte: (BEGIDO, Igor Jovedi. Plataforma Móvel Para Monitoramento Remoto por Câmera Comandada por Aplicação em Ambiente WEB. Universidade de São Paulo-USP, TCC-2014).

Para seu funcionamento são necessários os mesmos periféricos de um computador comum como mouse, teclado e monitor, ou seja, um computador que cabe literalmente na palma da mão.

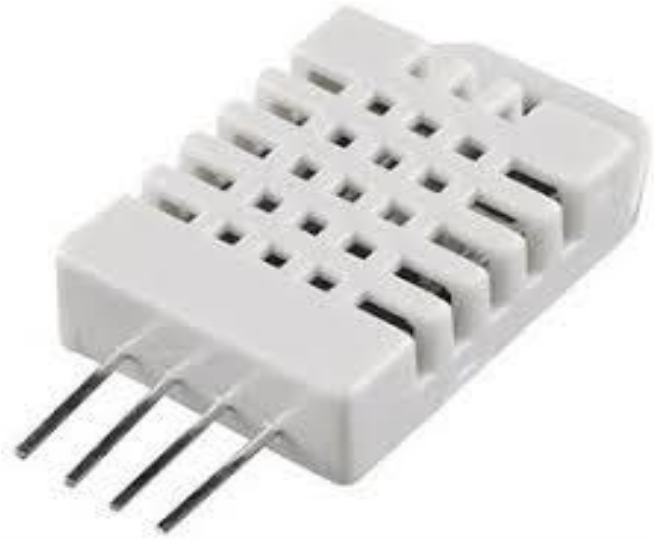


Figura 3 - Exemplo de um Sensor de Temperatura para *RaspBerry PI.*

Fonte: (http://raspibr.com.br/wp-content/uploads/2015/01/humidity-temperature-sensor-dht22-500x500.jpg)

3.2. Sistema Operacional

O *Raspberry Pi* utiliza o Linux e também o Free BSD como sistema operacional. O Linux é tecnicamente apenas o *kernel*, porém um sistema operacional é muito mais do que isso, composto por drivers, serviços e aplicações. Uma diversidade de distribuições de Linux tem sido desenvolvida ao longo dos anos. Algumas das mais comuns em computadores desktop são *Ubuntu*, *Debian*, *Fedora* e *Arch*. Cada uma delas tem as suas próprias comunidades de usuários e são ajustadas para aplicações específicas. Em virtude de o computador *Pi* ser baseado em um *chipset* de dispositivo móvel, ele tem requisitos de software diferentes de um computador desktop. O processador *Broadcom* tem algumas características próprias, que exigem drivers de dispositivos especiais (“binary blob”) e programas que não estão incluídos em nenhuma distribuição Linux padrão. Enquanto a

maioria dos computadores desktop tem gigabytes de memória RAM e centenas de gigabytes de armazenamento, o *Pi* é mais limitado em ambos os aspectos.

3.3. Raspbian

O *Raspbian* é um sistema operacional baseado no Debian otimizado para o *Raspberry Pi.* Ele disponibiliza mais de 35.000 pacotes pré-compilados para o melhor desempenho possível no *Raspberry Pi*. O *Raspbian* é encontrado no site do *Raspberry Pi. (RICHARDSON. et.al.2013*).

Esse sistema operacional é recomendado para quem tem menos conhecimentos dos sistemas Linux ou simplesmente queira um sistema pronto a funcionar. Além disso, é um sistema quase completo, pois apresenta diversas aplicações pré-instaladas, drivers usuais, ferramentas facilitadoras para as configurações necessárias, entre outros. Muitas aplicações e módulos dedicados à programação já estão inclusos na imagem do *Raspbian*, dessa forma basta iniciar o sistema para acessá-los. O *Raspbian* é mais recomendável para iniciantes com o *RaspBerry Pi* que desejam experimentar as potencialidades ou começar a programar e desenvolver projetos de sistemas embarcados,.

4. Python

4.1. O que é Python

Python é uma linguagem *open source* (linguagem de uso livre) de programação de alto nível, isto é, mais próxima da linguagem humana, interpretada, imperativa, orientada a objetos, com tipagem dinâmica e forte. Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos *Python Software Foundation*.

4.2. História da Linguagem

Python foi desenvolvida no final de 1989 pelo inglês Guido van Rossum do Instituto de Pesquisa Nacional para Matemática e Ciência da Computação (CWI), nos Países Baixos, como um sucessor da linguagem ABC capaz de tratar exceções e prover interface com o sistema operacional Amoeba por meio de scripts. Um dos focos primordiais de *Python* era aumentar a produtividade do programador.

Em 1991, Guido publicou o código (versão 0.9.0) no grupo de discussão *alt.sources*. Nessa versão já estavam presentes classes com herança, tratamento de exceções, funções e os tipos de dado nativos *list*, *dict, str*, e assim por diante.

A versão 1.0 foi lançada em janeiro de 1994. Novas funcionalidades incluíam ferramentas para programação funcional como *lambda, map, filter* e *reduce*. A última versão enquanto Guido estava na CWI foi o *Python* 1.2. Em 1995, ele continuou o trabalho no CNRI em Reston, Estados Unidos, de onde lançou diferentes versões. Na versão 1.4, a linguagem ganhou a capacidade de passar parâmetro pelo nome e não pela posição na lista de parâmetros e suporte nativo a números complexos, assim como uma forma de encapsulamento.

Em 2000, o time de desenvolvimento da linguagem se mudou para a *BeOpen* a fim de formar o time *PythonLabs*. A CNRI pediu que a versão 1.6 fosse lançada, para marcar o fim do desenvolvimento da linguagem naquele local. O único lançamento na *BeOpen* foi o *Python* 2.0, e após o lançamento o grupo de desenvolvedores da *PythonLabs* agrupou-se na *Digital Creations*.

Python 2.0 implementou *list comprehension*, uma relevante funcionalidade de linguagens funcionais como *SETL* e *Haskell*. A sintaxe da linguagem para essa construção é bastante similar à de *Haskell*, exceto pela preferência do *Haskell* por caracteres de pontuação e da preferência do *Python* por palavras reservadas alfabéticas. Essa versão 2.0 também introduziu um sistema coletor de lixo capaz de identificar e tratar ciclos de referências.

Python 2.1 era parecido com as versões 1.6.1 e 2.0. Sua licença foi renomeada para *Python Software Foundation License*.

A terceira versão da linguagem foi lançada em dezembro de 2008, chamada *Python* 3.0 ou *Python* 3000. Como noticiado desde antes de seu lançamento, houve quebra de compatibilidade com a família 2.x para corrigir falhas que foram descobertas neste padrão, e para limpar os excessos das versões anteriores. A primeira versão alfa foi lançada em 31 de agosto de 2007, a segunda em 7 de dezembro do mesmo ano.

Atualmente a linguagem é usada em diversas áreas, como servidores de aplicação e computação gráfica. Está disponível como linguagem de *script* em aplicações como *OpenOffice* (*Python UNO Bridge*), *Blender* e pode ser utilizada em procedimentos armazenados no sistema gerenciador de banco de dados *PostgreSQL* (*PL/Python*).

4.3. Características

Em *Python*, diferentemente de C++ ou Java, as funções são tratadas como objetos, característica de linguagens de programação funcional como Lisp, muito utilizada em aplicações de inteligência artificial. Outro ponto importante, oferecendo grande flexibilidade, é que, em *Python*, cada argumento de uma função pode assumir um valor *default.*[3]

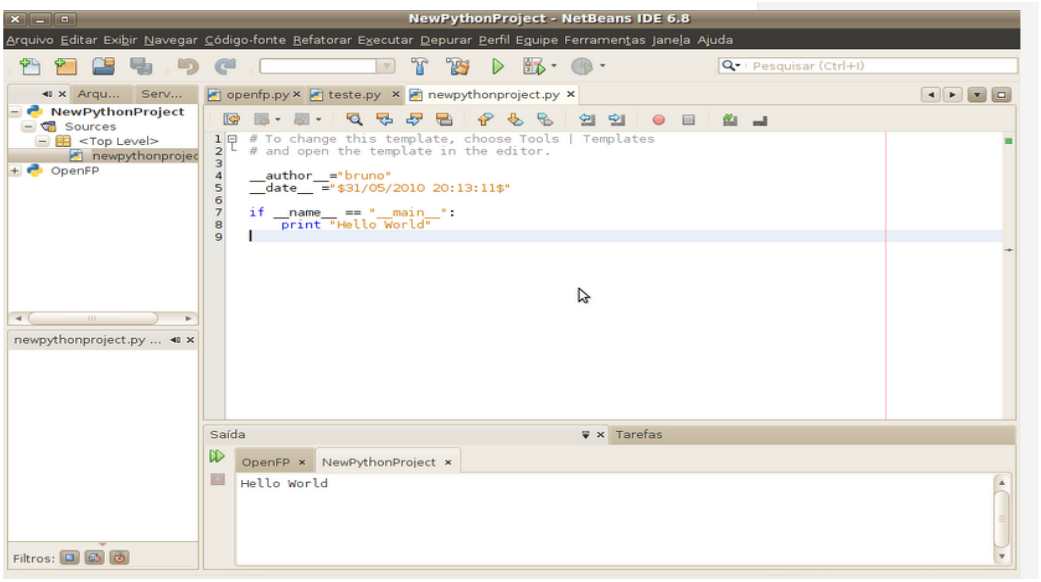


Figura 4 - Ambiente de Programação para Linguagem *Python*

Fonte:(http://1.bp.blogspot.com/_y_bGxMwbE9g/TLhPTMh2eiI/AAAAAAAAA_0/t3E63UkZyyc/s1600/netbeans_para_python1.JPG).

[3] **GUIA Python**. Disponível em <http://www.python.org.br>. Acesso em: 20 jan.2016.

5. RRDTool

5.1. O que é RRDTool

Escrita por Tobias Oetiker, sobre a licença *General Public License* (GNU) o *RRDTool* é uma ferramenta código aberto que gerencia um banco de dados do tipo Round Robin. Por sua função ser o armazenamento de dados temporais, o *RRDTool* é utilizado em grande escala para armazenar dados da utilização de banda em uma rede, taxa de requisições a um servidor, entrada/saída de pacotes, entre outros. Round Robin é uma técnica que trabalha com uma quantidade fixa de dados e um ponteiro para o elemento atual conforme exibido na Figura 6, onde *t0* até *t5* são os elementos do banco dispostos em intervalos fixos no tempo. O elemento atual é lido ou escrito e, após um intervalo predefinido de tempo, o ponteiro é movido para o próximo elemento. Assim, na criação da base de dados, é especificado o tamanho fixo da mesma que não sofrerá alteração durante seu tempo de vida. Ou seja, os dados são inseridos nas posições livres e quando não houver mais espaço, os dados mais antigos são sobrescritos, numa espécie de fila circular. Os dados são armazenados neste tipo de banco para que as leituras sejam regulares ao longo do tempo.

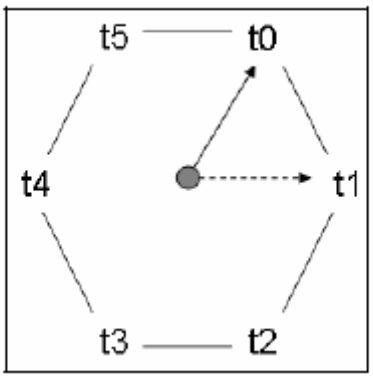


Figura 5 - Técnica Round Robin

Fonte: (Alcides, Luciano de. Monografia: Gerência de redes. Trabalho de conclusão de curso na Universidade de Pernambuco, 2006/2. 54p.)

5.2. Características

A ferramenta *RRDTool* pode ser utilizada para a coleta e armazenamento de qualquer tipo de dados que se apresentem em séries temporais, ou seja, é um mecanismo capaz de capturar o valor de uma variável em diversos intervalos de tempo. Os dados capturados são automaticamente interpolados com os dados anteriores, utilizando funções de consolidação, onde um valor resultante é armazenado (Alcides, 2006).

As principais características do *RRDTool* que o classificam como uma ferramenta ideal para coleta e apresentação de dados em intervalos fixos de tempo são:

- *RRA (Round Robin Arquives)*: Os valores dos elementos com as mesmas configurações de consolidação são armazenados em um mesmo RRA. Esta é uma forma eficiente de armazenar os valores, uma vez que sabemos ser fixo o tamanho de armazenamento. Por exemplo, se quisermos armazenar 1000 (mil) valores em

um intervalo de 5 minutos, a ferramenta *RRDTool* irá alocar espaço para os mil valores e mais uma área para cabeçalho. Este cabeçalho armazena o ponteiro que contém o endereço do último valor que foi escrito no banco. Os novos valores são então armazenados por meio do mecanismo de *Round Robin*. A utilização de *Round Robin Arquives* garante que o banco de dados não cresça ao longo do tempo e que dados antigos sejam automaticamente eliminados. A utilização de funções de consolidação nos permite armazenar os dados que realmente interessam, tais como: o número máximo de bits por segundo de entrada em um roteador, a média de utilização de CPU de um servidor de *streaming*, número máximo de requisições a um servidor Web, etc.. (Alcides, 2006)

- *Unknown Data* (dados desconhecidos): Como mencionado, o *RRDTool* armazena dados em intervalos fixos de tempo. Pode acontecer de não haver informação disponível no momento em que os dados devem ser inseridos no banco. Neste caso, a aquisição de dados não é possível. Comumente, estes problemas ocorrem por queda momentânea da rede. O *RRDTool* trata esses casos inserindo um valor *unknown* no banco de dados. Na consolidação de um conjunto de dados, a quantidade de valores *unknown* é calculada e quando um novo valor consolidado está pronto para ser inserido no banco *Round Robin Arquives*, é verificado se a porcentagem destes valores *unknown* está acima de um limite configurável. Se não estiver, um valor *unknown* é armazenado. Esta característica do *RRDTool* assume que é melhor armazenar um valor desconhecido do que um valor 0 (zero) ou qualquer outro valor válido no banco, uma vez que um valor válido poderia alterar as métricas no momento da consolidação dos dados (Alcides, 2006).

O *RRDTool* permite a geração de relatórios tanto de forma numérica como gráfica, tendo como base os dados armazenados em 1 (um) ou vários bancos RRD. Um dos nossos objetivos é apresentar os dados sobre todas as variáveis que iremos monitorar utilizando esta ferramenta gráfica. Por meio de scripts *shell*, podemos definir tamanho, cor e legendas dos gráficos. Esta função também nos permite redefinir os valores das entradas

de dados, para que assim possamos transformar bits em bytes, por exemplo, ou até realizar operações mais específicas como calcular a variância dos dados de entrada.

5.3. Funções do RRDTool

Create é a função usada para a criação de novos bancos de dados RRD (*Round Robin Database*). Para criação dos arquivos RRD devemos seguir o código abaixo:

```
rrdtool create nome_do_arquivo
[--start-time tempo_inicial]
[--step intervalo]
[DS: nome-ds:DST:heartbeat:min:max]
[RRA:CF:xfiles:step:rows]
```

Fonte: (Alcides, Luciano de. Monografia: Gerência de redes. Trabalho de conclusão de curso na Universidade de Pernambuco, 2006/2. 20p.)

No código para criação do banco *RRDTool* abaixo, pode-se observar os valores da memória consumida pelo servidor:

```
1 #! /bin/sh
2
```

```
3 $tempo_atual = $(date +%s)

4

5 ./rrdtool create memoria.rrd --start $tempo_atual –step 60

6 DS:mem_tot:GAUGE:120:U:U RRA:AVERAGE:0.5:1:1440
```

Fonte: (Alcides, Luciano de. Monografia: Gerência de redes. Trabalho de conclusão de curso na Universidade de Pernambuco, 2006/2. 20p.)

Na linha 3, a variável $tempo_atual recebe a hora atual da estação de gerenciamento em segundos.

A função usada para o armazenamento de dados no banco RRD é chamada de *update*. Deve-se utilizar a sintaxe abaixo para armazenar dados nos bancos:

```
rrdtool update nome_do_arquivo

[tempo_1]:[valor_1]

[tempo_2]:[valor_2]

[tempo_n]:[valor_n]
```

Fonte: (Alcides, Luciano de Monografia: Gerência de redes. Trabalho de conclusão de curso na Universidade de Pernambuco, 2006/2. 23p.)

O trecho do código utilizado abaixo apresenta o armazenamento dos dados que representa o consumo de memória pelo servidor:

```
1 #! /bin/sh

2

3 $tempo_atual = $(date +%s)

4

5 ./rrdtool update memoria_stream.rrd $tempo_atual:$memoria
```

Fonte: (Alcides, Luciano de. Monografia: Gerência de redes. Trabalho de conclusão de curso na Universidade de Pernambuco, 2006/2. 23p.)

Na linha 5, memoria_stream.rrd é o nome do arquivo RRD. O próximo parâmetro define os valores de consumo de memória, em *KBytes*, contidos na variável $memoria, que devem ser armazenados nos espaços disponíveis do banco determinados pela variável `$tempo_atual`.

A função que consulta dados armazenados no banco RRD em um intervalo de tempo é chamada de *Fetch*. Devemos utilizar a sintaxe abaixo para consultar dados nos bancos:

```
rrdtool fetch nome_do_arquivo

[Funcao_Consolidacao]

[--start tempo_inicial]

[--end tempo_final]
```

Fonte: (Alcides, Luciano de. Monografia: Gerência de redes. Trabalho de conclusão de curso na Universidade de Pernambuco, 2006/2. 24p.)

A função usada para a criação de gráficos com o RRD é chamada de *Graph*, além disso ela possibilita realizar cálculos aritméticos nos dados do banco. Esta é uma das funções que será utilizada na demonstração da ferramenta.

A função *graph* necessita primeiramente de uma fonte de dados. Então, devemos utilizar as chamadas definições de dados para indicar o banco RRD onde devem ser coletados os dados. Segue a sintaxe para definição de dados:

```
DEF:[nome_var]=[arquivo_rrd]:[nome-ds]:[CF]
```

Fonte: (Alcides, Luciano de. Monografia: Gerência de redes. Trabalho de conclusão de curso na Universidade de Pernambuco, 2006/2. 25p.)

A opção nome_var é o nome da variável que representa a fonte de dados. As opções rrdfile, ds-name e CF são: o nome do banco RRD, o nome da fonte de dados do banco e a função.

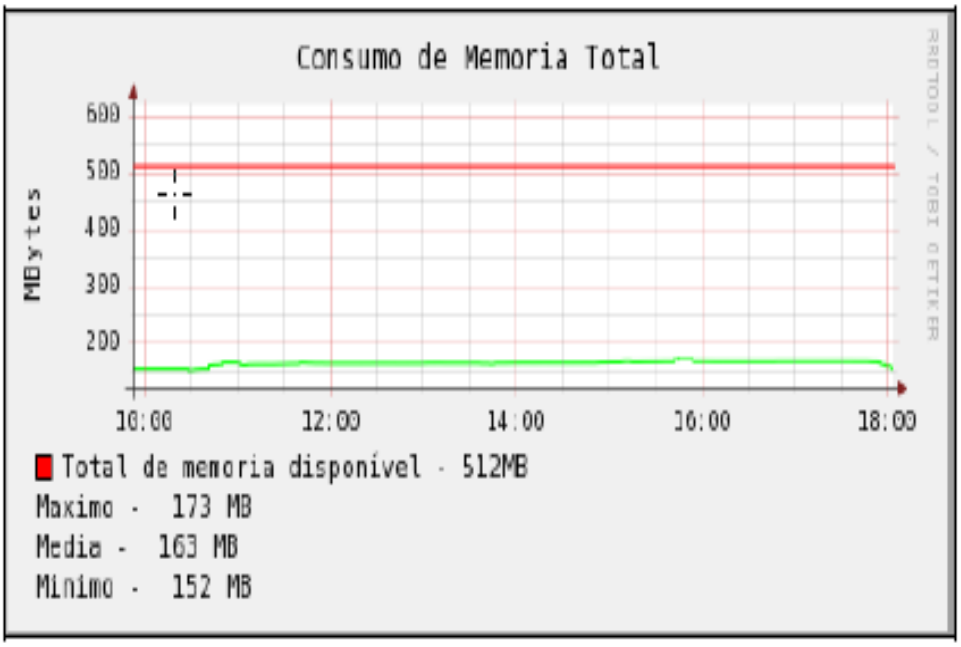


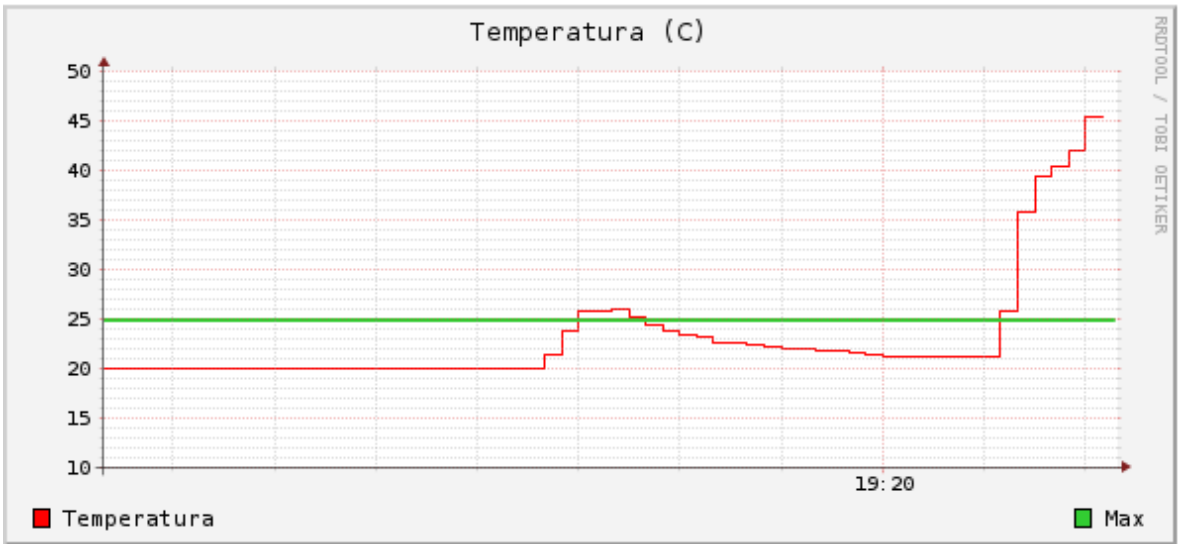
Figura 6 – Consumo de Memória Total utilização da função *graph***;**

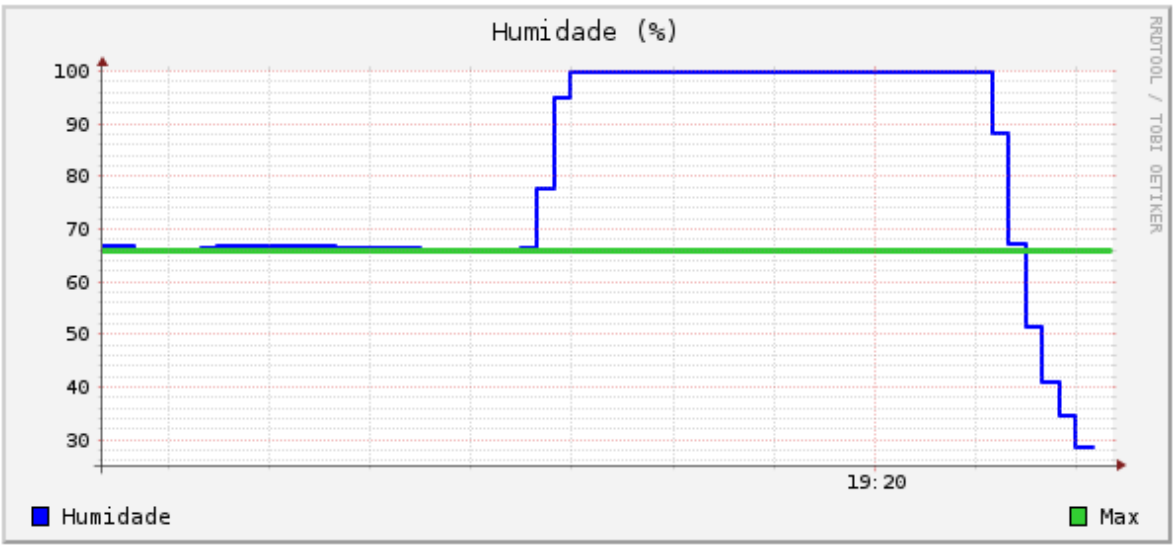
Fonte: (Alcides, Luciano de. Monografia: Gerência de redes. Trabalho de conclusão de curso na Universidade de Pernambuco, 2006/2. 54p.)

ESTUDO DE CASO

Para o desenvolvimento da pesquisa foram utilizadas as seguintes ferramentas: *RaspBerry Pi*, *RRDTool*, Sensor de Temperatura e Umidade. A linguagem de programação utilizada para a comunicação com estas ferramentas foi *Python*.

Para a demonstração do monitoramento, o sensor de temperatura e umidade ligado ao *RaspBerry Pi* capturou os dados do ambiente físico e armazenou na base de dados *RRDTool*, a qual gerou um gráfico de temperatura e outro de umidade.





Podemos observar nos gráficos de Temperatura e Umidade algumas alterações no decorrer do tempo. A primeira modificação ocorre quando a temperatura aumentou de 20ºC para 26ºC e, no mesmo instante, a umidade passou de 68% para 100%. Isto ocorreu devido ao assopro feito próximo ao sensor de temperatura e umidade. Após alguns minutos, a temperatura e a umidade normalizaram-se. Em seguida, ocorreu a segunda modificação. A temperatura aumentou de 20ºC para 25ºC e, ao mesmo tempo, a umidade diminuiu de 68% para 30%, pois um secador de cabelo foi ligado próximo ao sensor de temperatura e umidade, lançando ar quente e seco.

CONSIDERAÇÕES FINAIS

Ao término do presente trabalho, pode-se observar o nível de importância que o monitoramento de temperatura e umidade proporciona às empresas. É possível constatar que corporações que não possuem esse tipo de sistema não estão seguras a quaisquer tipos de falhas de refrigeração. Sendo assim, faz-se cada vez mais necessário que as organizações tenham um sistema de monitoramento para que não sejam prejudicadas financeiramente.

Tendo isso em vista, esse trabalho contribuirá com os estudos voltados à área de monitoramento de temperatura e umidade.

O ar refrigerado é indispensável em uma sala de servidores, pois o calor em excesso pode levar as máquinas ao superaquecimento. Esse gráfico foi gerado a partir das ferramentas utilizadas para a demonstração do monitoramento em um ambiente fechado com a temperatura média de 21 ºC, a qual é considerada ideal. Em ambientes sem refrigeração os servidores podem superaquecer.

Nessa pesquisa a faixa de umidade varia entre 55% a 60%. A faixa de umidade ideal de operação de cada datacenter é determinado pelo fabricante do produto, mas considera-se, de forma conservadora, que a umidade de um datacenter deve ser relativa a 45% e 65%. Em locais úmidos, conservar um desumidificador na sala contribui para manter as máquinas secas e livres de fungos, que podem prejudicar os servidores.

REFERÊNCIAS

ALCIDES, Luciano de. Monografia: **Gerência de redes. Trabalho de conclusão de curso na Universidade de Pernambuco**, 2006/2. 54p.

BEGIDO, Igor Jovedi. **Plataforma Móvel Para Monitoramento Remoto por Câmera Comandada por Aplicação em Ambiente WEB.** Universidade de São Paulo-USP, TCC-2014.

BURNS, Alan e Wellings, Andrew. **J.Real-Time SystemsAnd Programming Languages**, Addison Wesley, 2009.

COLNAGO, Guilherme Piazentini. **Desenvolvimento e Implementação de um Sistema de Monitoramento em Tempo Real da Tensão da Rede com Acesso Remoto**. 2009. 146p. Dissertação (Mestrado) – Centro Tecnológico – Universidade Federal do Espirito Santo, Espirito Santo, 2009.

H. Kopetz. ***Real-Time Systems Design for Distributed Embedded Applications***. Kluwer Academic Publishers, 1997.

H. Kopetz. **The Time-Triggered Model of Computation**. In *Proc. of the 19th Real-Time Systems Symposium (RTSS)*, pages 168–177. IEEE Computer Society Press, 1998.

MACHADO, José Araujo.; BINDA, Janaina Costa. **Monitoramento de Rede nas Instituições de Ensino Superior**. 2013, Assis, Brasil.

OLIVEIRA, Rômulo Silva de. FARINES, Jean-Marie.; FRAGA, Joni da Silva. **Sistemas de Tempo Real**. Florianópolis, 2000.

PYNE, Sandra. **Oxford: Dictionary of Computing. 1.ed.** Oxford University: Oxford, 1996. 394p.

R. Milner, ***A Calculus of Communicating Systems*****, Lecture Notes in Computer Science**, vol 92, 1980, Ed. Springer-Verlag.

RICHARDSON, Matt. WALLACE, Shawn. **Getting Started with Raspberry Pi**, 2013, Ed. Media Inc.

TANENBAUM, ANDREW S. **Redes de Computadores**. 3. ed. Campus, 1997. p.875.

FAQS | Raspberry Pi. Raspberry Pi, Disponível em: <http://www.raspberrypi.org/help/faqs>. Acesso em: 02 jan.2016.

FÓRUM da Raspberry Pi Foundation para usuários da plataforma. Disponível em: <http://www.raspberrypi.org/forums>. Acesso em: 28 jan.2016.

GUIA *Python*. Disponível em <http://www.python.org.br>. Acesso em: 20 jan.2016.

RRDTOOL, Oetiker, T.(2009). **OETIKER+PARTNER**, Disponível em: <http://oss.oetiker.ch/rrdtool>. Acesso em: 01 fev.2016.